ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი

ლექციათა კურსი საგანში

ოპერაციული სისტემები მობილური მოწყობილობებისათვის

მოწვეული ლექტორი **დ.ჭოხონელიძე**

2012

სარჩევი

1.0 საჭირო კომპონენტების ინსტალაცია 4
1.1 JDK ს ინსტალაცია 5
1.2 Eclipse ის ინსტალაცია 7
1.3 Android SDK ს ინსტალაცია 9
1.4 Android plugin ის ინსტალაცია Eclipse გარემოში 9
1.5 Android SDK Manager ით სხვადასხვა ვერსიების SDK ს დაყენება 12
1.6 AVD (Android Virtual Device) კონფიგურაცია 13
1.7 პირველი მაგალითი, პროექტის შექმნა და გაშვება 17
1.8 პროექტის შემადგენელი ნაწილების დეტალური განხილვა 21
1.9 კონფიგურაცია ნამდვილი Device ისთვის 25
2.0 Activities 30
2.1 Activity Life Cycle 30
2.2 Activity ს მონიტორინგი კატეგორიებად 33
3.0 Intent ობიექტი 33
3.1 ახალი Activity ის დამატება 33
3.2 კონფიგურაციის ფაილში შესაბამისი მეტა ინფორმაციის ჩამატება 35
3.3 შესაბამისი ვიზუალური შრის დამატება 35
3.4 Intent ობიექტის შექმნა და ახალი Activity ის გამოძახება 37
3.5 პარამეტრების გაყოლება Intent ობიექტზე 38
3.6 ახალი Activity ს გამოძახება დასრულების მოლოდინში 38
4.0 Layout ები 41
4.0.1 Linear Layout 42
4.0.2 Absolute Layout 45
4.0.3 Table Layout 45
4.0.4 Relative Layout 47
4.0.5 Frame Layout 49
4.0.6 ScrollView 51
4.1 სხვადასხვა ხედის რეჟიმები 54
4.2 Anchoring რეჟიმის გამოყენება 54
4.3 მდგომარეობის შენახვა რეჟიმის ცვლილებისას 56
4.4 Layout ების პროგრამულად დამატება 58

4.5 სენსორის ხდომილებების “მოსმენა“ 59
4.6 ხდომილებების დარეგისტრირება სხვადასხვა View ებისთვის 60
5.0 ვიზუალური კომპონენტები (შესავალი) 61
5.1 ზოგიერთი Basic View 61
5.2 ListView 64
5.3 SpinnerView 66
6.0 Data Persistence 67
6.1 SharedPreference ების გამოყენება 67
6.2 მონაცემთა ბაზის გამოყენება ინფორმაციის შესანახად 68
6.2.1 CRUD ის რეალიზაცია 74
7.0 Content Provider ები 78
7.1 სტანდარტული Content Provider ების გამოყენება 78
7.2 საკუთარი Content Provider ების შექმნა 79
8.0 Messaging და Networking 83
8.1 Messaging 83
8.2 Networking 88

1.0 საჭირო კომპონენტების ინსტალაცია

აუცილებელ/საჭირო კომპონენტებს მიეკუთვნება შემდეგი :

1. JDK http://www.oracle.com/technetwork/java/javase/downloads/index.html
2. Eclipse http://eclipse.org/downloads/
3. Android SDK http://developer.android.com/sdk/index.html
4. Android Plugin For Eclipse (Upldate URL: https://dl-ssl.google.com/android/eclipse/)

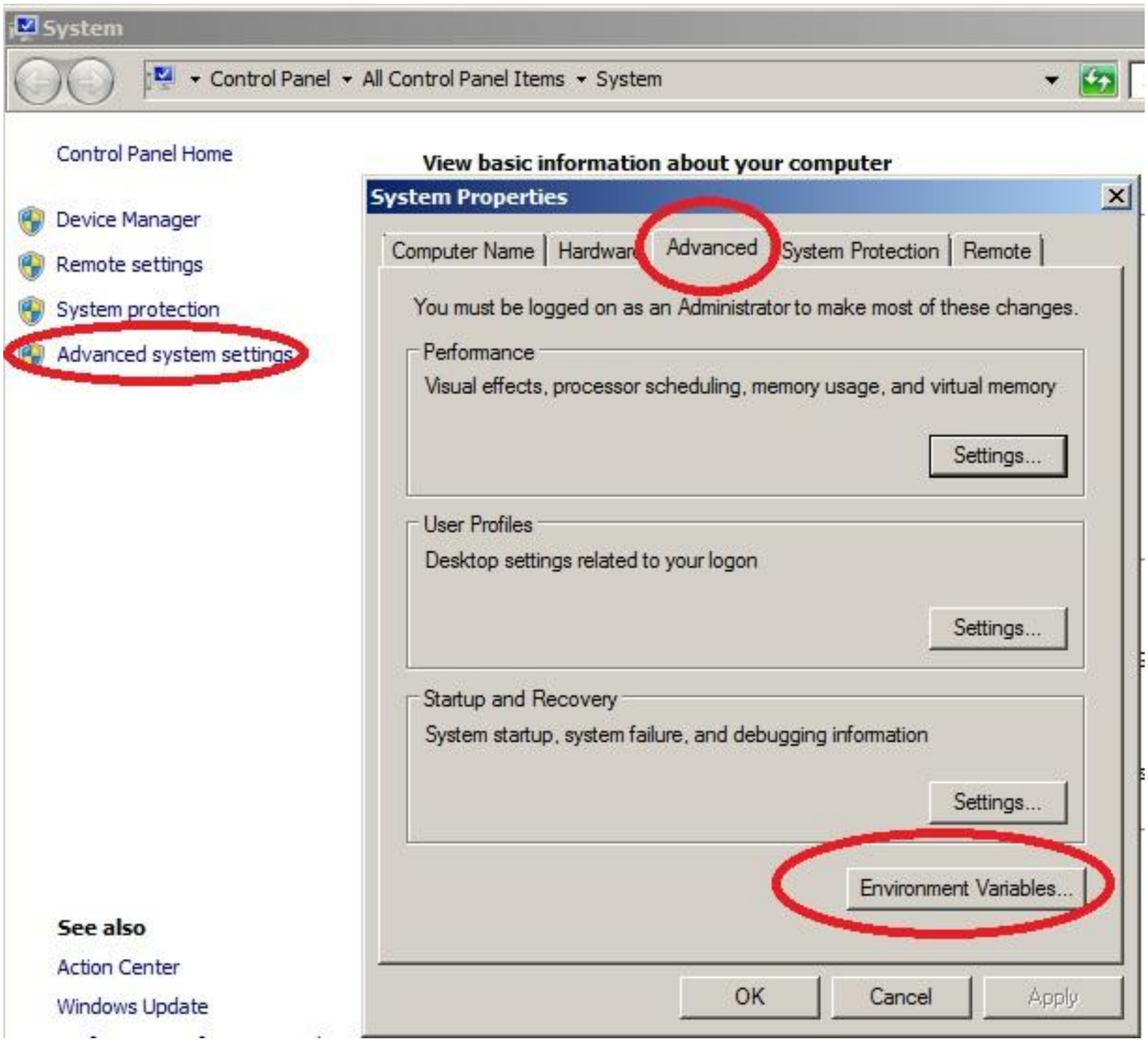
რომელთაგანაც პირველი წარმოადგენს Java Development Kit ს. ის ასევე შეიცავს Java Virtual Machine ს (JVM) რაც უშვებს ჯავაზე დაწერილ კოდს შესრულებაზე. უნდა აღინიშნოს, რომ აპლიკაციების უდიდესი ნაწილი Android პლატფორმისთვის იწერება ჯავა ზე, აქედან გამომდინარე მისი ინსტალაცია აუცილებელია. შემდეგი ძირითადი კომპონენტი არის გარემო (IDE (Integrated Development Environment)), რაც უხეშად რომ ვთქვათ წარმოადგენს პროგრამული კოდის რედაქტორს. იგი არის სპეციალური პროგრამა რომელშიც ვმუშაობთ რათა შევქმნათ პროექტები, გავუშვათ შესრულებაზე და დავინახოთ შედეგები. ერთ – ერთი ყველაზე გავრცელებული გარემო ჯავასთვის არის Eclipse , თუმცა არსებობს უამრავი სხვა როგორც უფასო ასევე ფასიანი (კომერციული გარემოები), მაგ: უფასო NetBeans, ფასიანი : JCreator, JBuilder და ა.შ. მე 3 კომპონენტი ეს არის Android SDK , რომელსაც მოგვიანებით განვიხილავთ , თუმცა მანამდე შეგვიძლია განვმარტოთ თუ რა არის ზოგადად SDK და რას წარმოადგენს Android SDK. SDK (Software Development Kit) წარმოაგდენს სხვადასხვა ხელსაწყოების (Tools ების) ერთობლიობას, რომლითაც ხდება გარკვეული პროგრამული პროდუქტის შექმნა, ასევე იგი შეიცავს სხვადასხვა ტიპის ბიბლიოთეკებსაც. Android SDK წარმოადგენს სხვადასხვა ხელსაწყოევის და ბიბლიოთეკების ერთობლიობას რომელსაც ვიყენებთ უშუალოდ Android ზე გასაშვები პროექტებისათვის. Eclipse გარემო წარმოადგენს OpenSource ტიპის (პროგრამული კოდი ხელმისაწვდომია და ღიაა) გარემოს, რომელსაც შეგვიძლია დავუყენოთ სხვადასხვა ტიპის plugin ები. Plugin ები საჭიროა კონკრეტული ტიპის პროექტებთან მუშაობის სიმარტივისთვის. ჩვენს შემთხვევაში მე 4 კომპონენტი ზუსტად ამ იდეას ემსახურება, ამ plugin ის დაყენების შემდეგ Eclipse

გარემოდან შესაძლებელი იქნება შევქმნათ პირდაპირ Android ის პროექტი და არა უბრალოდ ჩვეულებრივი ჯავა პროექტი.

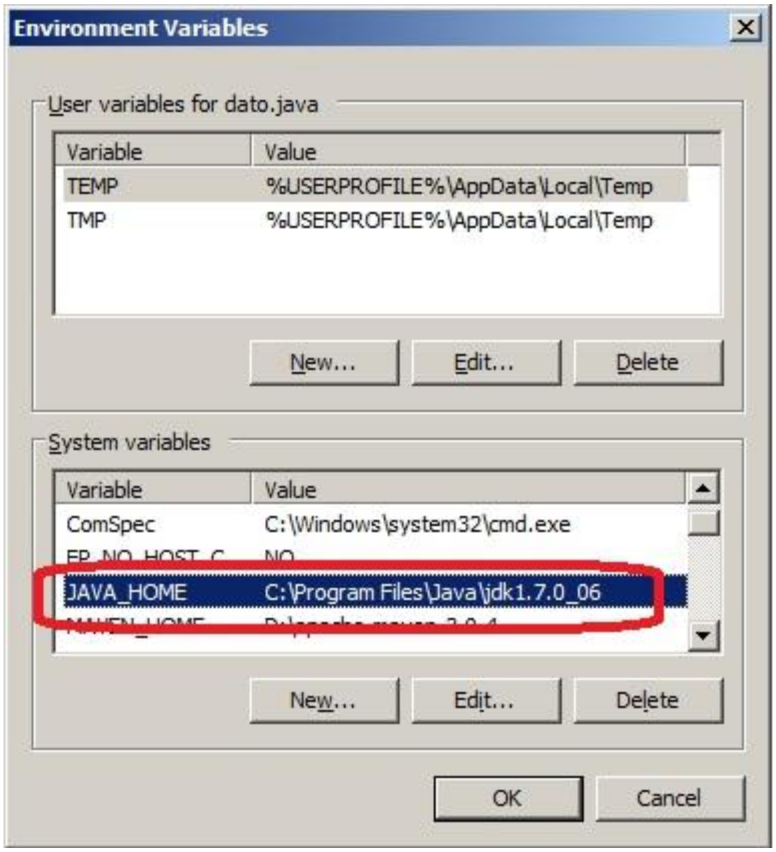
1.1 JDK ს ინსტალაცია

სანამ დავიწყებთ ინსტალაციას გადმოვწეროთ ბოლო JDK შემდეგი საიტიდან: http://www.oracle.com/technetwork/java/javase/downloads/index.html სიაში გვხდევა როგორც JDK ასევე JRE ს განოყოფილება, JDK შეიცავს JRE საც, აქედან გამომდინარე უმჯობესია გადმოვწეროთ და დავაყენოთ რომელიმე JDK. რა თქმა უნდა ყურადღება მივაქციოთ ოპერაციული სისტემის ვერსიას და არქიტექტურას რათა რაიმე გაუთვალისწინებელი არ მოხდეს.

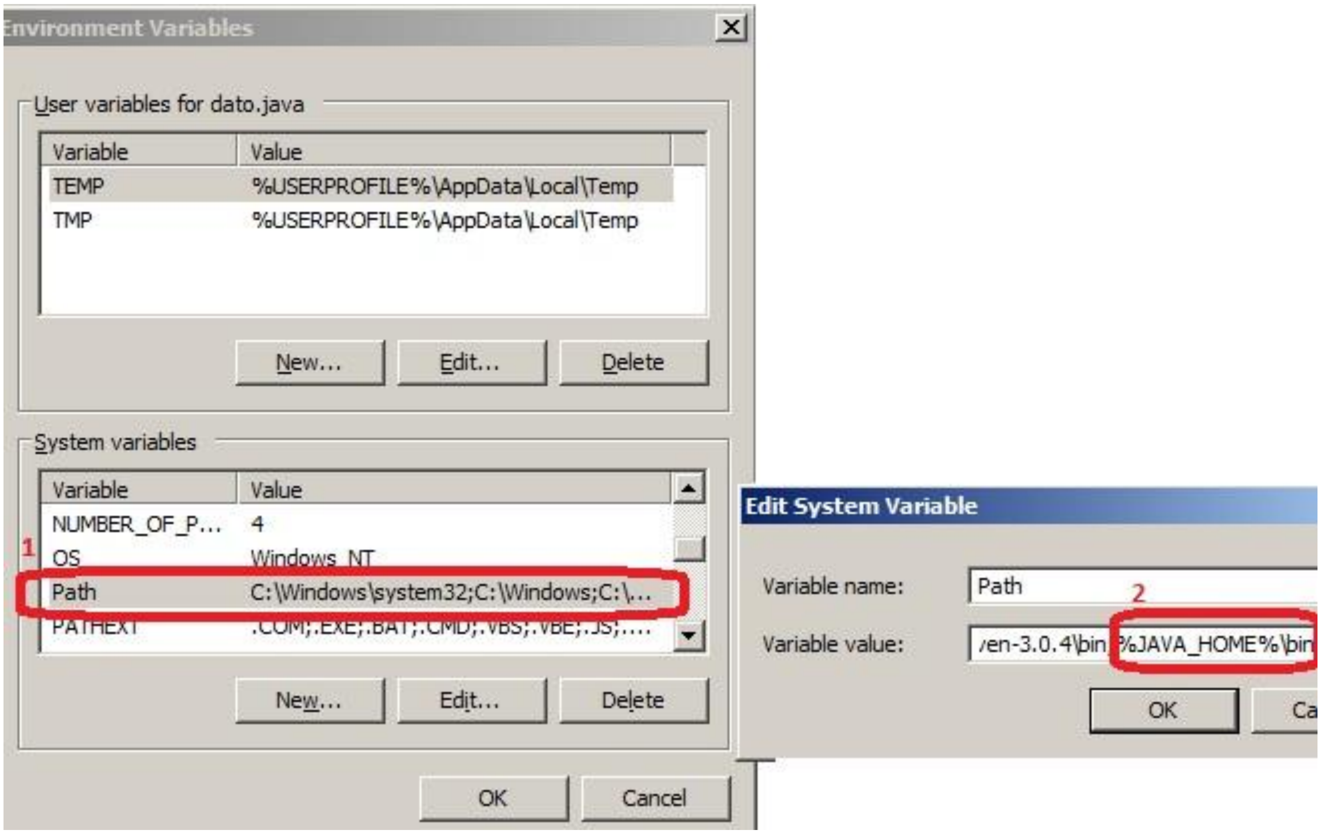
აუცილებელია ოპერაციული სისტემის ცვლადებში დავამატოთ java ze ინფორმაცია (თუ რა თქმა უნდა უკვე არ არის დამატებული ინსტალაციის შემდეგ). ამისათვის ვნიშნავთ My Computer ს, მაუსის მარჯვენა ღილაკზე დაჭერით შევდივართ properties ში, ვპოულობთ Advanced System Settings და ვაჭეთ Environment Variable ს ღილაკს:



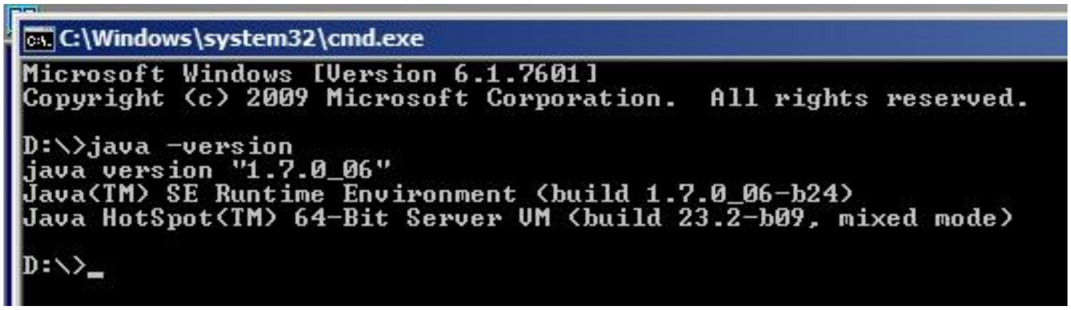
გამოსულ ფანჯარაში ვამატებთ Javas სრულ გზას ისე როგორც ეს სურათზეა ნაჩვენები:



შემდეგ იმავე ჩამონათვალში ვპოულობთ Path ცვლადს, ვახდენთ მის რედაქტირებას და უმატებთ უკვე დამატებულ JAVA_HOME ს შემდეგნაირად:

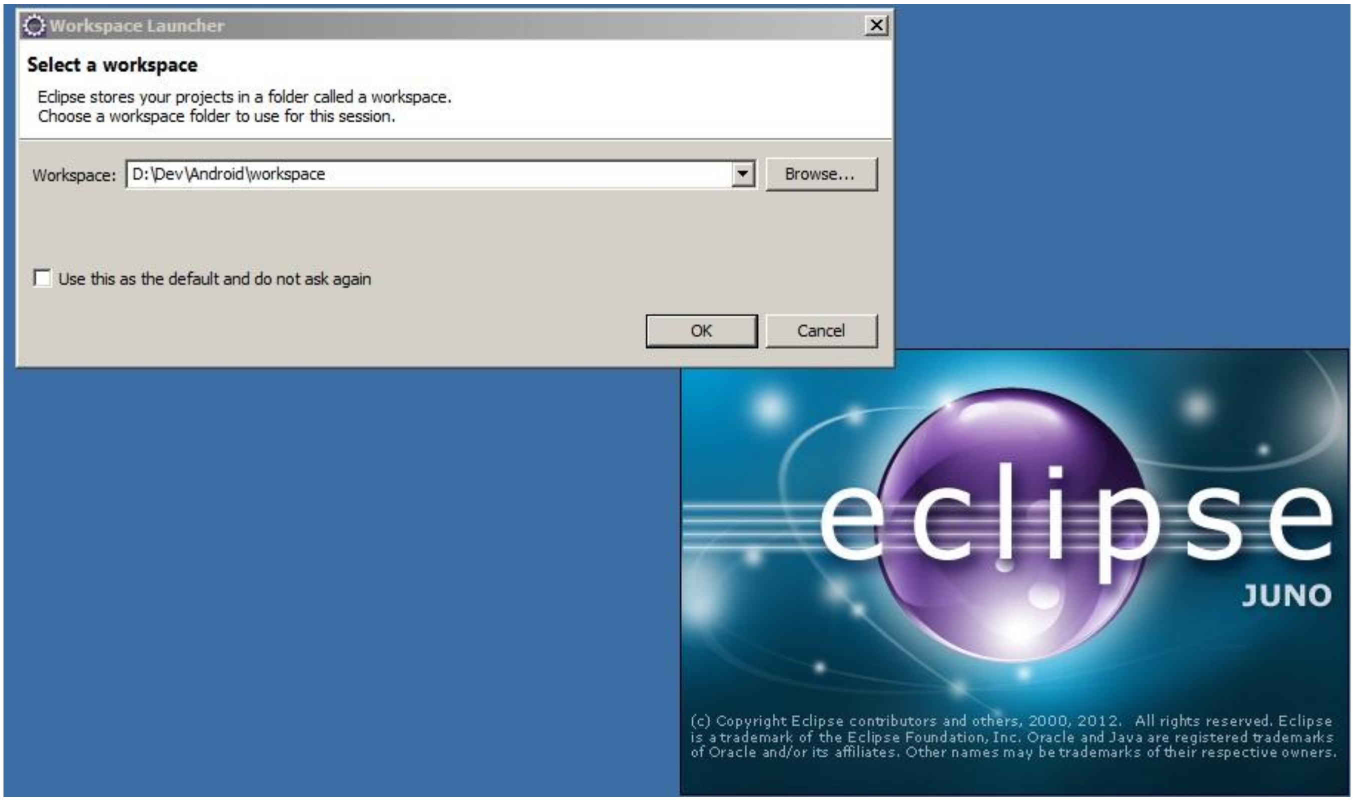


ამის შემდეგ თუკი გვაქვს რაიმე Command Line Editor გახსნილი (Far manager, Command prompt) ვხურავთ და ვხსნით Start->Accessories-Command prompt და ვუშვებთ ბრძანებას java –version რის შედეგადაც გამოვა დაახლოებით ასეთი შედეგი:



1.2 Eclipse ის ინსტალაცია

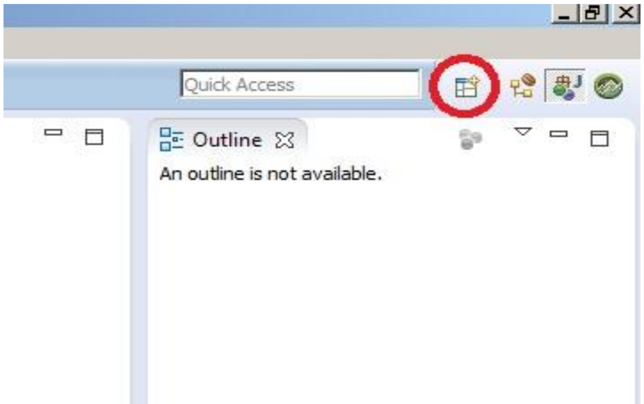
Eclipse გარემოს არ ჭირდება ინსტალაცია , თუმცა სანამ გავუშვებთ აუცილებელია გადმოვწეროთ Eclipse classic რომელიც ჩვენს შემთხვევაში სრულიად საკმარისია, შევინახოთ ის რომელიმე მყარ დისკზე (მაგრამ არა Desktop ზე , რაც არაა რეკომენდირებული) და მოვძებნოთ ფაილი eclipse.exe და გავუშვათ, ცოტა ხანში გამოვა დაახლოებით ასეთი ფანჯარა:

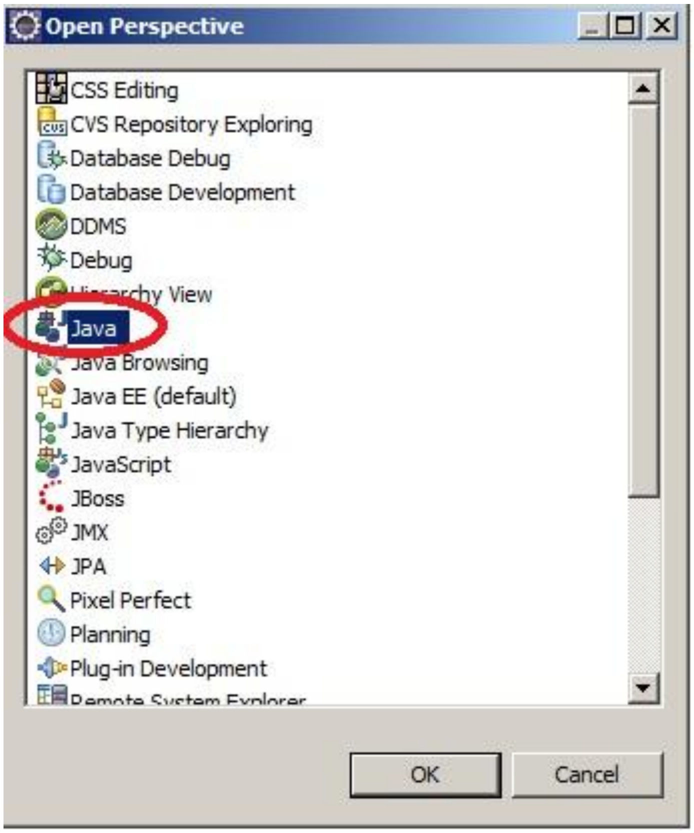


გამოსული ფანჯარა გვთავაზობს ე.წ workspace ის არჩევას, რაც იმას ნიშნავს რომ Eclipse გვეკითხება შემდეგს: "გთხოვთ აირჩიოთ დირექტორია სადაც შექმნით პროექტებს და სადაც ფიზიკურად, ფაილური სახით იქნება პროექტები განთავსებული". სასურველი დირექტორიის არჩების შემდეგ გაიხსნება გარემო რომელსაც უნდა ჩავურთოთ კონსოლი (ის წარმოადგენს Dos ის ფანჯრის ალტერნატივას სადაც გაშვებული

კოდი ბეჭდავს ინფორმაციას თუ რა თქმა უნდა ეს პროგრამულ კოდში დევს). მისი ჩართვა ხდება შემდეგნაირად: Window->Show view->Console.

გარემოს ასევე აქვს პერსპექტივის ცნება, რომელიც ჩვენს შემთხვევაში არის უბრალოდ Java , მისი ჩართვა ხდება შემდეგნაირად: ზედა მარჯვენა კუთხეში ვპოულობთ ღილაკ Open Perspective ს და მასზე დაჭერისას ვირჩევთ java ს, როგორც ეს სურათზეა ნაჩვენები:





ამის შემდეგ შეგვიძლია გადავიდეთ შემდეგ საფეხურებზე.

თუ Eclipse ის გაშვებისას მოხდა შეცდომა , დიდი ალბათობით მან ვერ იპოვა java აქედან გამომდინარე ხელით უნდა მივუთითოთ javas გამშვების სრული გზა, ეს კი ხდება შემდეგნაირად: ამოარქივებულ eclipse

ში ვპოულობთ ფაილს eclipse.ini, ვხსნით მას და ვამატებთ javas სრულ გზას, ისე როგორც ეს სურათზეა ნაჩვენები:

```
-startup
plugins/org.eclipse.equinox.launcher_1.3.0.v20120522-1813.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.200.v20120522-1813
-product
org.eclipse.epp.package.jee.product
--launcher.defaultAction
openFile
--launcher.XXMaxPermSize
256M
-showsplash
org.eclipse.platform
--launcher.XXMaxPermSize
256m
-vm
C:/Program Files/Java/jdk1.7.0_06/bin/javaw.exe
--launcher.defaultAction
openFile
-vmargs
-Dosgi.requiredJavaVersion=1.5
-Dhelp.lucene.tokenizer=standard
-Xms40m
-Xmx512m
```

მნიშვნელოვანია ერთი ფაქტორი, ეს პარამეტრი აუცილებლად უნდა ჩაემატოს იქ სადაც ნაჩვენებია , არ შეიძლება სხვა რომელიმე პარამეტრის ამოშლა ან გაუცნობიერებლად რედაქტირება.

1.3 Android SDK ს ინსტალაცია

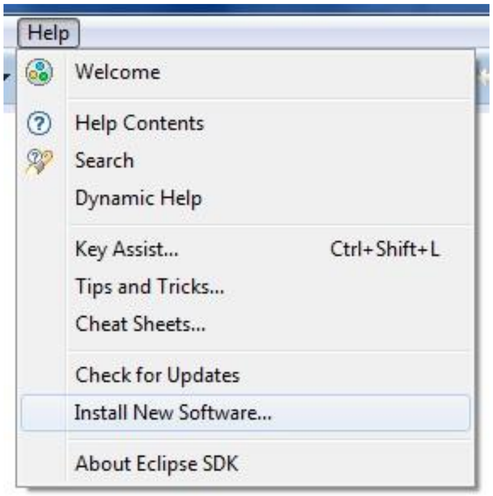
როდესაც გადმოვწერთ Android SDK ს შემდეგი მისამართიდან: http://developer.android.com/sdk/index.html , გავუშვათ exe ფაილი და დავიმახსოვროთ დირექტორია სადაც დავაყენებთ რადგან ის საჭირო იქნება შემდეგ Eclipse გარემოში მისათითებლად.

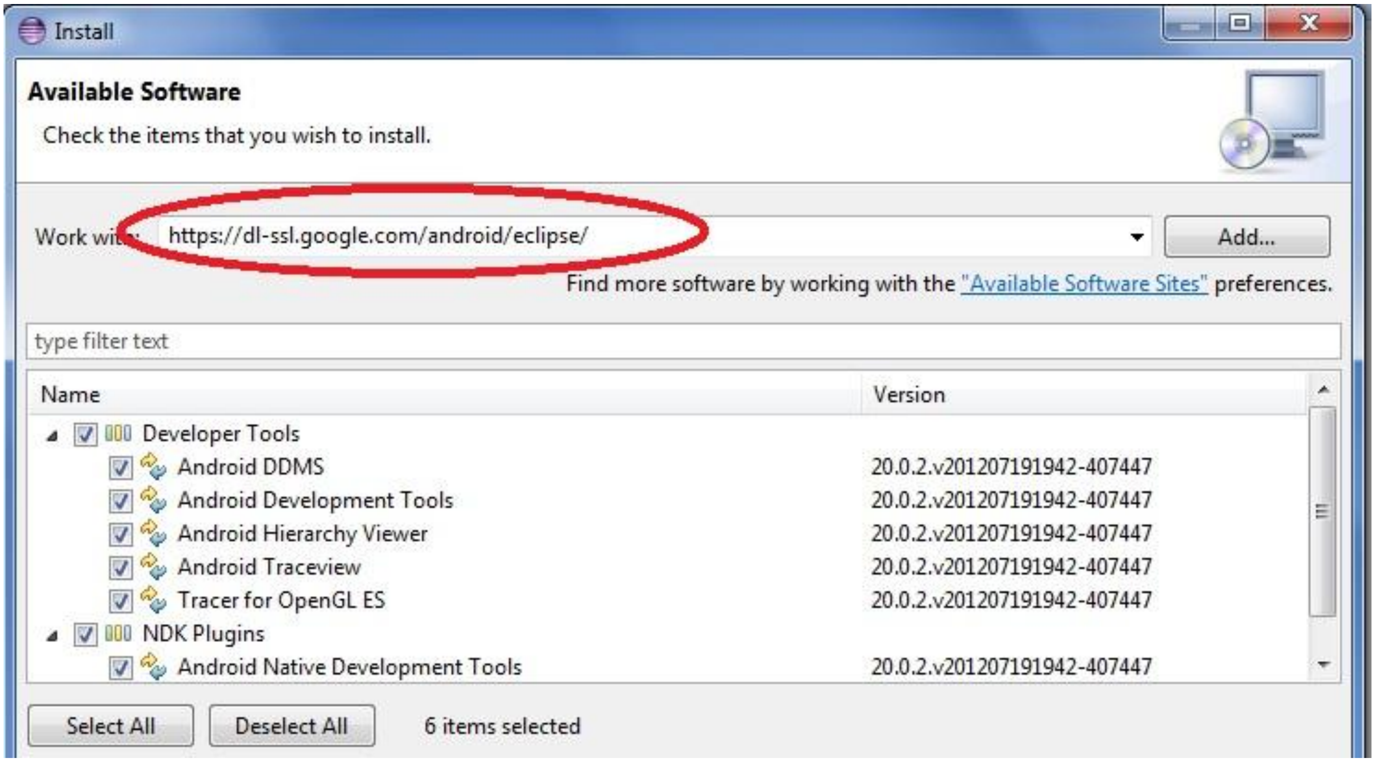
1.4 Android plugin ის ინსტალაცია Eclipse გარემოში

სანამ ინსტალაციას დავიწყებთ , აუცილებელია რომ გადმოწერილი და დაყენებული იყოს ის კომპონენტები , რომლებიც წინა თავებში განვიხილეთ

ამის შემდეგ ეტაპობრივად ვიწყებთ ინსტალაციას:

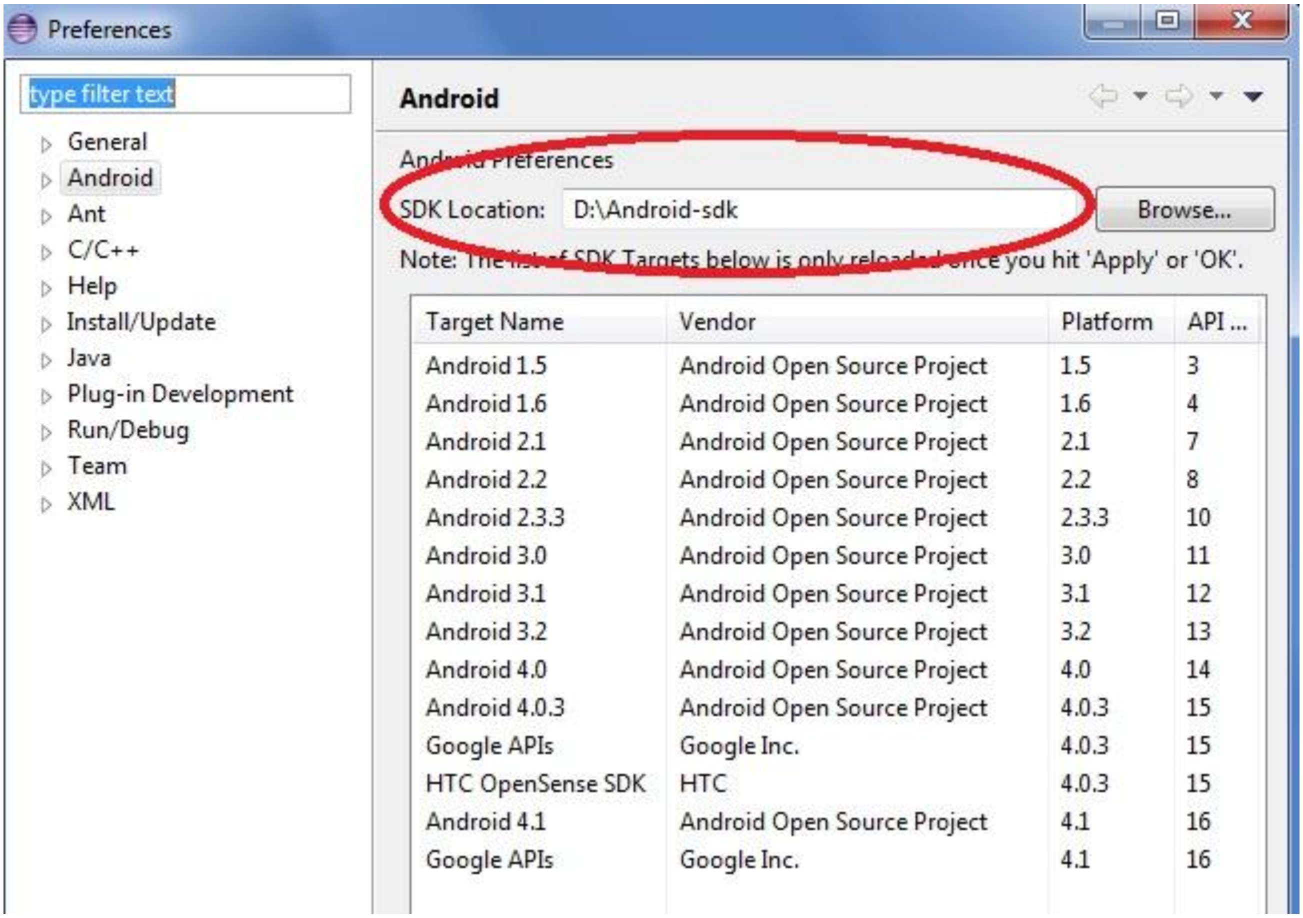
1. ვხსნით Eclipse ს და შევდივართ Help-> Install new software სადაც უნდა შევიყვანოთ plugin ის მისამართი https://dl-ssl.google.com/android/eclipse/





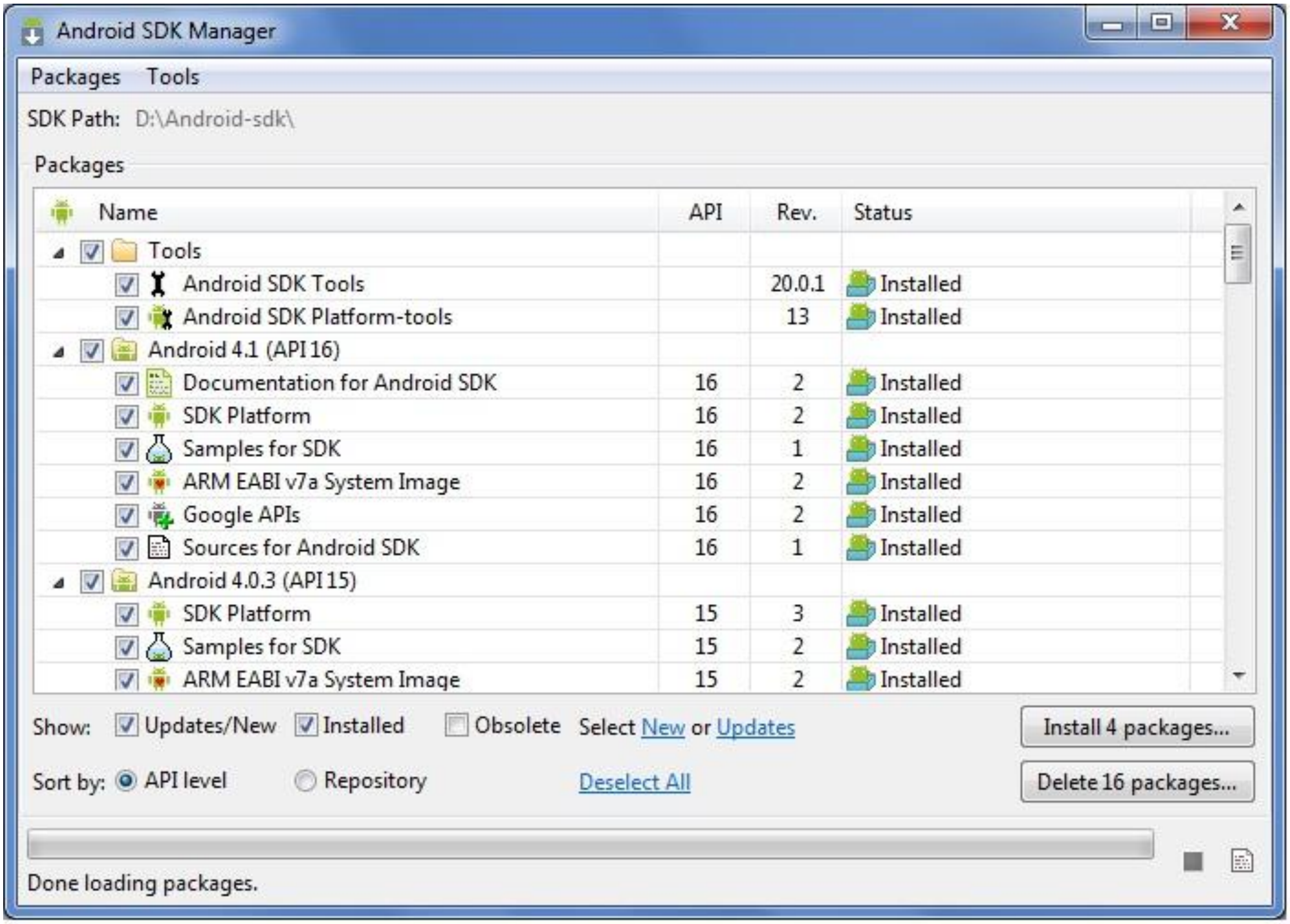
2. ვაჭერთ next ღილაკს , ვეთანხმებით ლიცენზიას და ვაჭერთ Finish ს რის შემდეგაც Eclipse დაიწყებს plugin ის ინტერნეტიდან გადმოწერას და მასში ჩასმას. უნდა აღინიშნოს რომ Eclipse მა შეიძლება შემოგვთავაზოს გადატვირთვა რაზეც აუცილებლად უნდა მივცეთ თანხმობა.
3. შევდივართ Windows->Preferences და ვაჭერთ Android ს რათა მიუთითოთ დაყენებული Android SDK ს სრული გზა (პარაგრაფის თავში აღწერილი (3)) ჩვენს შემთხვევაში ის დაყენებულია D:\Android-sdk

დირექტორიაში



1.5 Android SDK Manager ით სხვადასხვა ვერსიების SDK ს დაყენება

იმისათვის რომ მოვახდინოთ დაწერილი აპლიკაციების გაშვება ვირტუალურ გარემოში, აუცილებელია დავაყენოთ გარემოში AVD . AVD (Android Virtual Device) ს რომელიც წარმოადგენს ვირტუალურ მოწყობილობას აპლიკაციების გაშვებისათვის, საჭიროა დავაყენოთ ანდროიდის სხვადასხვა ვერსიებთან სამუშაო ბიბლიოთეკები. ამისათვის ვხსნით Window->Android SDK Manager,ვნიშნავთ ყველაფერს და ვაჭერთ install

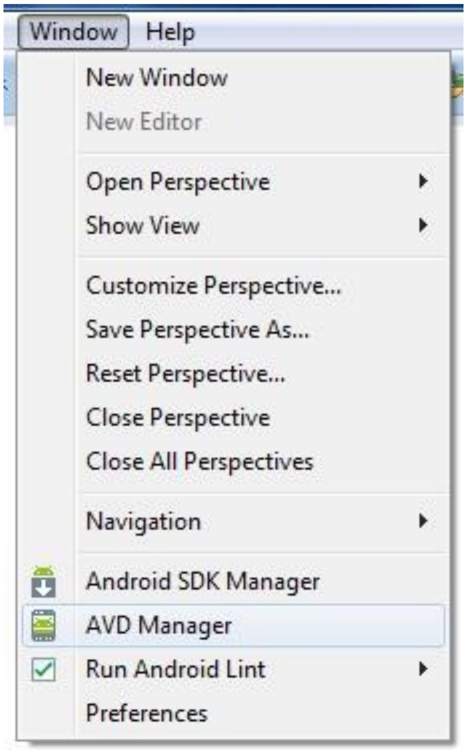


ამის შემდეგ მოხდება მონიშნული პაკეტების ინსტალაცია შესაბამის SDK ში, თითოეული ვერსია განკუთვნილია Android ის სხვადასხვა deviec ბისათვის, შესაბამისად თუ ჩვენ გვინდა რომ გვქონდეს გარემო ყველანაირი Device სათვის , უნდა მოვნიშნოთ ყველა პაკეტი და დავაყენოთ, თუკი რომელიმე პაკეტი მოითხოვს აუდენტიფიკაციას (სახელის და პაროლის შეყვანას), ის შეგვიძლია უგულებელვყოთ.

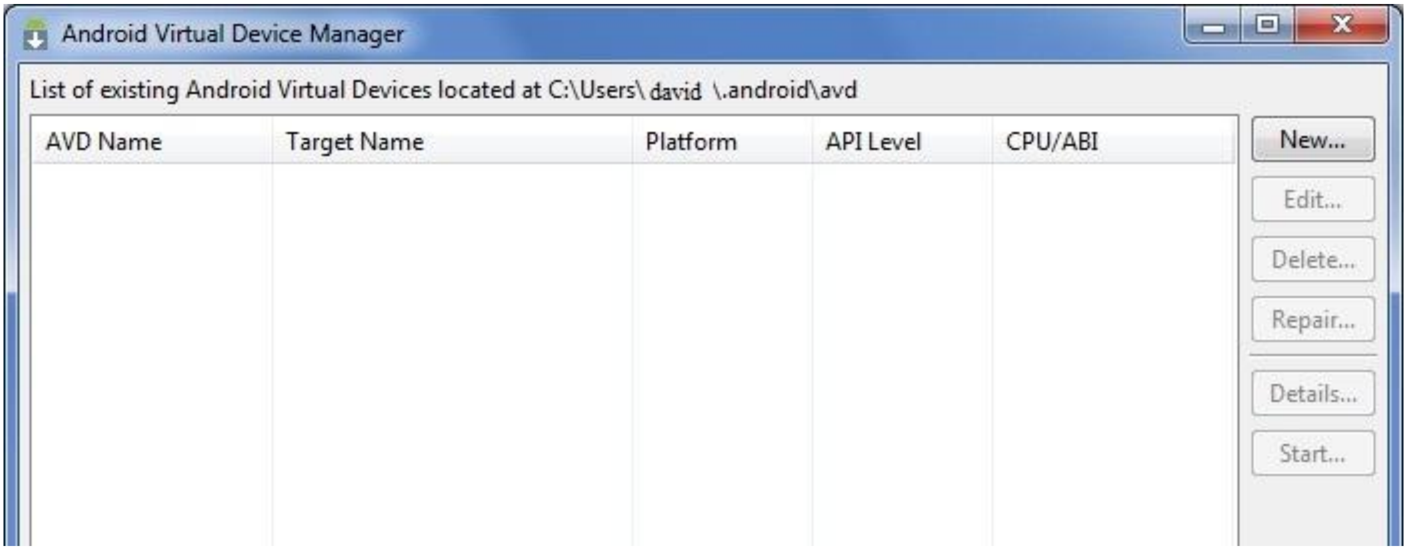
1.6 AVD (Android Virtual Device) კონფიგურაცია

იმისათვის რომ ანდროიდზე დაწერილი ნებისმიერი აპლიკაცია გაეშვას ემულატორში , აუცილებელია ერთჯერადად მოხდეს მისი კონფიგურაცია. კონფიგურაციისას ეთითება რომელი ვერსიის პლატფორმაზე ხდება ემულაცია, აქედან გამომდინარე შეგვიძლია ერთი აპლიკაცია გავატესტიროთ სხვადასხვა ვერსიებზე რათა დავრწმუნდეთ რომ კარგად მუშაობს. AVD ს კონფიგურაცია ხდება შემდეგნაირად:

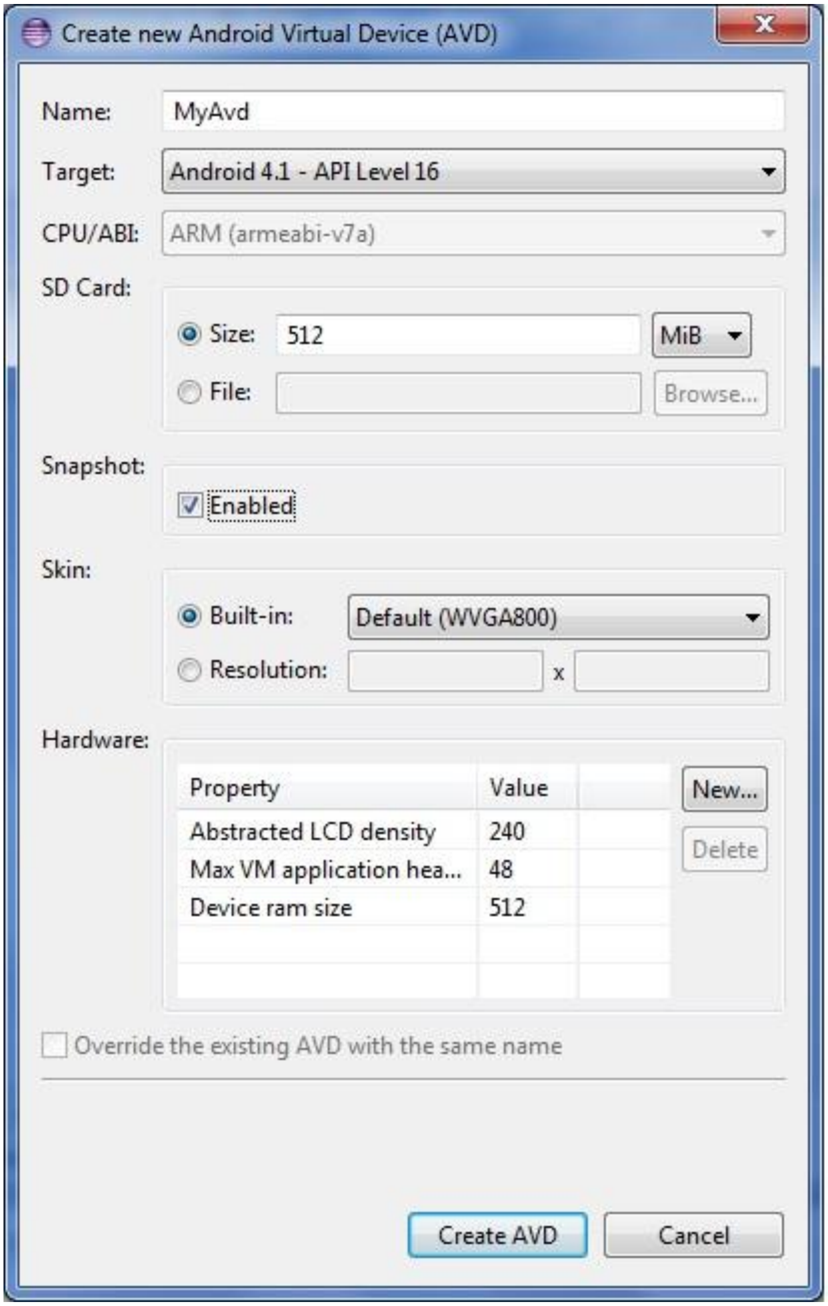
Eclipse გარემოში შევდივართ Window->AVD Manager ში:



რის შედეგადაც გამოდის ყველა იმ ვირტუალური ემულატორების სია რომელიც უკვე არის დაკონფიგურირებული (რა თქმა უნდა პირველად ეს სია ცარიელი იქნება)



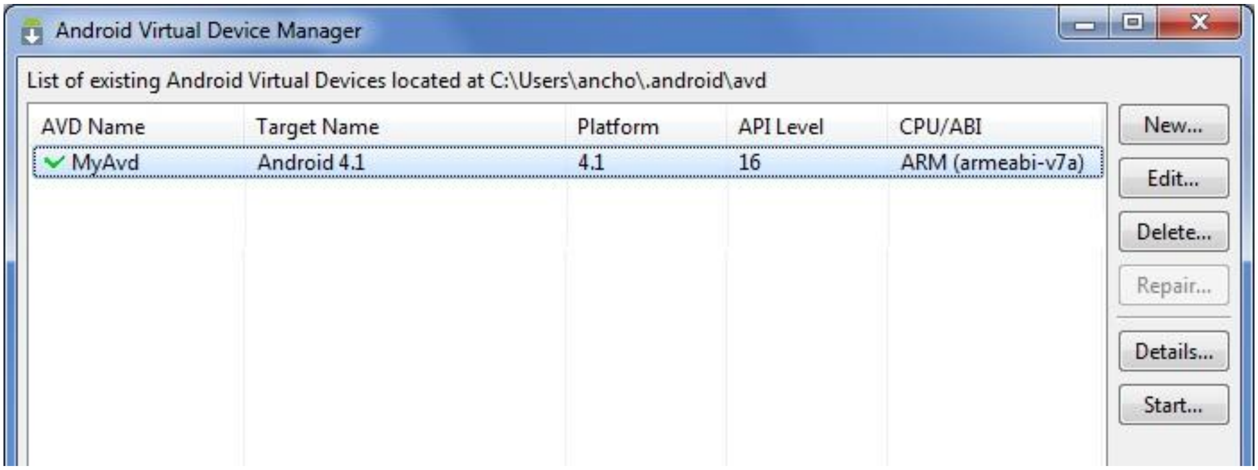
იმისათვის რომ დავამატოთ ახალი ემულატორი ვაჭერთ New ღილაკს და ვავსებთ პარამეტრებს:



ძირითადი პარამეტრები არის:

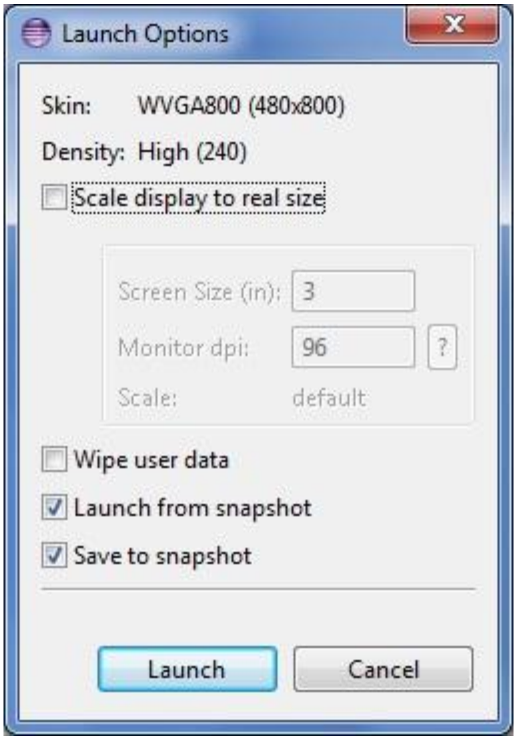
- Name ემულატორის სახელი
- Target – პლატფორმის ვერსია რომლის ემულაციაც უნდა მოხდეს
- SD Card – მეხსიერების ბარათის ზომა
- Snapshot – ეს პარამეტრი ძალიან მნიშვნელოვანია, თუ მას დავტოვებთ ჩართულს, როდესაც ემულატორს პირველად გავუშვებთ მისი ჩატვირთვის სიჩქარე ძალიან ნელი იქნება თუმცა მოხდება დამახსოვრება და ყოველ შემდეგ გაშვებაზე ჩაიტვირთება საკმაოდ ჩქარა
- Skin – საშუალება გვაქვს ავირჩიოთ როგორც ჩადგმული რეზოლუცია ასევე ხელით მიუთითოთ თუ რა რეზოლუციაზე გვინდა ემულატორის გამოტანა
- Hardware – აპარატურის მახასიათებლების ემულაციისათვის

და ვაჭერთ Create AVD ღილაკს რის შედეგადაც ემულატორის ჩამონათვალში მოხვდება ჩვენს მიერ დამატებული ემულატორი:

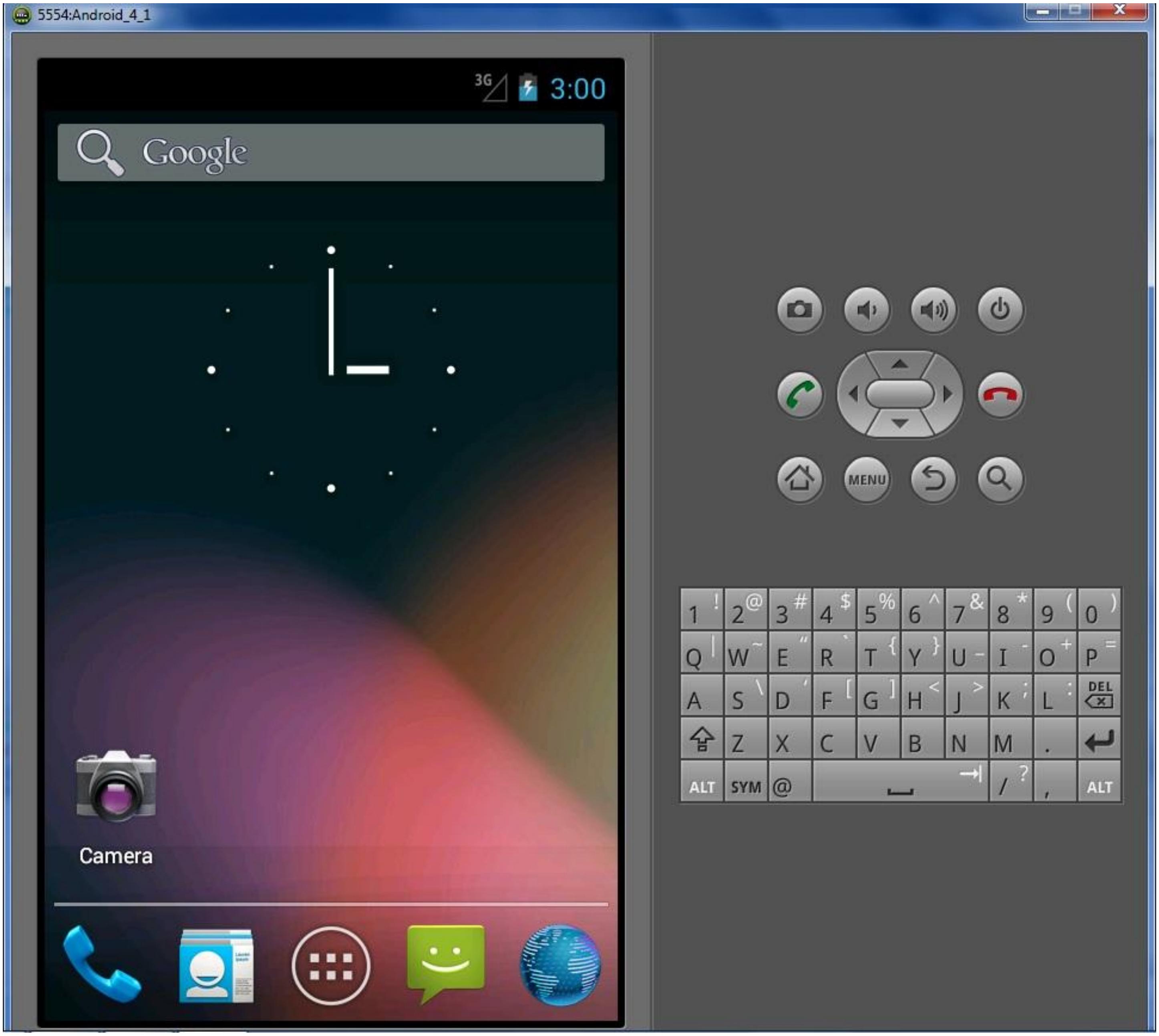


ამის შემდეგ ვაჭერთ Start ღილაკს რათა გაეშვას ემულატორი

სისტემა შეგვეკითხება ხომ არ გვინდა რაიმე სახის ცვლილება გაშვებამდე:



Launch ზე დაჭერის შემდეგ კი უკვე გაეშვება ემულატორი და ცოტა ხნის შემდეგ დავინახავთ დაახლოებით ასეთ სურათს:


5554:Android_4_1
3:00
Google
Camera
MENU
DEL
ALT
SYM
ALT

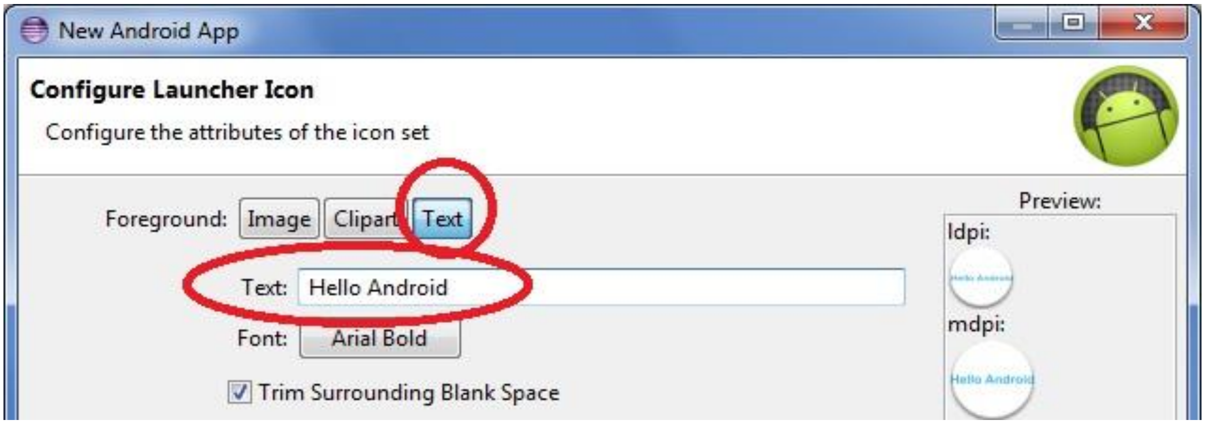
1.7 პირველი მაგალითი, პროექტის შექმნა და გაშვება

იმისათვის რომ მოვახდინოთ დემონსტრირება თუ როგორ უნდა გავუშვათ პროექტი და დავინახოთ შედეგი, შევქმნათ ის Eclipse გარემოში შემდეგნაირად:

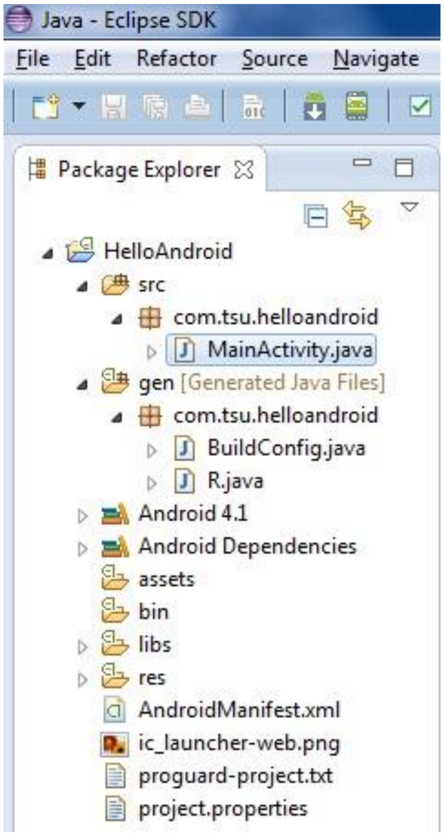
1. File->New->Other და ვირჩევთ Android Application Project ან Android Sample Project (როგორც ეს სურათზეა ნაჩვენები)

მნიშვნელოვანია შემდეგი პარამეტრები:
1. Application Name - აპლიკაციის სახელი რომლითაც დარეგისტრირდება პროექტი
2. Project Name – პროექტის სახელი
3. Package Name – java პაკეტი რომელშიც შემდეგ მოთავსდება Activity ები (Activity ებს განვიხილავთ მოგვიანებით)
4. Build SDK – ოფიციალური ანდროიდის ვერსია რომლისთვისაც იწერება პროექტი
5. Minimum Required SDK – ის ვერსია რომელსაც მინიმუმ უნდა აკმაყოფილებდეს აპლიკაცია რომ გაეშვას. ჩვენს შემთხვევაში სამაგალითო პროექტი ოფიციალურად ეშვება 4.1 ვერსიაზე მაგრამ ასევე გაეშვება ნებისმიერზე 2.2 და ზემოთ რადგან Minimum Required SDK ში მითითებულია 2.2. სხვა სიტყვებით რომ ვთქვათ ეს პარამეტრი წარმოადგენს ვერსიის ქვედა ზღვარს.

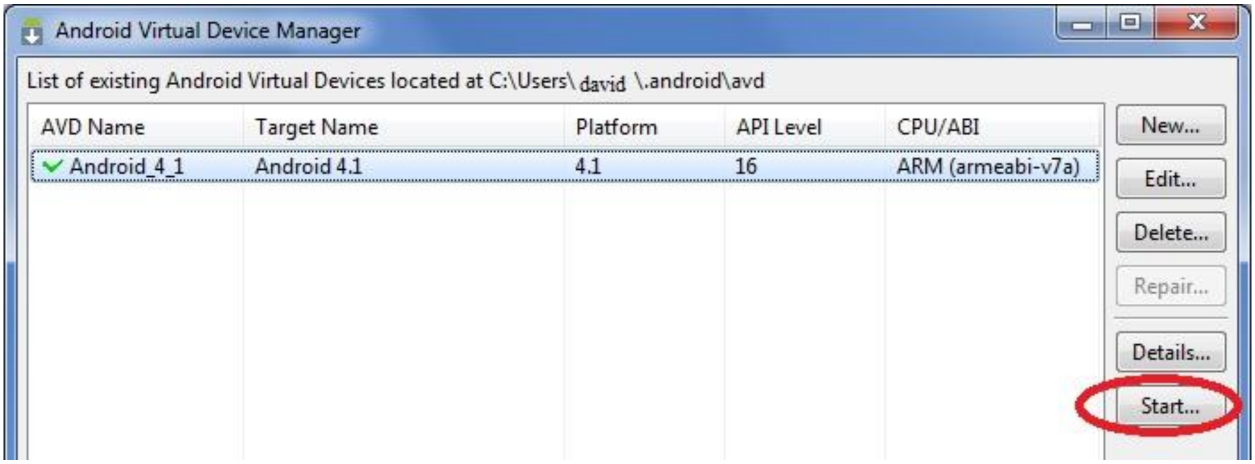
2. ვაჭერთ next s და ვირჩევთ ისეთ Foreground პარამეტრს როგორც არის ნაჩვენები სურათზე



3. კვლავ გადავდივართ next ით და ვეთანხმებით Finish ღილაკზე დაჭერით რის შედეგადაც ჩვენგან მარცხნივ გამოჩნდება პროექტი

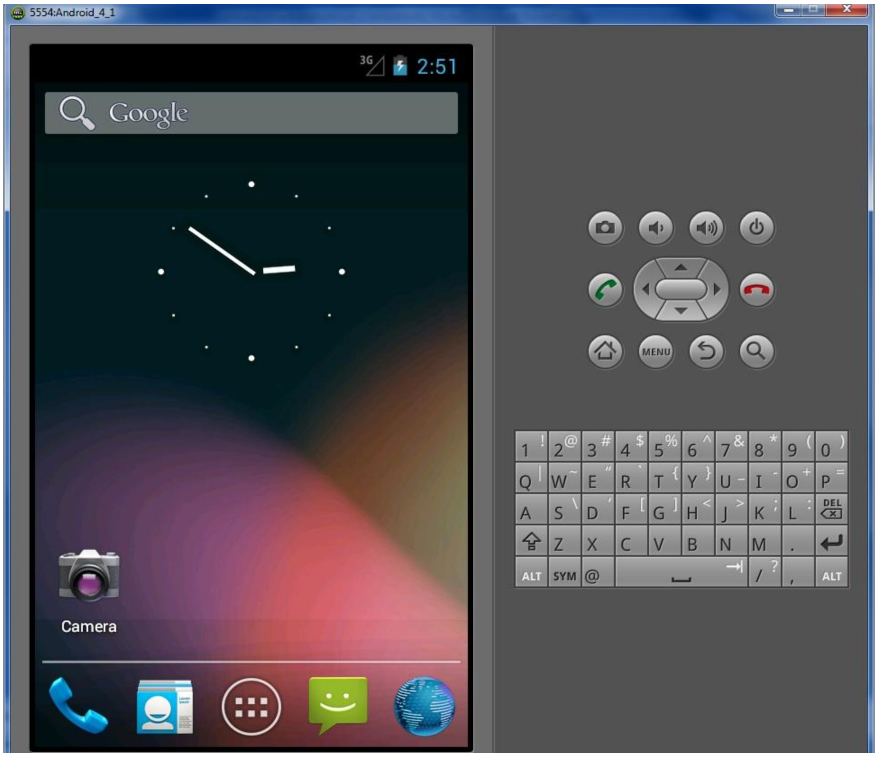


4. ამის შემდეგ უნდა გავუშვათ ვირტუალური ემულატორი რომელზეც მოხდება პროექტის გაშვების სიმულაცია, ამისათვის ვხსნით Window->AVD Manager და ვაჭერთ Start ღილაკს:



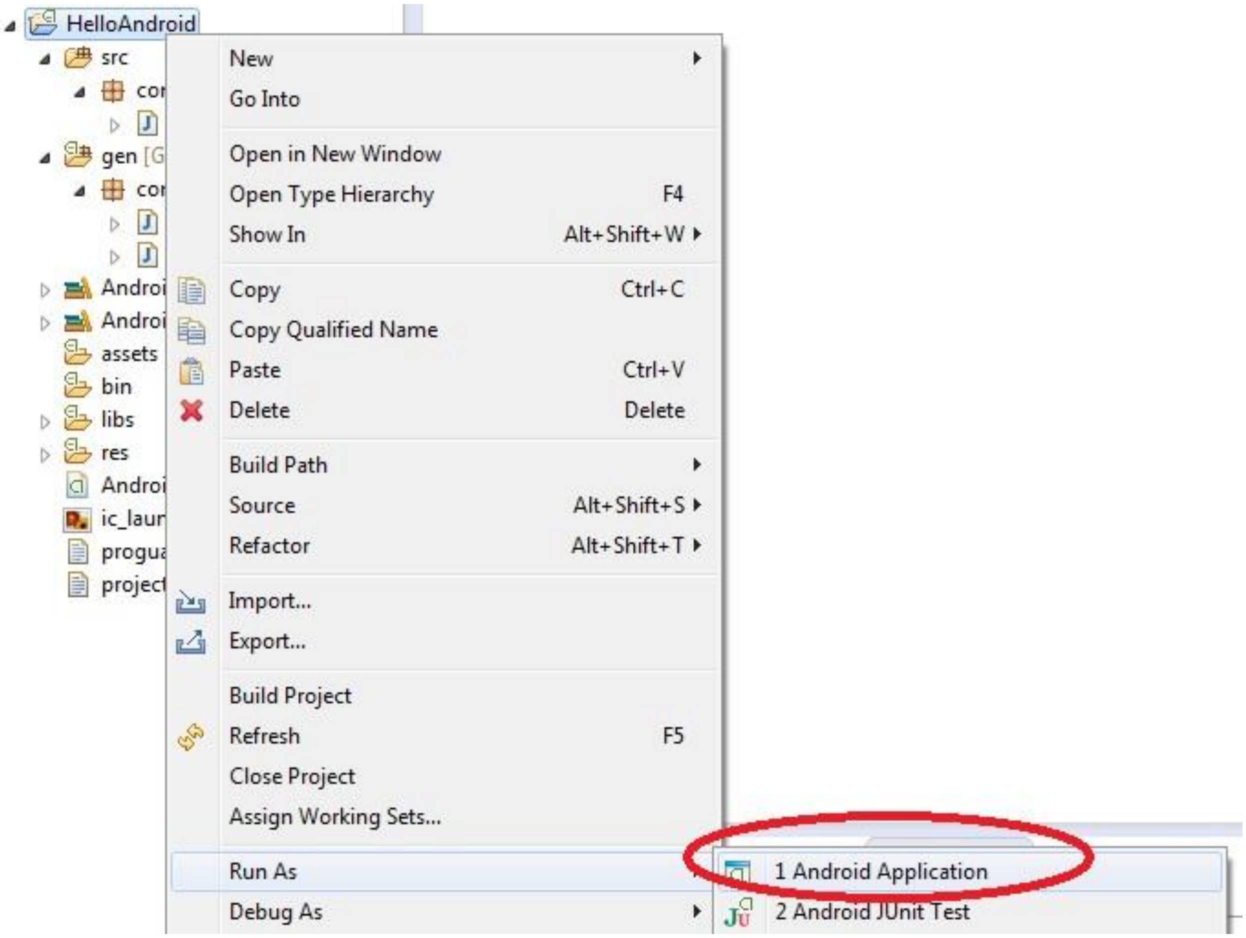
ემულატორი დაიწყებს გაშვებას და რამოდენიმე ხანში მოხდება მისი ეკრანზე გამოტანა, აქ დიდი მნიშვნელობა აქვს კომპიუტერს რომელზეც ეშვება ემულატორი, რაც უფრო კარგი და ძლიერი კონფიგურაციის კომპიუტერია მით უფრო ადვილად და კარგად ეშვება ის.საბოლოოდ

ეკრანზე გამოვა დაახლოებით ასეთი ფანჯარა

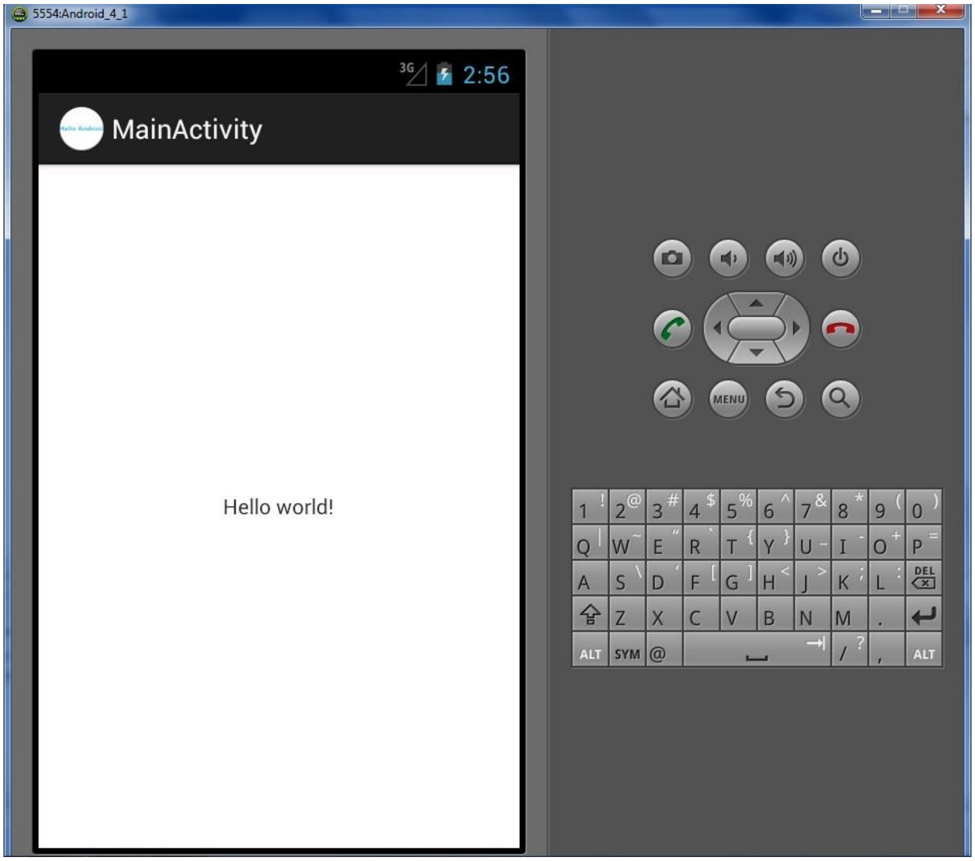


სურათზე გამოსული ემულატორი წარმოადგენს 4.1 ვერსიის შესაბამის ემულატორს, სხვა ვერსიას გააჩნია სხვანაირი ინტერფეისი, შესაბამისად რომელი ვერსიისთვისაც დავაკონფიგურირებთ AVD ს იმ ვერსიის ემულატორი გამოვა.

5. ბოლო ეტაპზე როდესაც უკვე ემულატორი არის გაშვებული და ყველაფერი მზადაა პროექტის გასაშვებად, შეგვიძლია გავუშვათ ახლადშექმნილი პროექტი შემდეგნაირად: მოვნიშნოთ და მაუსის მარჯვენა ღილაკზე დაჭერით ავირჩიოთ ის პუნქტი რაც სურათზეა ნაჩვენები:


HelloAndroid
src
gen
assets
bin
libs
res
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Build Project
Refresh F5
Close Project
Assign Working Sets...
Run As
Debug As
1 Android Application
2 Android JUnit Test

6. საბოლოოდ მივიღებთ დაახლოებით ასეთ შედეგს:



1.8 პროექტის შემადგენელი ნაწილების დეტალური განხილვა

პროექტი შედგება როგორც უშუალოდ პროგრამული ნაწილისაგან, ასევე დამატებითი ფაილებისაგან სადაც თითოეულს გააჩნია ცალკეული, დამოუკიდებელი ფუნქცია, პროექტის შემადგენელი ნაწილები შესაძლებელია დავყოთ შემდეგ ძირითად ნაწილებად:

- კონფიგურაციის ფაილები
- პროგრამული კოდი
- რესურსები
- ბიბლიოთეკები

ნებისმიერი პროექტი აუცილებლად შეიცავს თავის ძირეულ დირექტორიაში AndroidManifest.xml ფაილს , რომელშიც არის აღწერილი პროექტის ძირითადი კონფიგურაცია, მაგალითისთვის შეგვიძლია განვიხილოთ წინა თავში განხილული HelloWorld პროექტის AndroidManifest.xml ფაილი რომელიც არის აღწერილი შემდეგნაირად:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.david.helloandroid"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
```

```
<application
    android:debuggable="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
        android:name=".MainActivity"
        android:label="@string/title_activity_main" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>

</manifest>
```

ახლა განვიხილოთ მისი შიგთავსის ძირითადი ტეგები. ყველა ის ქვეტეგი რომელსაც კონფიგურაციის ფაილი მოიცავს აუცილებლად უნდა იყოს აღწერილი <manifest> ტეგის შიგნით. <manifest> ტეგი წარმოადგენს ძირითად ტეგს რომელშიც ვაკეთებთ ცვლადის აღწერას შემდეგნაირად : xmlns:android=http://schemas.android.com/apk/res/android რათა სხვა ქვეტეგებმა მოახდინონ გარკვეული პარამეტრებისათვის მნიშვნელობების მინიჭება ამ ცვლადის მეშვეობით, ჩვენს შემთხვევაში ცვლადს სახელად ქვია android , ხოლო xmlns (Xml NameSpace) წარმოადგენს სტანდარტულ საშუალებას მის აღსაწერად. package ატრიბუტი რომლის მნიშვნელობაც არის com.david.helloandroid წარმოადგენს იმ ძირითად ჯავა პაკეტს რომელზეც ცოტა მოგვიანებით ვისაუბრებთ და რომელშიც უშუალოდ უნდა მოხდეს პროგრამული კოდის შესაბამისი კლასების დამატება/დაწერა. ეს არის ზუსტად ის Package Name რაც წინა თავში განვიხილეთ როდესაც პროექტის შექმნისას გარემო გვეკითხებოდა ამ პარამეტრს. შემდეგი ორი პარამეტრიდან android:versionCode და android:versionName, პირველი წარმოადგენს ვერსიის კოდს, ხოლო მეორე დასახელებას. ამ პარამეტრებში ჩვენი სურვილისამებრ შეიძლება ჩაიწეროს რაიმე მნიშვნელობები.

შემდეგი მნიშვნელოვანი ტეგი ეს არის <uses-sdk> რომელსაც განსაზღვრული აქვს ძირითადად შემდეგი ორი პარამეტრი : minSdkVersion და targetSdkVersion. ახლა განვმარტოთ რას ნიშნავს ჩვენს შემთხვევაში აღწერილი android:minSdkVersion=”8” android:targetSdkVersion=”15”. როგორც ვიცით android ოპერაციულ სისტემას გააჩნია ვერსიები, თითოეული ვერსიას აქვს თავისი დასახელება , ასე მაგალითად Android 4.0.3 – Ice Cream Sandwich , Android 3.2 – HoneyComb და ა.შ. ხოლო თითოეულ ამ ვერსიას (3.2, 4.0.3 და ა.შ) გააჩნია პროგრამირების ვერსია რომელსაც ეწოდება API (Application Programming Interface) ვერსია და თითოეულ android ვერსიას შეესაბამება API ვერსია. ზუსტად ეს API ვერსია არის ის რაც აღვწერეთ android:minSdkVersion და android:targetSdkVersion ში. შეიძლება დაიბადოს კითხვა რა საჭიროა ორი ვერსიის აღწერა ? ან საერთოდ რას ნიშნავს minSdkVersion და targetSdkVersion ? და რისთვისაა ეს ორივე ერთად საჭირო? საქმე იმაშია რომ რეალურად სხვადასხვა ტიპის მოწყობილობებს გააჩნიათ სხვადასხვა ტიპის ვერსიები (მაგ. მობილურებს ძირითადად უყენიათ Android 2.3 Gingerbeard ხოლო tablet ებს კი Android 3.2 ან Android 4.x), შესაბამისად თუ გვინდა დავწეროთ ისეთი აპლიკაცია რომელსაც ექნება როგორც ერთი ისე სხვა ვერსიების მხარდაჭერა ამისათვის უნდა განვსაზღვრო მინიმუმ რა ვერსიასთან არის თავსებადი ჩვენი აპლიკაცია და ძირითადად რომელი ვერსიისთვის არის დაწერილი . minSdkVersion ზუსტად განსაზღვრავს იმას თუ მინიმუმ რომელი API ვერსიისათვის (და შესაბამისად რომელი Android ვერსიისათვის) ვაპირებთ

აპლიკაციის შექმნას, ხოლო targetSdkVersion კი განსაზღვრავს თუ რომელი API ვერსიისათვის (და შესაბამისად რომელი ანდროიდ ვერსიისათვის) არის უშუალოდ გამიზნული აპლიკაცია. ჩვენს განხილულ მაგალითზე შეგვიძლია ვთქვათ შემდეგი : ჩვენმა დაწერილმა აპლიკაციამ წარმატებით უნდა იმუშაოს მინიმუმ Android 2.2 Froyo (Api 8), მაგრამ ოფიციალურად გამიზნულია Android 4.0.3 (Api 15) ისათვის. არის კიდევ ერთი პარამეტრი maxSdkVersion რომელიც ანალოგიურად განსაზღვრავს რომელი ვერსიის ჩათვლით უნდა მოხდეს აპლიკაციის მხარდაჭერა.

ახლა კი გადავიდეთ შემდეგ ტეგზე რომელიც არის <application> და რომელიც შეესაბამება უშუალოდ აპლიკაციის კონკრეტულ დეტალებს, მას გააჩნია შემდეგი ძირითადი ატრიბუტები/თვისებები:

- android:debuggable="true" ეს პარამეტრი განკუთვნილია რეალურ device ზე ტრასირების საჭიროებისათვის. თუ ტრასირება არ არის საჭირო შეგვიძლია ეს პარამეტრი უბრალოდ არ დავწეროთ
- android:icon="@drawable/<სურათის სახელი>" როდესაც აპლიკაცია ეშვება ზედა მარცხენა კუთხეში გამოდის პატარა სურათი ლოგოს სახით, ხოლო ეს სურათი არის რესურსებში რომელსაც მოგვიანებით განვიხილავთ, სურათის ტიპის რესურსებზე როგორც წესი მიმართვა ხორციელდება სინტაქსით @drawable/<შემდეგ კი სურათის სახელი.გაფართოება>
- android:label="@string/app_name" აპლიკაციას გააჩნია სახელი რომელი სახელითაც ის ჩანს device ში, ხოლო ეს სახელები არის აღწერილი strings.xml ფაილშ რომელსაც მოგვიანებით განვიხილავთ ხოლო ამ პარამეტრის სახელი არის app_name ჩვენს შემთხვევაში რომლის უკანაც რაიმე სტრიქონი დგას. მიმართვას კი აქ სინტაქსი @string/<რაიმე სახელის გასაღები რომელსაც უდგას უკან რაიმე სტრიქონული მნიშვნელობა>
- android:style="@style/AppTheme" აპლიკაციას აგრეთვე გააჩნია თემა/ვიზუალური სტილი რომელშიც ის გამოდის , ეს პარამეტრი კონტროლდება styles.xml ფაილში რომელსაც მოგვიანებით განვიხილავთ , მანამდე კი შეგვიძლია ვთქვათ , რომ styles.xml ფაილში არის აღწერილი თემა სახელად AppTheme რომელსაც უკან უდგას აღწერა თუ როგორ უნდა გამოიყურებოდეს აპლიკაცია, მიმართვის სინტაქსი არის შემდეგნაირი @style/<თემის აღმწერი სახელი რომელშიც წერია რაიმე თემა>

application ტეგი გარდა ზემოთ აღწერილი პარამეტრებისა შეიცავს კიდევა უამრავ სხვა პარამეტრებს , რომლებსაც დროდადრო განვიხილავთ ახლა კი გადავიდეთ მისი ერთ–ერთი ქვეტეგის <activity> ს აღწერაზე, რომელიც წარმოადგენს ამოსავალ წერტილს android ის აპლიკაციებში. მისი ძირითადი პარამეტრები არის :

- android:name=".MainActivity" სახელი რომელსაც უკან უდგას ჯავა კლასი , ეს ჯავა კლასი არის აღწერილი იმ პაკეტში რომელიც არის manifest ტეგის ერთ–ერთი პარამეტრი და მას სახელად ქვია MainActivity წერტილის გარეშე, ხოლო აქ ვახდენთ წინ წერტილის დასმას
- android:label="@string/title_activity_main" რაიმე სტრიქონული ტიპის სახელი, შეგვიძლია გამოვიყენოთ დამატებითი ინფორმაციის მისათითებლად, ჩვენს შემთხვევაში მასში წერია strings.xml ფაილში აღწერილი title_activity_main სახელის მქონე სტრიქონის მნიშვნელობა (strings.xml და მის სტრუქტურას განვიხილავთ მოგვიანებით)

activity ტეგს აქვს ძალიან მნიშვნელოვანი <intent-filter> ტეგი ორი ქვეტეგით, ესენია : action და category. action თავის მხრივ აქვს ატრიბუტები რომელთაგან მთავარია android:name სადაც ჩვენს შემთხვევაში წერია „android.intent.action.MAIN“, ხოლო category ტეგის სახელში კი android.intent.category.LAUNCHER. ეს ორი პარამეტრი ერთად ნიშნავს შემდეგს: MainActivity არის მონიშნული როგორც მთავარი (android.intent.action.MAIN) და ამავე დროს როგორც გამშვები (android.intent.category.LAUNCHER). მთავარი ნიშნავს, რომ სისტემა პროექტის გაშვებისას ეძებს მას რათა მისით დაიწყოს მოქმედება და ამავდროულად ის იყოს გამშვები. სხვა სიტყვებით რომ ვთქვათ როგორც C ზე დაწერილ კონსოლურ აპლიკაციაში/პროგრამაში ყოველთვის გვაქვს main() ფუნქცია ასევეა აქაც, უბრალოდ აქ გვაქვს Activity კლასი რომელსაც ამ ზემოთ აღწერილი პარამეტრების შესაბამისად „ვეუბნებით“ სისტემას , რომ ისაა მთავარი და გამშვები.

პროგრამული კოდი აღწერილია პროექტის ძირეული დირექტორიის src ქვედირექტორიაში რომელიც შეიცავს პაკეტს და შიგნით java კლასებს (შეგახსენებთ რომ პაკეტი ესაა წერტილებით გამოყოფილი ქვედირექტორიების ერთობლიობა, ასე მაგალითად თუ პაკეტს ქვია com.david.tsu ეს ნიშნავს რომ ფაილურად არის საქაღალდე com რომელშიც არის საქაღალდე david და რომელშიც არის საქაღალდე tsu).ჩვენს შემთხვევაში გვაქვს MainActivity.java რომელსაც აქვს შემდეგი სახე:

```
package com.david.helloandroid;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }


    }
```

მოთავსებული პაკეტში com.david.helloandroid.

ასევე გვაქვს გენერირებული ჯავა ფაილები რომლებსაც სისტემა იყენებს რათა მოთხოვნისამებრ styles.xml ფაილში გაწერილი კომპონენტებისათვის დააგენერიროს მეტა ინფორმაცია ჯავა კლასების და მისი ველების სახით, გენერირება ხდება ყოველ შენახვაზე და გენერირებული ფაილები მოთავსებულია პროექტის ძირითადი დირექტორიის gen ქვედირექტორიაში

რესურსები არის მოთავსებული პროექტის ძირითადი დირექტორიის res ქვედირექტორიაში. იგი შეიძლება შეიცავდეს შემდეგი ტიპის ქვე რესურსებს:

- drawable ტიპის რესურსები წარმოადგენს იმ სურათების ნაკრებს რომელიც გვჭირდება აპლიკაციაში, მაგალითად ჩვენს შემთხვევაში გვაქვს icon რომელიც გამოდის აპლიკაციის გაშვებისას მის ზედა მარცხენა კუთხეში, ასევე გვაქვს სურათი რომელიც გვინდა რომ ჩანდეს device ზე სანამ შევალთ აპლიკაციაში. გამომდინარე იქიდან რომ ერთი და იგივე სურათები შეიძლება სხვადასხვა ტიპის რეზოლუციებზე იყოს საჭირო (ან სხვა ტიპის რეზოლუციებზე

სხვა ტიპის სურათები შეიძლება დაგვჭირდეს), drawable ტიპის რესურსები ნაწილდება შემდეგნაირად: drawable-hdpi, drawable-ldpi და ა.შ

- layout ტიპის რესურსები წარმოადგენს ვიზუალური შრის აღმწერ რესურსების ფაილს(ებს) რომელიც მოცემულია xml ფორმატში , დეტალურად ამ ტიპის რესურსს განვიხილავთ მოგვიანებით, როდესაც ვიზუალური კომპონენტების აღწერაზე გადავალთ , მანამდე კი საკმარისია ვიცოდეთ რომ ის განსაზღვრავს Activity ს ვიზუალურ მხარეს (მასშია აღწერილი ყველა ის ღილაკი და სხვა კომპონენტი რასაც Activity შეესაბამება და რასაც მომხმარებელი უყურებს გაშვებისას)
- values ტიპის რესურსები რომელშიც გვაქვს ფაილი strings.xml ხოლო მასში აღწერილია ყველა სტრიქონული ტიპის ცვლადები რომელზე მიმართვაც შეიძლება დაგვჭირდეს AndroidManifest.xml იდან ან თუნდაც რაიმე სხვა ადგილიდან, ასე მაგალითად ჩვენს შემთხვევაში მას აქვს სახე:

```
<resources>

    <string name="app_name">Helloandroid</string>
    <string name="hello_world">Hello world!</string>
    <string name="title_activity_main">MainActivity</string>

</resources>
```

თუ დავუბრუნდებით AndroidManifest.xml ფაილს სადაც წერია @string/<რაიმე სახელი> მივხვდებით რომ ეს არის ზუსტად ის სახელები რაც გაწერილია strings.xml ფაილის <string> ტეგში

- სხვა ტიპის რესურსები – გამომდინარე პროექტის საჭიროებიდან

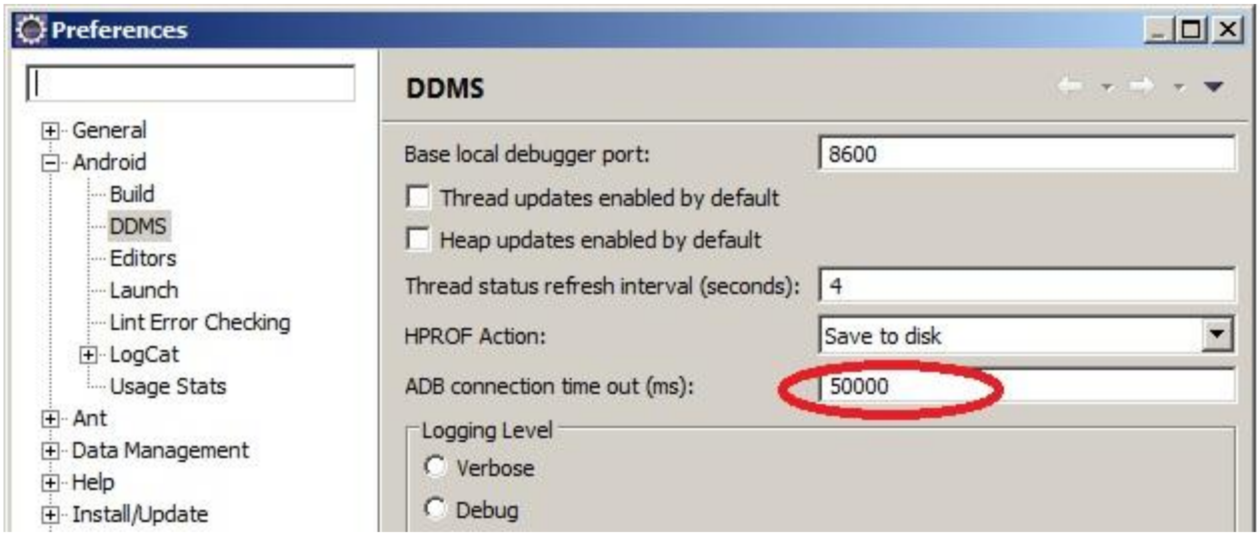
პროექტს ასევე შეიძლება ქონდეს მიბმული დამატებითი და ძირითადი ბიბლიოთეკები პროექტის საჭიროებიდან ისინი ჩანს eclipse გარემოში პროექტის ქვედირექტორი libs ში ასევე android-dependency ში.

მნიშვნელოვანია კიდევ ერთი რამ, როდესაც ხდება პროექტის შექმნა და minimum required sdk ს არჩევა თუ კარგად დავაკვირდებით ვერსიებს გვერდით ფრჩხილებში უწერიათ Api ვერსია მაგრამ ასევე წერია ორი ერთი და იგივე API ვერსია სხვადასხვა დასახელების Android ვერსიისთვის, მაგალითად Android 3.2 (API13) და Google Inc. (API13) აქ შეიძლება გაჩნდეს კითხვა: თუ ვერსიის იდენტიფიცირება ხდება API ს მიხედვით მაშინ რა განსხვავებაა Android 3.2 სა და Google Inc. ს შორის? საქმე იმაშია რომ როდესაც ვირჩევთ Google Inc. მას ასევე მოყვება ბიბლიოთეკა Google Map ებთან სამუშაოდ (Google ის რუქები) ხოლო სხვა ვერსიის არჩევის შემთხვევაში ეს ბიბლიოთეკა შესაძლოა არ გვქონდეს.

1.9 კონფიგურაცია ნამდვილი Device ისთვის

წინა თავში განვიხილეთ, თუ როგორ უნდა გავუშვათ ემულატორი რათა შემდეგ მასზე მოხდეს დაწერილი აპლიკაციის ემულაცია, მაგრამ ეს ყველაფერი აზრს კარგავს თუ არ ვიცით როგორ უნდა გავუშვათ ნამდვილ Device ზე. სხვა სიტყვებით რომ ვთქვათ, გვაქვს გარემო სადაც ვწერთ კოდს და ამასთანავე მიერთებული გვაქვს ის Device რომელზეც უნდა ვნახოთ ჩვენი დაწერილი აპლიკაცია (Device ში მოისაზრება ნებისმიერი ფიზიკური მოწყობილობა რომელზეც ოპერაციულ სისტემად აყენია Android ის რომელიმე ვერსია).

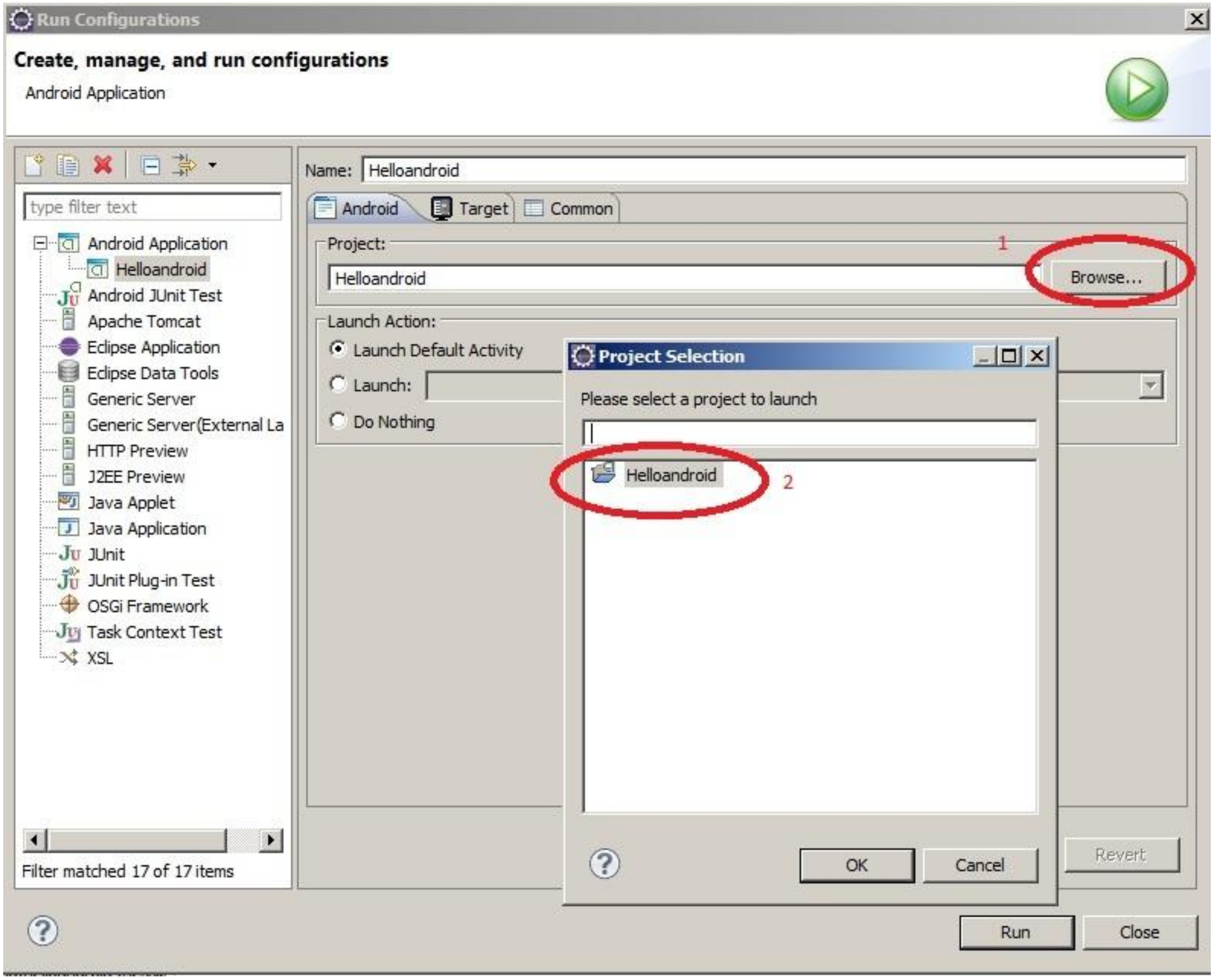
პირველ ეტაპზე უნდა გავზარდოთ Timeout, ეს არის დრო მილიწამებში რომელიც შესაძლოა მოანდომოს სისტემამ ჩვენი დაწერილი აპლიკაციის Device ზე გადასატანად/გასაშვებად ან თუნდაც ემულატორზე. პირობითად ის არის დაყენებული 5000 მილიწამზე (5 წამი), რაც არც ისე ბევრია და საჭიროებს გაზრდას. იმისათვის რომ მოვახდინოთ eclipse გარემოში (სადაც კოდს ვწერთ და საიდანაც უნდა მოხდეს გადატანა/გაშვება აპლიკაციის) ვხსნით Window->Preferences და ვაყენებთ timeout ს 50000 მილიწამზე (50 წამი), როგორც ეს სურათზეა ნაჩვენები:



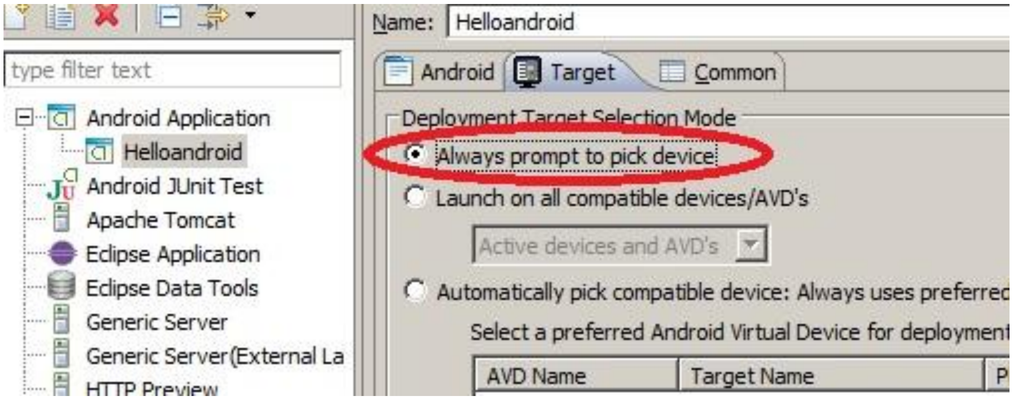
შემდეგი აუცილებელი ნაბიჯია დავაყენოთ USB Driver იმ Device ისთვის რომელზეც გვინდა აპლიკაციის გაშვება, რათა გარემომ დაინახოს ეს Device როგორც ისეთი მოწყობილობა რომელზეც შესაძლებელია ანდროიდ აპლიკაციის გაშვება. თითოეული Device სთვის ასეთი driver უნიკალურია, ამ სახელმძღვანელოს წერისას მოვახდინე Samsung galaxy tab GT-P6210 სათვის შესაბამისი driver ის გადმოწერა და დაყენება ოფიციალური საიტიდან: http://www.samsung.com/us/support/owners/product/GT-P6210MAYXAR . მისი წარმატებით დაყენების შემდეგ აუცილებელია

1. მოწყობილობის გადატვირთვა,
2. კომპიუტერიდან შესაბამისი USB ს გამოძრობა
3. მოწყობილობის ჩართვა
4. თავიდან მიერთება

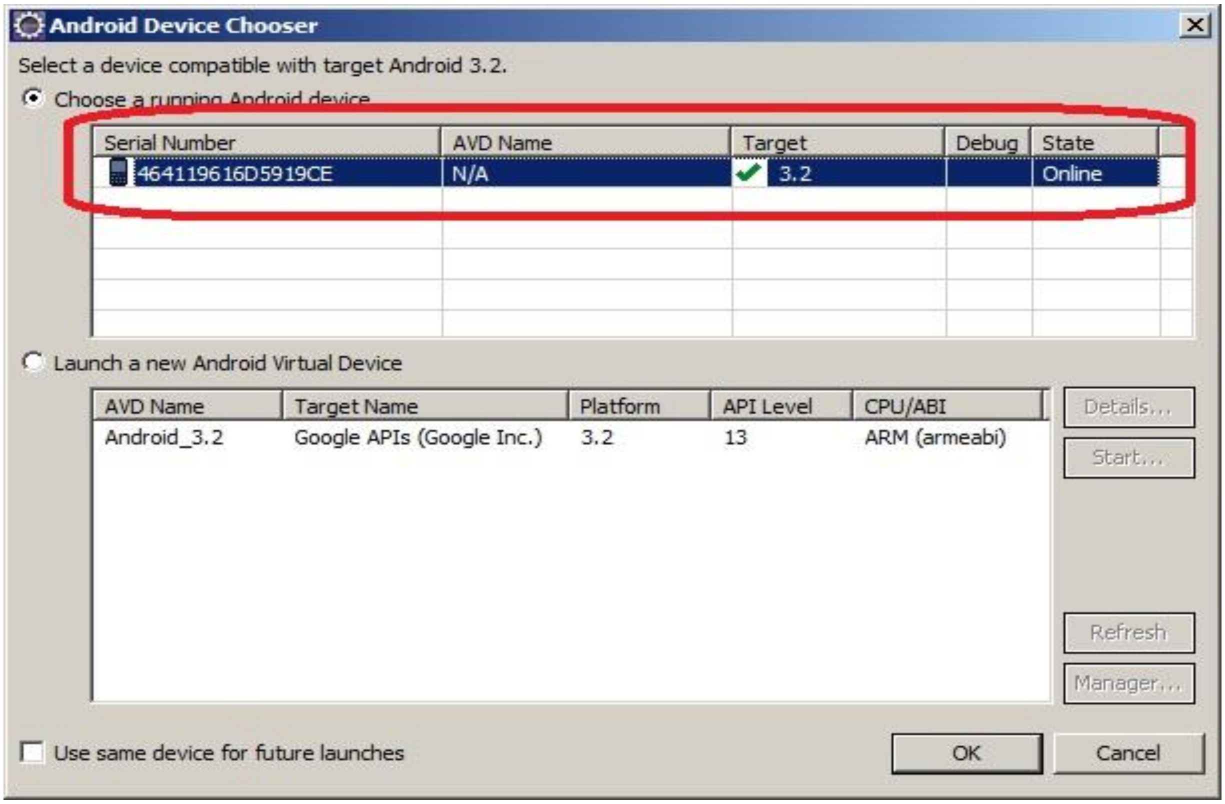
შემდეგ ეტაპზე უნდა მოვახდინოთ Eclipse ის კონფიგურაცია ისეთნაირად რომ გვკითხოს ემულატორზე გვინდა გაშვება თუ ნამდვილ Device ზე (რა თქმა უნდა გარემოს თავისით ვერ ეცოდინება რომელზე გვინდა გავუშვათ ჩვენი აპლიკაცია, თუ ეს პარამეტრი არ ჩავრთეთ შესაძლოა სისტემამ არ შემოგვთავაზოს არჩევა და ყოველთვის ემულატორზე მოახდინოს აპლიკაციის გაშვება). ამისათვის უკვე გამზადებული პროექტისთვის შევდივართ Run->Run Configuration ში browse ღილაკით ვირჩევთ პროექტს როგორც ეს სურათზეა ნაჩვენები:



შემდეგ გადავდივართ Target ტაბზე და ვნიშნავთ „Always prompt to pick device“, როგორც სურათზეა ნაჩვენები:



ამის შემდეგ შეგვიძლია ვცადოთ პროექტის გაშვება შემდეგნაირად: მოვნიშნოთ ის და მაუსის მარჯვენა ღილაკზე დაჭერისას ავირჩიოთ Run as->Android application რის შემდეგადაც გამოვა შემდეგი ფანჯარა:



როგორც სურათიდან ჩანს , სისტემამ დაინახა მიერთებული მოწყობილობა რომელიც ხასიათდება შემდეგი პარამეტრებით:

- Serial number სერიული ნომერი რომელიც ყველა მოწყობილობას აქვს უნიკალური
- AVD Name ვირტუალური მოწყობილობის სახელი
- Target ოპერაციული სისტემის ვერსია (ჩვენს შემთხვევაში ეს არის Android HoneyComb იგივე 3.2 ვერსია)
- Debug პარამეტრი ტრასირებისთვის
- State მდგომარეობა/სტატუსი

როგორც ვხედავთ სისტემამ შემოგვთავაზა არჩევა ემულატორსა და ნამდვილ მოწყობილობას შორის.

არსებობს საკმაოდ ბევრი სიტუაცია რის გამოც შეიძლება სისტემამ ვერ დაინახოს მიერთებული მოწყობილობა (ვერ დაინახოს ცნებაში იგულისხმება რომ თვითონ Eclipse გარემო ვერ ხედავს როგორც მოწყობილობას რომელზეც შეიძლება გაეშვას აპლიკაცია და არა უბრალოდ ისე როგორც ეს ხდება კომპიუტერთან Flesh მოწყობილობის მიერთებისას), ამ პრობლემის გამომწვევი ძირითადი მიზეზებია:

1. არასწორი დრაივერის დაყენება ან საერთოდ დრაივერის არ არსებულობა კომპიუტერზე
2. დრაივერი დაყენებულია თუმცა მოწყობილობა არ არის გადატვირთული
3. თვითონ მოწყობილობიდან არ არის ჩართული USB Debugging ი, რომელიც კეთდება მოწყობილობაში შემდეგნაირად : შევდივართ Settings->Application->Development და ვრთავათ USB Debugging ს.

2.0 Activities

ნებისმიერი Android ის აპლიკაცია შეიცავს ერთ ან რამოდენიმე Activity ს. Activity ეს არის აპლიკაციის კომპონენტი რომელიც გვთავაზობს „Screen“ ს რათა მომხმარებელმა შეძლოს რაიმეს გაკეთება. სხვა სიტყვებით რომ ვთქვათ როდესაც აპლიკაცია ეშვება და მომხმარებელი ხედავს „რაიმეს“ ეკრანზე, ამ „რაიმეს“ უკან უდგას ერთი ან რამოდენიმე Activity რომელთაგანაც ერთი ყოველთვის არის მთავარი. ეს სისტემა შეგვიძლია შევადაროთ მაგალითად C/C++ ზე დაწერილ რაიმე პროგრამას, როცა ხდება პროგრამის გაშვება , კონსოლური აპლიკაციის შემთხვევაში ხდება main() ფუნქციის გამოძახება, ხოლო main() ფუნქციაში პროგრამისტს შეიძლება ეწეროს მიმართვა რამე სხვა ფუნქციაზე, სტრუქტურაზე, კლასზე და ა.შ. ჩვენს შემთხვევაში main() ფუნქციის ნაცვლად გვაქვს Activity კლასების სიმრავლე ხოლო იმის მითითება თუ აქედან რომელია Main , ხდება კონფიგურაციის ფაილში რომელსაც მოგვიანებით განვიხილავთ.

იმისათვის რათა მოვახდინოთ Activity ს განსაზღვრა რეალიზაცია უნდა გავუკეთოთ Activity კლასს:

```
public class MainActivity extends Activity {

    //...............................................

}
```

2.1 Activity Life Cycle

ყოველ Activity ს გააჩნია თავისი სიცოცხლის ხანგრძლივობა, დაწყებული აპლიკაციის გაშვებიდან დამთავრებული მისი გადანდგურებით. არსებობს ძირითადი ფაზები რომლებსაც გადის თითოეული Activity, ამ ფაზებს ასევე შეგვიძლია ვუწოდოთ Event ებიც. ძირითად ფაზებს წარმოადგენენ:

1. onCreate() – იძახება როდესაც Activity პირველად იქმნება, ნებისმიერი Activity ვალდებულია გადაფაროს ეს მეთოდი, ამ მეთოდში ხდება სტატიკური ინიციალიზაცია სხვადასხვა კომპონენტების, მაგალითად: სიების, View ების , ამ მეთოდს ასევე გადაეცემა Bundle ობიექტი რომელიც შეიცავს Activity ს წინა მდგომარეობას (ეს ობიექტი განხილული იქნება მოგვიანებით).
2. onStart() – იძახება სანამ Activity უშუალოდ დანახვადი არ გახდება მომხმარებლის მხარეს , ამ მეთოდის გამოძახებას სისტემა უშუალოდ თვითონ მოაყოლებს ან onResume() მეთოდის გამოძახებას თუ Activity უნდა გამოვიდეს წინა ფონად, ან onStop() თუ ის უნდა გახდეს დამალული (hidden).
3. onResume() – იძახება უშუალოდ მანამდე სანამ Activity დაიწყებს რაიმე ქმედების განხორციელებას მომხმარებელთან მიმართებაში. ამ დროს მიმდინარე Activity არის ყველაზე პირველი Activity სტეკში, ამ მეთოდს ყოველთვის მოყვება onPause() ის გამოძახება სისტემის მიერ.
4. onPause() - იძახება როდესაც სისტემამ აუცილებლად უნდა დააბრუნოს/გაააქტიუროს სხვა Activity. ეს მეთოდი გამოიყენება რათა მოხდეს შეუნახავი მონაცემების დამახსოვრება, გააჩეროს ისეთი პროცესები რასაც მიმდინარე აპლიკაცია იყენებს და რაც ტვირთავს CPU ს. ამ მეთოდს მოყვება onResume() მეთოდი თუკი Activity უნდა დაბრუნდეს ან onStop() თუ იგი უნდა გაუჩინარდეს მომხმარებლისთვის.

5. onStop() – იძახება როდესაც მიმდინარე Activity აღარ არის ხილვადი მომხმარებლის მხარეს. ეს შეიძლება მოხდეს მისი განადგურების გამო ან სხვა Activity ზე მიმართვის გამო. ამ მეთოდს ყოველთვის მოსდევს onRestart() თუ ეს Activity ბრუნდება უკან რათა მომხმარებელს სურს რაიმე ქმედების განხორციელება ან onDestroy() თუ იგი უნდა განადგურდეს.
6. onDestroy() – იძახება სანამ მოხდება Activity ს გადანდგურება, ეს მეთოდი სისტემამ შესაძლოა გამოიძახოს რათა მოახდინოს მეხსიერების შენარჩუნება/გასუფთავება ან უბრალოდ იმიტომ რომ Activity ასრულებს სრულად თავის მუშაობას და მისი აქტიურობა აღარ არის საჭირო.
7. onRestart() – იძახება როდესაც Activity გაჩერდა იმისათვის რათა მოხდეს მისი თავიდან გაშვება.

ყოველივე ზემოთ აღწერილი შეგვიძლია წარმოვადგინოთ ცხრილის სახით:

მეთოდი	ექვემდებარება თუ არა გადადგურებას	შემდეგი შესაძლო მეთოდები
onCreate()	არა	onStart()
onRestart()	არა	onStart()
onStart()	არა	onResume() ან onStop()
onResume()	არა	onPause()
onPause()	დიახ	onResume() ან onStop()
onStop()	დიახ	onResume() ან onStop()
onDestroy()	დიახ	————

Table 1 – lifecycle მეთოდების მოკლე აღწერა

ამ ცხრილში გვაქვს ერთი ძალიან მნიშვნელოვანი ველი „ექვემდებარება თუ არა განადგურებას“. რაც იძლევა პასუხს კითხვაზე :შეუძლია სისტემას გაანადგუროს აპლიკაცია მიმდინარე მეთოდის დაბრუნების შემდეგ? მოცემული მეთოდებიდან ასეთი არის მხოლოდ 3 : onPause(), onStop(), onDestroy(). onPause( ) არის პირველი ამ სამიდან რადგან როდესაც Activity იქმნება გარანტირებულად ვიცით რომ მოხდება მინიმუმ onPause() ის გამოძახება სანამ მოხდება შესაბამისი პროცესის მოკვლა სისტემის პროცესებიდან მის მიერ.

იმისათვის რათა მთლიანი წარმოდგენა გვქონდეს Activity ს სიცოცხლის ხანგრძლივობაზე, შეგვიძლია იგი წარმოვადგინოთ გრაფიკული სახით:

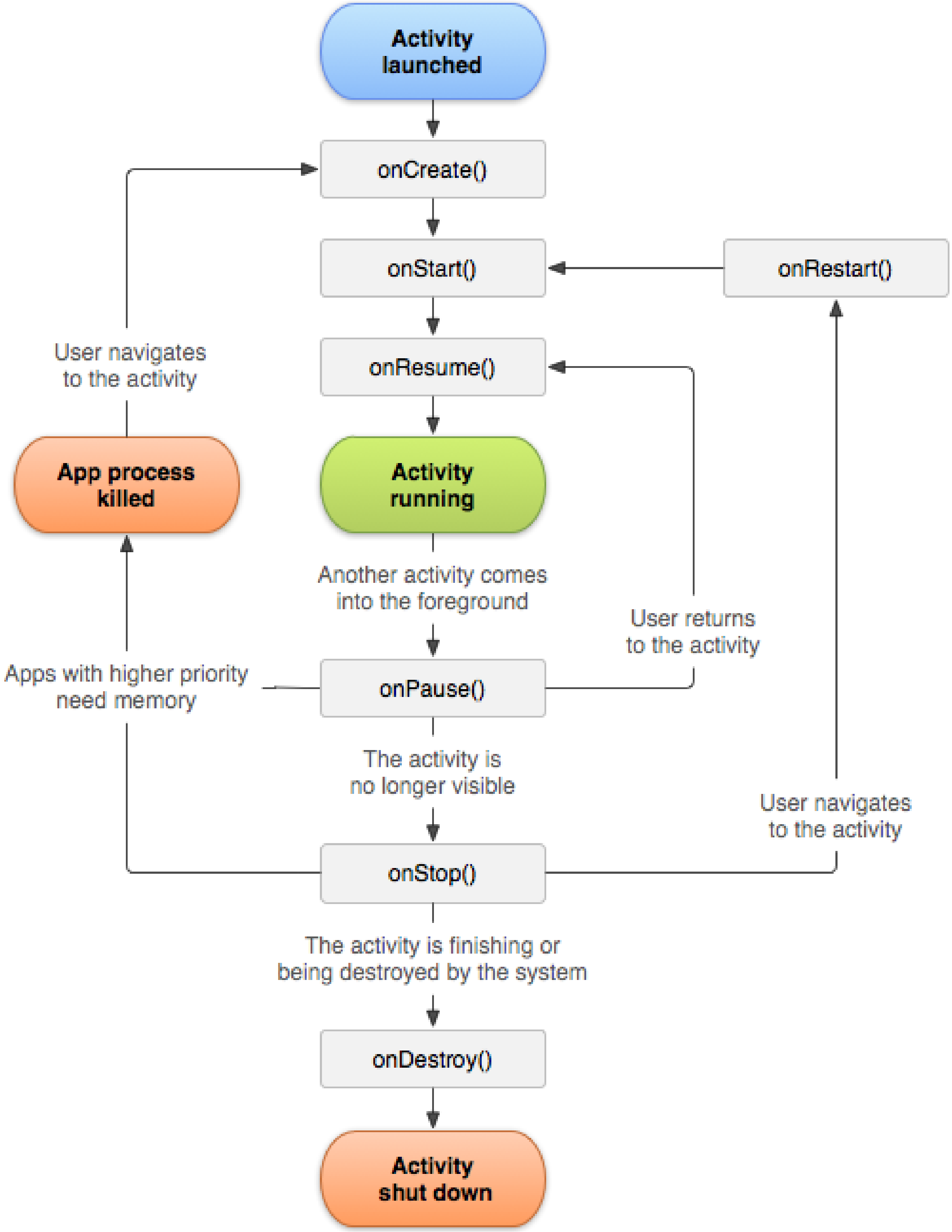

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Figure 1

2.2 Activity ს მონიტორინგი კატეგორიებად

ზემოთ განხილული Event ები შესაძლოა დავყოთ კატეგორიებად, რომელთაგანაც ძირითადს წარმოადგენს შემდეგი 3:

- Entire lifetime - განსაზღვრავს მთელ სიცოცხლის ხანგრძლივობას დაწყებული onCreate() დან onDestroy() ს ჩათვლით, ამ შემთხვევაში იგულისხმება, რომ სანამ მოხდება აპლიკაციის განადგურება onDestroy() მეთოდში უნდა მოხდეს წვდომის დახურვა ყველა რესურსზე რომელსაც აპლიკაცია იყენებს.
- Visible lifetime – განსაზღვრავს ციკლს onStart()-onStop() ის ჩათვლით, რომლის დროსაც მომხმარებელს უშუალოდ აქ წვდომა მიმდინარე Activity ზე
- Foreground lifetime – განსაზღვრავს ციკლს onResume()-onPause() ის ჩათვლით. ამ დროს მიმდნარე Activity არის ყველა სხვა Activity ზე წინ და მომხმარებელს უშუალოდ მასზე უწევს მოქმედებების ჩატარება.

3.0 Intent ობიექტი

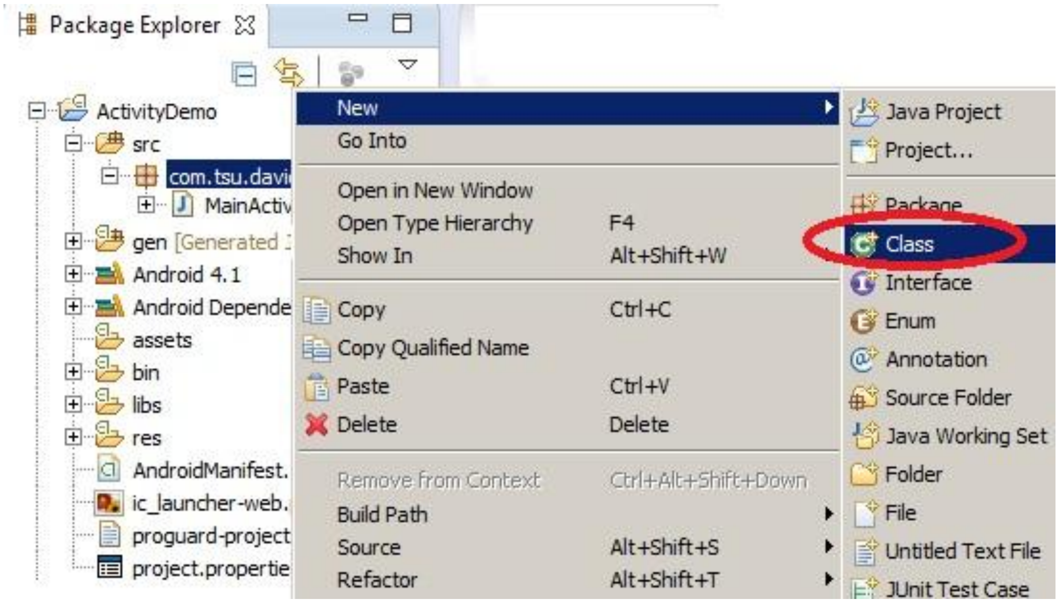
ხშირ შემთხვევაში საჭიროა ერთი რომელიმე Activity დან სხვა რომელიმეზე მიმართვა, მისთვის გარკვეული პარამეტრების გადაცემა და ა.შ . ასეთ შემთხვევაში აპლიკაციაში გვჭირდება საშუალება რომელიც რაღაც ხდომილებაზე გამოიძახებს სხვა Activity ს , გადასცემს მას პარამეტრებს, დაელოდება მის დასრულებას და ა.შ.

Intent წარმოადგენს ზუსტად ამ საშუალებას რომელსაც ვიყენებთ ყოველივე ზემოთთქმულის განსახორციელებლად. იმისათვის რათა ყოველივე ეს განვახორციელოთ საჭიროა შემდეგი:

1. არსებულ პროექტში შევქმნათ რაიმე Activity რომელზეც უნდა მოხდეს მიმართვა და რომელიც აუცილებლად იქნება Activity საბაზო კლასის შთამომავალი
2. მოვახდინოთ კონფიგურაციის ფაილში (AndroidManifest.xml) შესაბამისი ინფორმაციის დამატება
3. შესაბამისი შრის (ვიზუალური კომპონენტების განლაგების/layout) ის დამატება
4. ბიზნეს ლოგიკის საფუძველზე განვსაზღვროთ ხდომილების მონაკვეთი (ღილაკზე დაჭერა და ა.შ) და შევქმნათ Intent ობიექტი და გამოვიძახოთ შესაბამისი Activity

3.1 ახალი Activity ის დამატება

ახალი Activity ს დასამატებლად საჭიროა Eclipse გარემოში, მოვძებნოთ პროექტი რომელზეც ვმუშაობთ და პაკეტი რომელშიც არის Java კლასები, ამ პაკეტის მონიშვნისა და მარჯვენა ღილაკზე დაჭერის შემდეგ შევქმნათ ახალი Java კლასი, ისე როგორც ეს სურათზეა ნაჩვენები:



და დავარქვათ მას რაიმე სახელი, მაგალითად InfoActivity:

ამის შემდეგ ვხსნით ახლადშექმნილ java კლასს (InfoActivity ჩვენს შემთხვევაში) , ვხდით Activity კლასის მემკვიდრედ და უფარავთ onCreate() მეთოდს:

```
import android.app.Activity;

public class InfoActivity extends Activity {


  public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);

         //რაც შეეხება R.layout.infoactivity ს ის წარმოადგენს ერთ ერთი შრის (layout) ის იდენტიფიკატორს,
        //რომელზეც მოგვიანებით ვისაუბრებთ
         setContentView(R.layout.infoactivity);


  }
//……….
}
```

3.2 კონფიგურაციის ფაილში შესაბამისი მეტა ინფორმაციის ჩამატება

აპლიკაციამ უნდა იცოდეს ახალი Activity ს შესახებ ინფორმაცია, ამიტომ საჭიროა ახალ დამატებულ Activity ზე ჩავამატოთ AndroidManifest.xml ფაილში შესაბამისი მეტა ინფორმაცია, შესაბამისად ვხსნით AndroidManifest.xml ფაილს რომელიც არის პროექტის ძირეულ დირექტორიაში და უკვე გაწერილი <activity> ტეგის ქვემოთ/ზემოთ ვამატებთ ახალს, მაგალითად ჩვენს შემთხვევაში მას ექნება შემდეგი სახე:

```
<activity
 android:name=".InfoActivity"
 android:label="Some Info Activity">
 <intent-filter>
     <action android:name="com.tsu.david.activity.INFO"/>
     <category android:name="android.intent.category.DEFAULT"/>
 </intent-filter>
</activity>
```

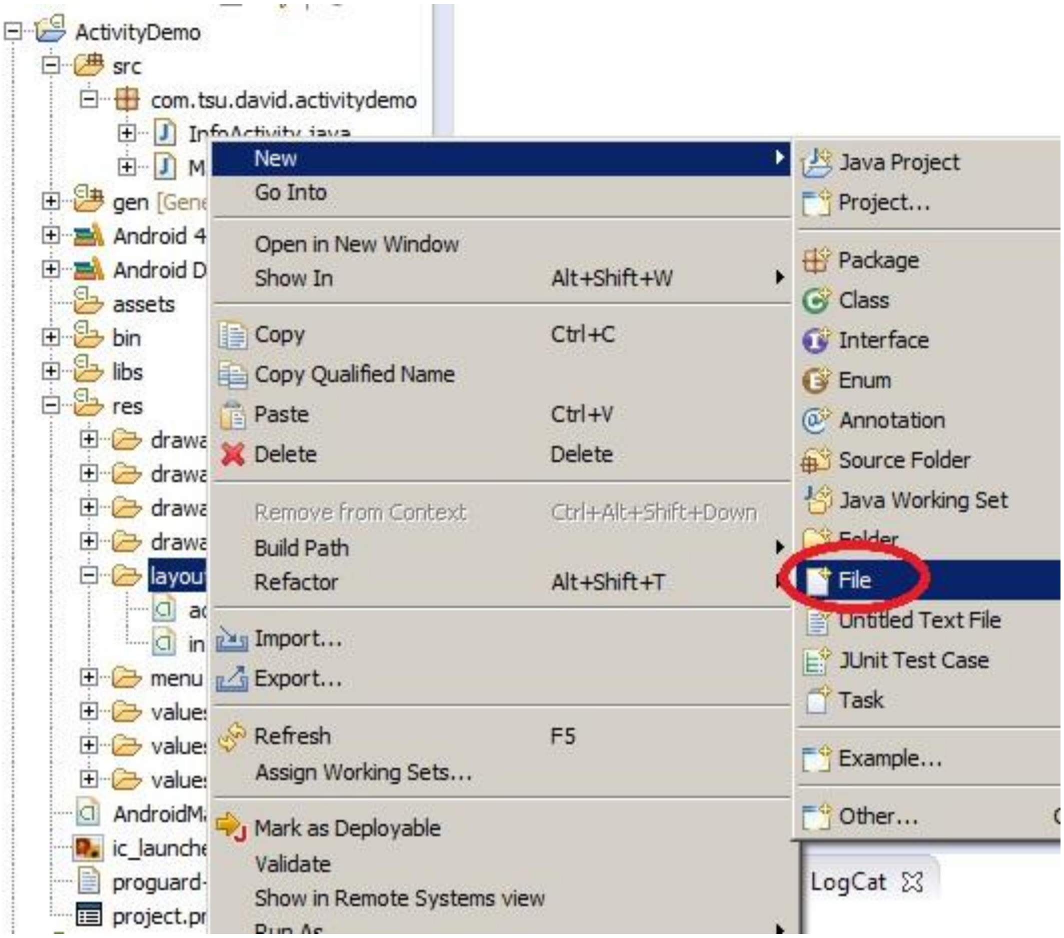
მისი სახელი (android:name) უნდა ემთხვეოდეს კლასის სახელს, label ში ჩაწერილი სტრიქონი უბრალოდ გამოჩნდება ამ activity ზე გადასვლისას სათაურად. ყურადღება მივაქციოთ <intent-filter> ტეგს რომელსაც იყენებს Intent ობიექტი, აქ არის ორი ძირითადი ქვე ტეგი:

- <action> - ამ ტეგის სახელი უნდა იყოს უნიკალური რათა ამ სახელით შემდეგ Intent ობიექტს ვუთხრათ თუ რომელ Activity ზე გვინდა გადასვლა
- <category> – არსებობს სხვადასხვა კატეგორიის Activity ები, რომელთაგან ერთ–ერთი არის DEFAULT, რომელიც მიუთითებს რომ მასზე გადასვლისას წინა Activity გაუჩინარდება დროებით.

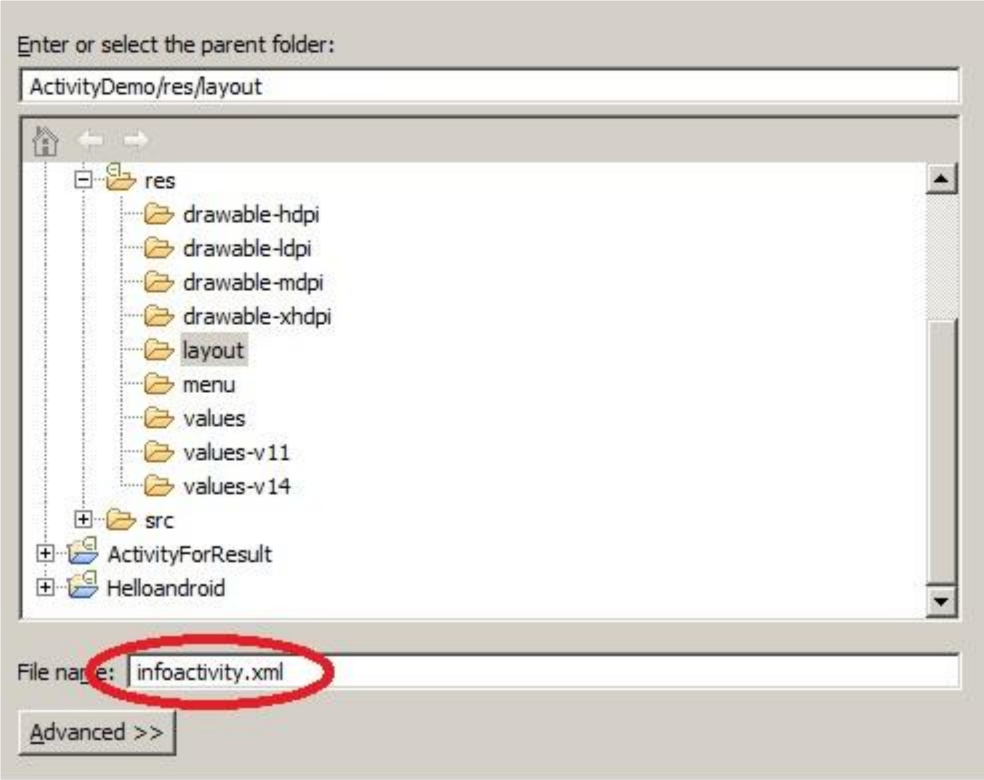
3.3 შესაბამისი ვიზუალური შრის დამატება

როგორც აღვნიშნეთ თითოეულ Activity ს შეესაბამება რაიმე ვიზუალური გარსი, სადაც მოთავსებულია კომპონენტები. სისტემამ უნდა იცოდეს რომელ Activity ს რომელი ვიზუალური შრე შეესაბამება, შესაბამისად თითოეული Activity სათვის შესაბამისი ფაილები უნდა იყოს აღწერილი ცალ ცალკე.

ვიზუალურ შრეებზე ინფორმაციის აღწერა ხდება xml ტიპის ფაილებში , პროექტის ძირეულ დირექტორიაში მოვძებნოთ res/layout ქვედირექტორია და დავამატოთ ახლადშექმნილი InfoActivity სათვის შესაბამისი ფაილი:



დავარქვათ სახელი იგივე მხოლოდ პატარა ასოებით (infoactivity ჩვენს შემთხვევაში):



გავხსნათ ახლადშექმნილი infoactivity.xml და ჩავწეროთ შემდეგი ტიპის ინფორმაცია:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
          android:orientation="vertical"
          android:layout_width="fill_parent"
          android:layout_height="fill_parent">
<TextView
    android:id="@+id/info_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />

</LinearLayout>
```

ზოგადად ვიზუალურ მხარეზე ვისაუბრებთ მე 4 თავში , მანამდე კი საკმარისია ვიცოდეთ, რომ ზემოთ აღწერილ xml ფაილში გვაქვს წრფივი layout (რაც ნიშნავს ვიზუალურ კომპონენტების მიმდევრობით განლაგებას) და რომელზეც არის მხოლოდ არარედაქტირებადი ტექსტური ველი.

3.4 Intent ობიექტის შექმნა და ახალი Activity ის გამოძახება

აქამდე განხილული პუნქტები აუცილებელია რათა შემდეგ მოვახდინოთ Intent ობიექტის შექმნა და ამ ახალ Activity ზე მიმართვა. ვთქვათ გვაქვს კოდში ხდომილება ღილაკზე დაჭერა რომლის შედეგადაც უნდა გამოვიდეს რაიმე ახალი ვიზუალური ობიექტი რაიმე ინფორმაციით და დაუშვათ ეს უნდა იყოს ჩვენს მიერ დამატებული InfoActivity, ასეთ შემთხვევაში უნდა მოვიძიოთ ღილაკზე დაჭერის ხდომილება და მასში ჩავწეროთ შემდეგი:

```
Intent intent = new Intent("com.tsu.david.activity.INFO");
startActivity(intent);
```

ყურადღება მივაქციოთ Intent კლასის კონსტრუქტორში გადაცემულ პარამეტრს „com.tsu.david.activity.INFO“ . როცა იძახება startActivity() სისტემა რეაგირებს შემდეგნაირად:

- ნახულობს AndroidManifest.xml ფაილს და ეძებს შესაბამის Activity ს რომელიც დავამატეთ შესაბამისი <action android:name ით რაც გადაცემულია Intent ის კონსტრუქტორში
- ახდენს შესაბამისი კლასის მოძიებას, რომელიც აუცილებლად უნდა იყოს სტანდარტული Activity კლასის მემკვირდრე და ფარავდეს onCreate() მეთოდს
- შედის შესაბამის Activity ში და ეძებს setContentView(R.layout.<xml ფაილის სახელი სადაც გაწერილია ვიზუალური მხარე>) და მხოლოდ ამის შემდეგ გამოაქ რეზულტატი ეკრანზე.

უნდა აღინიშნოს , რომ თუკი ვერ მოხდა რომელიმე პუნქტის შესრულება , მოხდება შესაბამისი Exception (გამონაკლისი სიტუაცია რომელიც გამოიყენება Java ში შეცდომების დამუშავებისას) და შედეგი იქნება ნაჩვენები logcat viewer კონსოლში.

3.5 პარამეტრების გაყოლება Intent ობიექტზე

ხშირად საჭიროა ერთი Activity დან მეორეზე გადასვლისას მეორესთვის რაიმე პარამეტრების გადაცემა (მაგალითად თუ პირველ Activity ზე ხდება სახელის და პაროლის შეყვანა ხოლო მეორემ უნდა შეამოწმოს და გაატაროს აუდენტიფიკაცია/ავტორიზაცია). რა თქმა უნდა არსებობს საშუალება რომელიც უზრუნველჰყოფს პარამეტრების გადაცემას, შეგვიძლია გამოვიყენოთ შემდეგი სინტაქსი:

```
Intent intent = new Intent("com.tsu.david.activity.INFO");
Bundle extras = new Bundle();
extras.putString("username", “dato.java”);
extras.putString("password", “some hashed passwpord”);
intent.putExtras(extras);

startActivity(intent);
```

ხოლო იმ Activity ში , რომელსაც ეს პარამეტრები სჭირდება , უნდა მოახდინოს მათი ამოღება რაც შემდეგი სინტაქსით ხდება:

```
Bundle extras = getIntent().getExtras();

String passedUsername = extras.getString("username");

String passedUsername = extras.getString("password");
```

3.6 ახალი Activity ს გამოძახება დასრულების მოლოდინში

არსებობს საშუალება რომელიც გულისხმობს ახალი Activity ს გამოძახებას ხოლო შემდეგ მისი უშუალო დასრულების ხდომილების დაჭერასაც, მაგალითად მოხმარების შემთხვევა შეიძლება იყოს შემდეგი :

1. A Activity დან ხდება B ს გამოძახება
2. B ახდენს რაიმე ლოგიკის რეალიზაციას
3. B ასრულებს ყველაფერს და მართვას უბრუნებს A ს შესაბამისი დასრულების სტატუსით

4. A „ელოდება“ Bს და როგორც კი დაასრულებს B თავის საქმეს გაგზავნილი კოდის მიხედვით ახდენს იდენტიფიცირებას მართლა A არის თუ რომელიმე სხვა და შემდეგ უკვე ახდენს რაიმე ლოგიკის რეალიზაციას.

A მ შესაძლოა B ს გარდა სხვა მრავალ Activity ს მიმართოს და მოახდინოს დალოდება , შესაბამისად უნდა იცოდეს რომელი დაბრუნდა და დაბრუნებულს ის „ცნობს“ ე.წ კოდის მიხედვით რომლითაც თავიდანვე აგზავნის .

ახლა კი მოვიყვანოთ კოდის მაგალითი სადაც ხდება გაგზავნა – დალოდება – რაიმე ლოგიკა

ჯერ მოვიყვანოთ ფრაგმენტი საიდანაც ხდება გაგზავნა MainActivity.java s ერთ – ერთი ფრაგმენტი

```
    private static final String MAIN_INFO_PREFIX = MainActivity.class.getName();
    private EditText username;
    private final int requestCode = 1;

    class SubmitListener implements View.OnClickListener {

            public void onClick(View v) {
                    String value = username.getText().toString();
                    Log.d("onClick()", "Clicked submit button in " + MAIN_INFO_PREFIX + " class");
                    Intent submittedIntent = new
Intent("com.david.activityforresult.SUBMITTEDACTIVITY");
                    Log.d("Value", value);
                    Bundle extras = new Bundle();
                    extras.putString("username", value);
                    submittedIntent.putExtras(extras);
                    startActivityForResult(submittedIntent, requestCode);

            }

    }
```

ამ კოდის მიხედვით , ღილაკზე დაჭერისას მოხდება username ის შენახვა და SUBMITTEDACTIVITY ს გამოძახება მეთოდით: `startActivityForResult(submittedIntent, requestCode);` ყურადღება მივაქციოთ რომ ვიძახებთ startActivityForResult() რომელიც გულისხმობს დალოდებას და მეორე პარამეტრად გადავცემთ კოდს რომლითაც უნდა ვიცნოთ ის როცა დაბრუნდება.

ახლა ვნახოთ რა ხდება როცა სისტემა გადავა შესაბამის Activity ზე:

```
public class SubmittedActivity extends Activity {

    public void onCreate(Bundle savedInstance) {
        super.onCreate(savedInstance);
        setContentView(R.layout.submittedactivity);
        Bundle extras = getIntent().getExtras();
        String passedUsername = extras.getString("username");

        Intent data = new Intent();
        data.setData(Uri.parse("You have 5 messages dear "+passedUsername));
        setResult(RESULT_OK, data);
        finish();
      }
    }
```

კოდიდან ხდება

- გადმოცემული Username ის ამოღება
- ახალი მესიჯის ფორმირემა
- setResult() მეთოდის გამოძახება რომელსაც პირველ პარამეტრად გადაეცემა სტატუსი თუ რამდენად წარმატებულად დასრულდა შესაბამისი Activity ს
- finish() მეთოდი საბოლოოდ ასრულებს Activity ს და მართვას უბრუნებს გამომძახებელს

დაბრუნებულის იდენტიფიცირება როგორც უკვე ვთქვით უნდა მოვახდინოთ უკვე გაგზავნილი კოდის მიხედვით, მაგრამ აუცილებელია რომ მოვახდინოთ `public void onActivityResult(int reqCode, int resultCode, Intent data)` მეთოდის გადაფარვა იქ , სადაც უნდა მოხდეს დაბრუნება, ჩვენს შემთხვევაში MainActivity.java ში. ამ მეთოდს ექნება შემდეგი სახე:

```
public void onActivityResult(int reqCode, int resultCode, Intent data) {
        if (requestCode == reqCode) {
              if (resultCode == RESULT_OK) {
                    String formattedString = data.getData().toString();
                    //Do something with text….
              }
        }
 }
```

პირველი პარამეტრი reqCode წარმოადგენს დაბრუნებულის კოდს რომელსაც ვადარებთ იმ კოდს რაც მანამდე გადავეცით გასაგზავნად (რადგან შეიძლება ბევრი გავაგზავნოთ და ამიტომ უნდა მივხვდეთ თუ რომელი დაბრუნდა).

მეორე პარამეტრი წარმოადგენს resultCode ს, ის აღნიშნავს თუ რაოდენ წარმატებულად დასრულდა გამოძახებული Activity.

მესამე პარამეტრი არის Intent ტიპის ობიექტი რომელშიც შესაძლოა იყოს რაიმე ინფორმაცია რაც გამოგზავნილია უკვე გამოძახებული Activity დან (ჩვენს შემთხვევაში ფორმირებული ტექსტი)

4.0 Layout ები

ვიზუალური მხარის ერთ–ერთ ფუნდამენტურ ნაწილს წარმოადგენს Layout ები რომელსაც დაწვრილებით განვიხილავთ მომდევნო ქვე–თავებში. როგორც წინა თავის მაგალითებში ვნახეთ მათი აღწერა ხდება xml ტიპის ფაილებში თუმცა Android ს აქვს მხარდაჭერა თვითონ პროგრამული კოდიდან დინამიურად ხდებოდეს ვიზუალური ნაწილის ცვლილება (კომპონენტების დამატება/რედაქტირება/წაშლა) რასაც მოგვიანებით განვიხილავთ.

წინა თავებში ჩვენ განვიხილეთ Activity ები, მისი მახასიათებლები, ერთ–ერთ უმთავრეს მახასიათებელს რაც გააჩნია აპლიკაციას არის მისი ვიზუალური მხარე რომელიც Android ის შემთხვევაში წარმოდგენილია ე.წ View ების სახით. ნებისმიერი ვიზუალური კომპონენტი (ღილაკი, label და ა.შ) წარმოადგენს View ს ერთ ერთ რეალიზაციას. ცალკეულად ასეთი View ების (ვიზუალური კომპონენტების) განლაგება არც ისე მომგებიანია გარკვეული მიზეზების გამო, რომელთაგანაც ძირითადია შემდეგი:

- სხვადასხვა Device ებს გააჩნიათ სხვადასხვა რეზოლუციების მხარდაჭერა, შესაბამისად ღილაკების და სხვა ვიზუალური კომპონენტების უბრალოდ დადებამ შეიძლება გარკვეული პრობლემები გამოიწვიოს
- ზომების ავტომატური ცვლილებების შესაძლებლობა – შეიძლება აპლიკაციის ლოგიკიდან გამომდინარე საჭირო იყოს რაღაც მომენტში ფანჯრების გაზრდა–დაპატარავება, ასეთ შემთხვევაში არ უნდა მოხდეს ვიზუალური კომპონენტების არევა

ზემოთ აღნიშნული პრობლემების გადაჭრისათვის შემოღებულ იქნა ViewGroup ის ცნება რაც გულისხმობს კომპონენტების დაჯგუფებას რაიმე წესის მიხედვით და ეს წესი თავისთავად გულისხმობს როგორც სხვადასხვა ტიპის მოწყობილობებზე მორგებას ასევე ზომების და ა.შ დინამიურ ცვლილებასაც. ViewGroup ების კონკრეტულ რეალიზაციას ახდენენ ე.წ Layout ები. Android ის შემთხვევაში გვხვდება ძირითადად შემდეგი ტიპის Layout ები:

1. Linear Layout
2. Absolute Layout
3. Table Layout
4. Relative Layout
5. Frame Layout
6. Scroll View

თითოეულ ამ Layout ს გააჩნია სხვადასხვა ტიპის თვისებები რაც პროგრამისტს საშუალებას აძლევს თავის აპლიკაციას მოარგოს ის რომელიც უფრო მიმზიდველს და გამოყენებადს ხდის მას.

4.0.1 Linear Layout

Linear Layout სხვანაირად მას წრფივ Layout საც უწოდებენ, წარმოადგენს ViewGroup ს რომელიც ალაგებს ვიზუალურ კომპონენტებს ერთ სვეტად ან ერთ სტრიქონად (კონფიგურაციიდან გამომდინარე). ამ Layout ის მაგალითი ჩვენ უკვე ვნახეთ წინა თავებში, იგი აღწერილი იყო main.xml ფაილში და მას ჰქონდა დაახლოებით ასეთი სახე:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
  ........................................
   </LinearLayout>
```

ყურადღება მივაქციოთ პარამეტრს android:orientation რომლის დანიშნულებაა ამ layout ს მიუთითოს თუ როგორ უნდა განლაგდნენ მასზე ვიზუალური კომპონენტები (vertical გულისხმობს ერთ სვეტად განლაგებას, ხოლო horizontal კი ერთ სტრიქონად), არსებობს ატრიბუტები რომლებიც არის საერთო როგორც View ებისათვის (ღილაკები და ა.შ) ასევე ViewGroup ბისათვის, ეს ატრიბუტები არის შემდეგი:

ატრიბუტი	განმარტება
layout_width	განსაზღვრავს View/ViewGroup ის სიგრძეს
layout_height	განსაზღვრავს View/ViewGroup ის სიგანეს
layout_marginTop	განსაზღვრავს დამატებით მანძილს ზემოდან View/ViewGroup მდე
layout_marginBottom	განსაზღვრავს დამატებით მანძილს ქვემოდან View/ViewGroup მდე
layout_marginLeft	განსაზღვრავს დამატებით მანძილს მარცხნიდან View/ViewGroup მდე
layout_marginRight	განსაზღვრავს დამატებით მანძილს მარჯვნიდან View/ViewGroup მდე
layout_gravity	განსაზღვრავს მდებარეობას/პოზიციას მშობლის მიმართ
layout_weight	განსაზღვრავს დამატებით რამდენი ადგილი გამოეყოს ViewGroup ში View ს
layout_x	განსაზღვრავს View/ViewGroup ის მდებარეობის x კოორდინატს
layout_y	განსაზღვრავს View/ViewGroup ის მდებარეობის y კოორდინატს

ამ ატრიბუტთაგან layout_gravity და layout_weight გამოიყენება მხოლოდ Linear Layout და TableLayout ში (TableLayout ს მოგვიანებით განვიხილავთ).

აღსანიშნავია layout_width და layout_height ის შემდეგი ორი კონსტანტური მნიშვნელობა:

1. wrap_content – fill_parent ისაგან განსხვავებით, ამ პარამეტრის გამოყენების შემთხვევაში კომპონენტის ზომა განისაზღვრება თავისი რაიმე კონკრეტული მნიშვნელობით (ეს მნიშვნელობები განისაზღვრება უნიკალურად თითოეული View სთვის). აღნიშნულ კომპონენტს ენიჭება გარკვეული ზომა და არ იწელება მთელს screen ზე.
2. fill_parent - ამ პარამატრის დაყენებისას კომპონენტის (View ს) ზომა განისაზღვრება მშობლის ზომის მიხედვით (შეიძლება ითქვას რომ ის ავსებს მშობელ კომპონენტს, რომელზეც არის დადებული მიმდინარე ვიზუალური კომპონენტი)

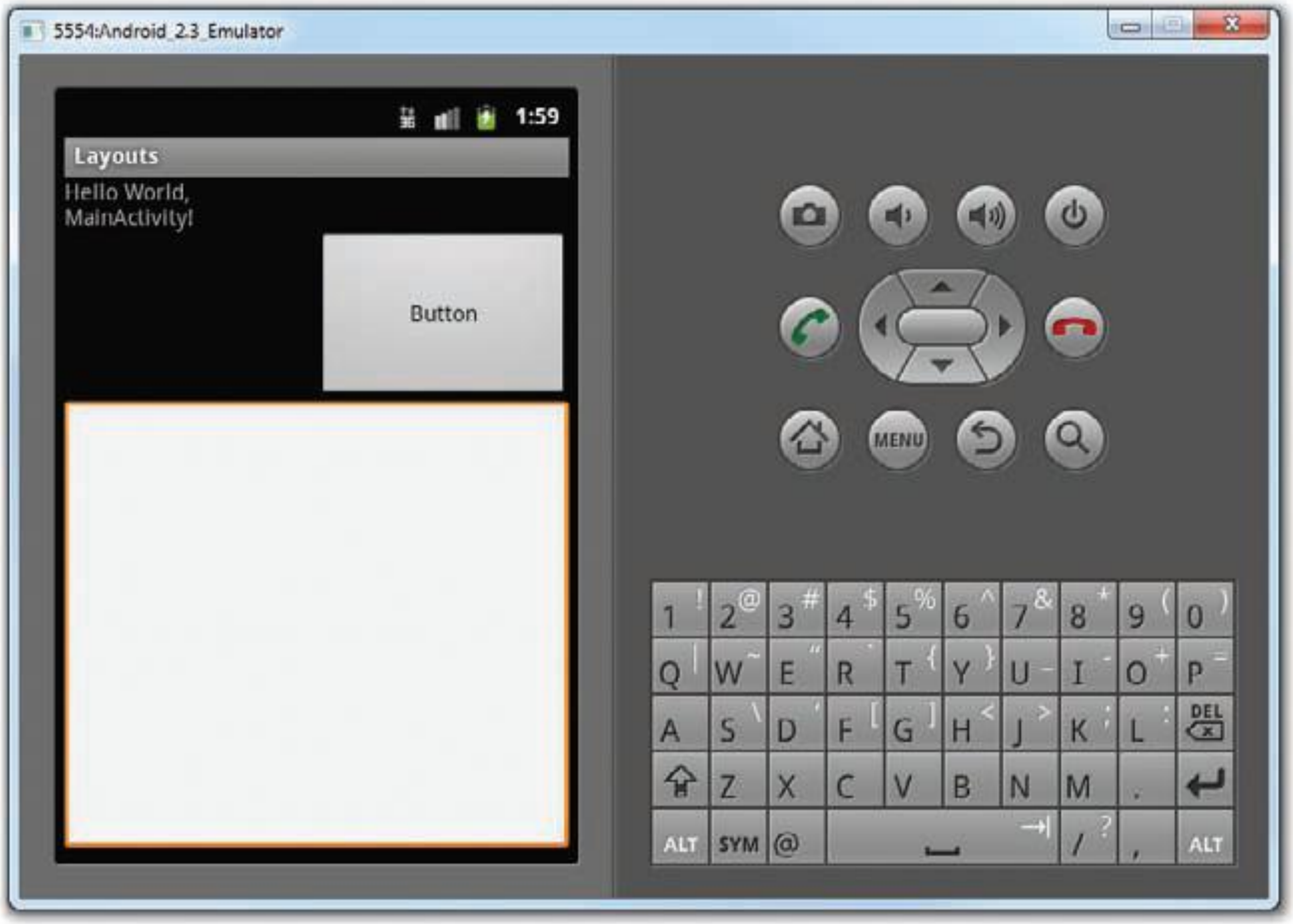
რასაკვირველია, სავსებით შეგვიძლია თითოეულ კომპონენტს მივცეთ კონკრეტული ზომები. Android ის შემთხვევაში განისაზღვრება შემდეგი ზომის შემდეგი ერთეულები:

ერთეული	განმარტება
dp (Density independent pixel)	წარმოადგენს რეკომენდირებულ ზომის ერთეულს. 160 dp = 1 inch , ამ ზომის ერთეულს გააჩნია ხშირი გამოყენება აპლიკაციებში
sp (Scale independent pixel)	თითქმის იგივეა რაც dp უბრალოდ sp ს ძირითადად იყენებენ font ების ზომების განსაზღვრაში
pt (Point)	Pt განისაზღვრება როგორც 1/72 ნაწილი ინჩის
px (Pixel)	განსაზღვრავს ჩვეულებრივ პიქსელს, მისი გამოყენება არ არის რეკომენდირებული, რადგან შესაძლოა აპლიკაციაში გარკვეული კომპონენტების ზომები არ დაჯდეს სწორად სხვადასხვა ტიპის device ებზე

ახლა კი მოვიყვანოთ main.xml ის მაგალითი რომელიც გვიჩვენებს ატრიბუტების გამოყენებას და ვნახოთ როგორ გამოჩნდება ის ემულატორში:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="105dp”"
        android:layout_height="”wrap_content”"
        android:text="”@string/hello”" />

    <Button
        android:layout_width="”160dp”"
        android:layout_height="”wrap_content”"
        android:layout_weight="”0.2”"
        android:text="”Button”
        android:layout_gravity=”right”" />

    <EditText
        android:layout_width="”fill_parent”"
        android:layout_height="”wrap_content”"
        android:layout_weight="”0.8”"
        android:textSize="”18sp”" />

   </LinearLayout>
```

დაკომპილირების შედეგად ემულატორზე/device ზე გამოჩნდება კომპონენტების დაახლოებით ასეთი განლაგება:



როგორც ვხედავთ კომპონენტების განლაგების ორიენტაცია (orientation ატრიბუტი) დაყენებულია vertical ზე, შესაბამისად უკვე ცხადია რომ კომპონენტები განლაგებული იქნება 1 სვეტად. პირველი კომპონენტი TextView არის მოთავსებული ყველაზე ზემოთ რაც ლოგიკურიცაა, ყურადღება მივაქციოთ შემდეგ კომპონენტს რომელიც არის ღილაკი და არის გაწეული მარჯვნივ. რა თქმა უნდა ეს შემთხვევითი არაა. რადგან ამ კომპონენტს აქვს დაყენებული layout_gravity=”right” ეს ნიშნავს რომ ის ამ სვეტში უნდა იყოს მარჯვენა მხარეს მოთავსებული ხოლო layout_weight=”0.2” კი ნიშნავს , რომ სვეტის ნაწილის 20% უნდა ეკავოს ამ კომპონენტს. მესამე კომპონენტში layout_weight =”0.8” ს შესაბამისად უკავია ადგილის 80% რაც ჩანს კიდეც სურათზე.

4.0.2 Absolute Layout

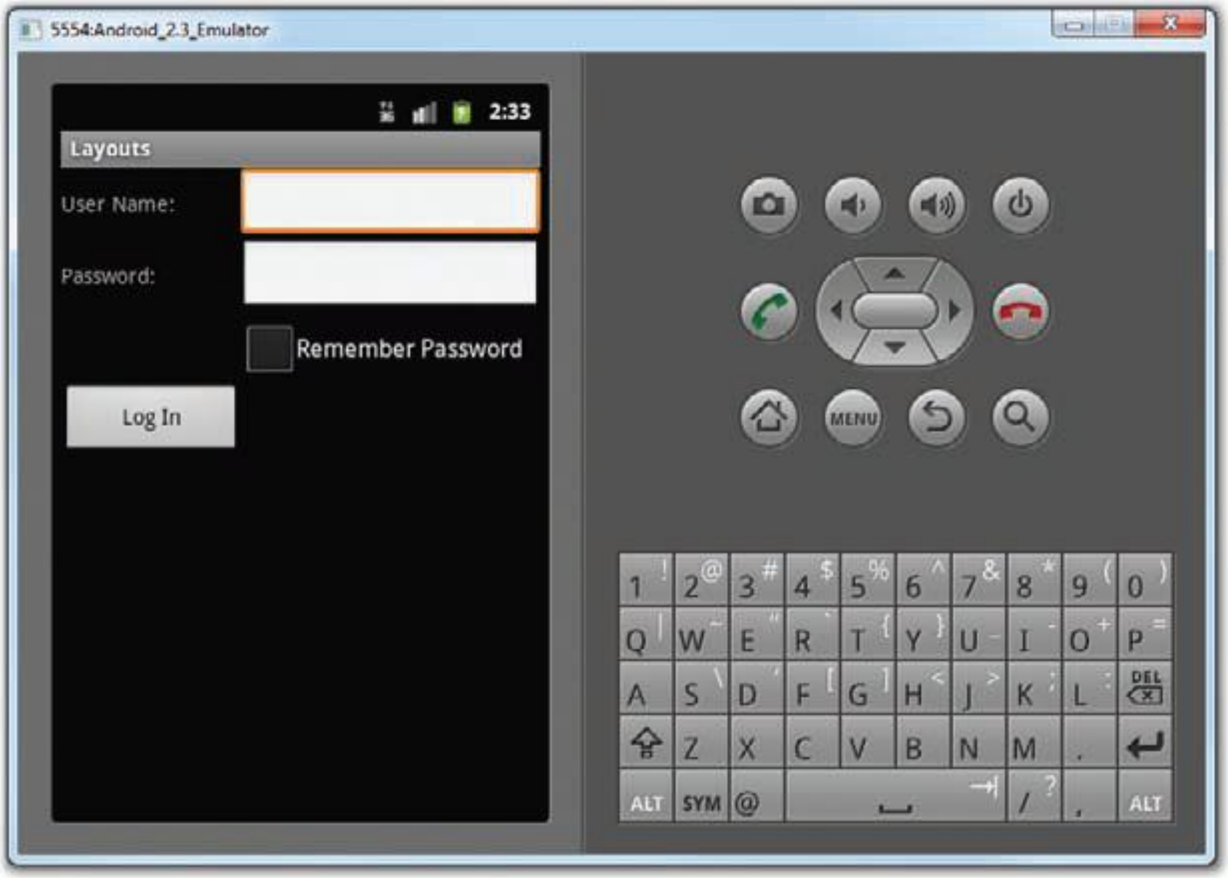
Absolute layout ი საშუალებას იძლევა განვსაზღვროთ კომპონენტების ზუსტი განლაგება. სიზუსტეში იგულისხმება კოორდინატების ზუსტი მითითება android_layout_x და android_layout_y ატრიბუტების გამოყენებით. რასაკვირველია ჩვენ წინასწარ არ ვიცით რა ფიზიკური ზომის მოწყობილობებზე შეიძლება მოხდეს აპლიკაციის გაშვება, აქედან გამომდინარე ეს layout არ არის რეკომენდირებული 1.5 ვერსიიდან. თუმცა მისი მხარდაჭერა არსებობს რაც იმას ნიშნავს, რომ მისი გამოყენება შესაძლებელია.

4.0.3 Table Layout

Table Layout გვაძლევს საშუალებას ვიზუალური კომპონენტები განვალაგოთ სტრიქონებისა და სვეტების ერთობლიობის ხარჯზე. შეგვიძლია წინასწარ განვსაზღვროთ თუ რამდენი სტრიქონი და სვეტი გვექნება და ამის შესაბამისად განვალაგოთ კომპონენტები, მაგალითისათვის შეგვიძლია მოვიყვანოთ აუდენტიფიკაციის ფორმა, რომელსაც აქვს სახელი, პაროლი, დამახსოვრება და შესვლის ღილაკი. მისი შესაბამისი xml ფაილი (თუ მთავარ Activity ზე ჩანს მაშინ main.xml) გამოიყურება შემდეგნაირად:

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TableRow>
        <TextView
            android:text="User"
            android:width="120px" />
        <EditText
            android:id="@+id/txtUserName"
            android:width="200px" />
    </TableRow>

    <TableRow>
        <TextView android:text="Password:" />
        <EditText
            android:id="@+id/txtPassword"
            android:password="true" />
    </TableRow>

    <TableRow>
        <TextView />
        <CheckBox
            android:id="@+id/chkRememberPassword"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Remember" />
    </TableRow>

    <TableRow>
        <Button
            android:id="@+id/buttonSignIn"
            android:text="Log" />
    </TableRow>

</TableLayout>
```

შესაბამისი პროექტის გაშვებისას მივიღებთ დაახლოებით ასეთ შედეგს:



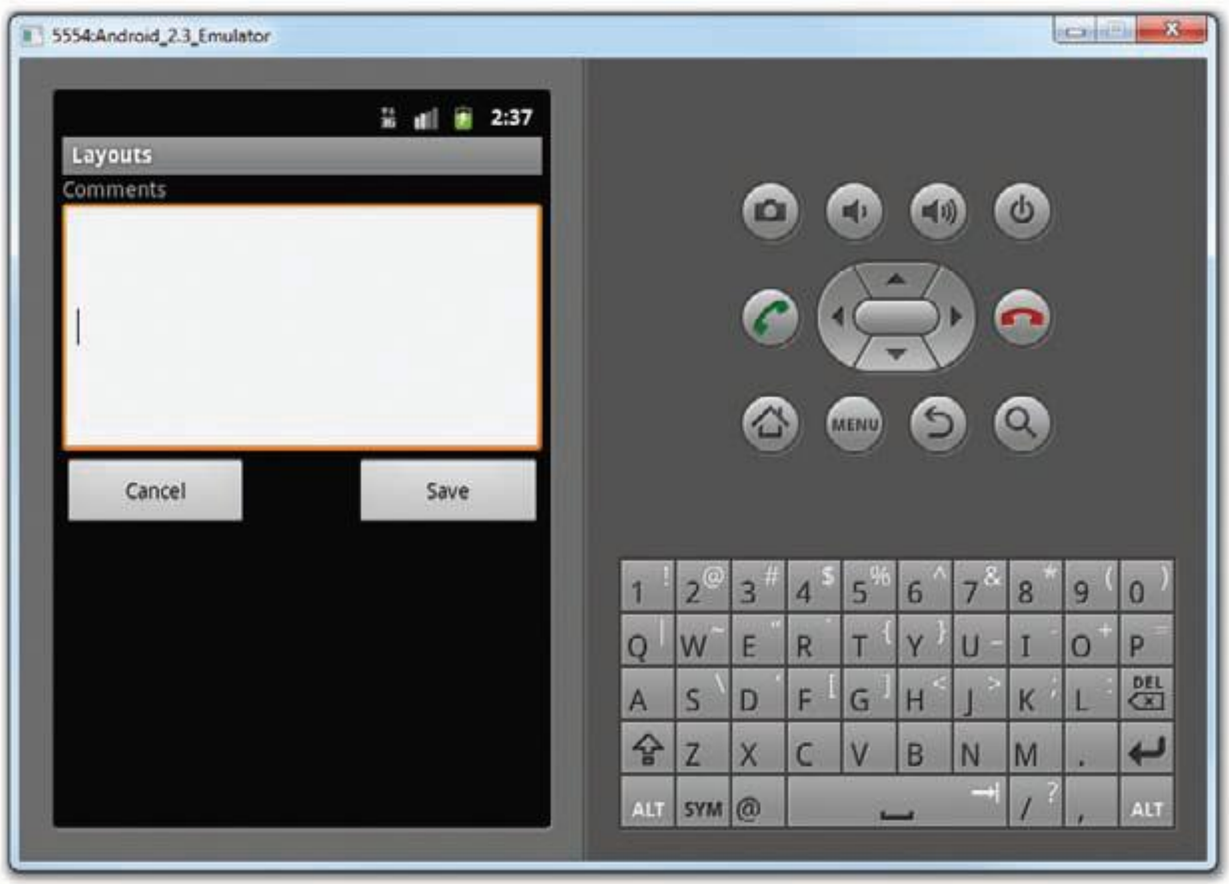
თუ კარგად დავაკვირდებით ზემოთ ნაჩვენებ xml ფაილს ვნახავთ, რომ სულ გვაქვს 4 TableRow ტეგი, რომელთაგანაც თითოეული განსაზღვრავს სურათზე ნაჩვენებ თითო სტრიქონს, ხოლო თითოეულ ამ ტეგში განსაზღვრული კომპონენტები კი წარმოადგენენ შესაბამის სვეტის ელემენტებს.

4.0.4 Relative Layout

ვიზუალურ კომპონენტებს შეგვიძლია დავუყენოთ მიმართებები, რაც იმას ნიშნავს, რომ შეგვიძლია განვსაზღვროთ რომელი ვიზუალური კომპონენტი რომელიმე სხვასთან რა მიმართებაში იმყოფება, სწორედ ამისათვის გამოიყენება relative layout . შეგვიძლია მოვიყვანოთ შემდეგი მაგალითი:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TextView
        android:id="@+id/lblComments"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="Comments" />

    <EditText
        android:id="@+id/txtComments"
        android:layout_width="fill_parent"
        android:layout_height="170px"
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true"
        android:textSize="18sp" />

    <Button
        android:id="@+id/btnSave"
        android:layout_width="125px"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/txtComments"
        android:layout_below="@+id/txtComments"
        android:text="Save" />

    <Button
        android:id="@+id/btnCancel"
        android:layout_width="124px"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/txtComments"
        android:layout_below="@+id/txtComments"
        android:text="Cancel" />

   </RelativeLayout>
```

პროექტის გაშვების შედეგად მივიღებთ დაახლოებით ასეთ სურათს:



ამ ყოველივეს მართვა ხორციელდება შემდეგი ატრიბუტების მეშვეობით:

ატრიბუტი	მნიშვნელობა
layout_alignParentTop	true/false იმის მიხედვით მოახდინოს თუ არა გასწორება მშობლის მიმართ ზემოთ
layout_alignParentLeft	True/false იმის მიხედვით მოახდინოს თუ არა გასწორება მშობელი კომპონენტის მიმართ მარცხნივ
layout_alignLeft	ახდენს კომპონენტის გასწორებას იმ კომპონენტის მარცხნივ რომლის იდენტიფიკატორის გადაცემაც ხდება
layout_alignRight	ახდენს კომპონენტის გასწორებას იმ კომპონენტის მარჯვნივ რომლის შესაბამისი იდენტიფიკატორის გადაცემაც ხდება
layout_below	ახდენს იმ კომპონენტის ქვემოთ მიმდინარე კომპონენტის განთავსებას, რომლის იდენტიფიკატორის გადაცემაც ხდება
layout_centerHorizontal	true/false იმის მიხედვით გვინდა თუ არა ამ კომპონენტის ცენტრის მიმართ გასწორება

განხილულ მაგალითში პირველი კომპონენტა ამ ორი პარამეტრის საფუძველზე:

```
android:layout_alignParentLeft="true"
android:layout_alignParentTop="true"
```

ეუბნება სისტემას რომ ეს კომპონენტი განთავსდეს ზედა მარცხენა კუთხეში (და თუ სურათს დავაკვირდებით ეს ასეც არის, რადგან სიტყვა comments წერია ზედა მარცხენა კუთხეში). შემდეგი კომპონენტი ამ სამი პარამეტრის საფუძველზე :

```
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true"
```

ეუბნება layout ს, რომ განთავსდეს იმ კომპონენტის მარცხნივ, რომლის იდენტიფიკატორიც არის lblComments (Comment ტექსტი ჩვენს შემთხვევაში), და იმის ქვემოთ რომლის იდენტიფიკატორიც ასევე არის lblComments და მოახდინოს ცენტრის მიმართ სწორება (რაც პარამეტრით android:layout_centerHorizontal=”true”). დანარჩენი ორი ღილაკიც იგივე პრინციპით არის განთავსებული , მათგან ერთი მარჯვნივ ხოლო მეორე მარცხნივ.

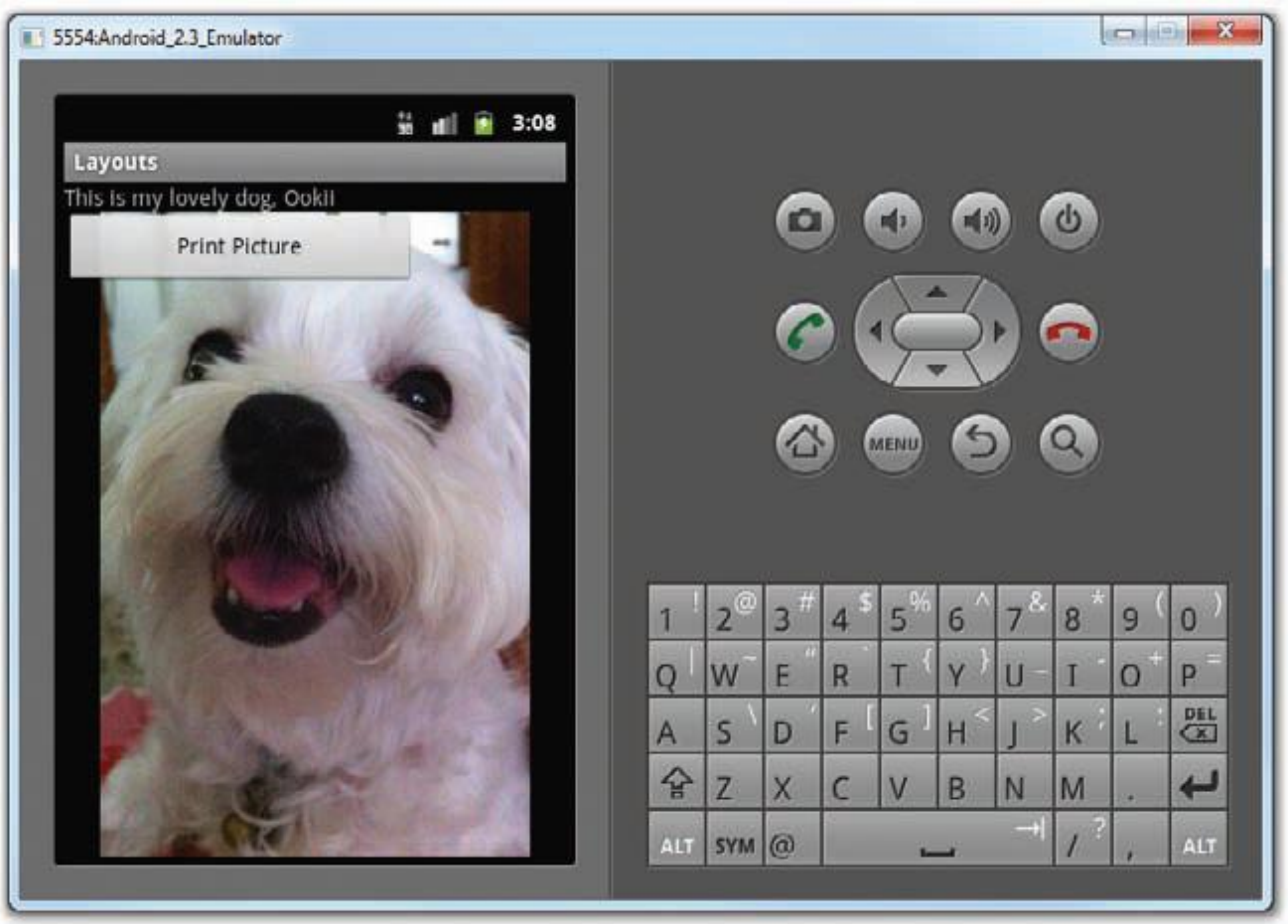
4.0.5 Frame Layout

ეს Layout ძირითადად გამოიყენება მაშინ, როდესაც გვინდა რაიმე ერთი კომპონენტი გამოვაჩინოთ მთელს Screen ზე , რომელზეც დევს აღნიშნული layout.თუ მოვახდენთ ერთზე მეტი კომპონენტის დამატებას, მაშინ მისი განთავსება მოხდება მოხდება ზედა მარცხენა კუთხეში. ხოლო წინა კომპონენტის მდებარეობა მთელს გამოყოფილ screen ზე შენარჩუნდება. განვიხილოთ მარტივი მაგალითის შესაბამისი layout.xml :

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TextView
        android:id="@+id/lblComments"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="This is my lovely dog Okii" />

    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true" >

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ookii" />
    </FrameLayout>

   </RelativeLayout>
```

ImageView წარმოადგენს კომპონენტს , რომელიც კითხულობს სურათს , src ატრიბუტი კი მიუთითებს გზას (ეს კომპონენტი განხილული იქნება მოგვიანებით). შედეგი კი იქნება დაახლოებით ასეთი:

თუკი ImageView ტეგის დაბლა მიუმატებთ რაიმე სხვა კომპონენტს, მაგალითად ღილაკს, შედეგი იქნება დაახლოებით ასეთი:



როგორც ვთქვით, ამ Layout ში მხოლოდ ერთი კომპონენტი უნდა იყოს დამატებული, ხოლო მეორეს დამატებამ კი გამოიწვია მისი ზედა მარცხენა კუთხეში განთავსება

4.0.6 ScrollView

ScrollView საშუალებას იძლევა, განვალაგოთ კომპონენტები წრფივი თანმიმდევროვით, ის წარმოადგენს FrameLayout ის ერთ ერთ სახეს. იგი როგორც წესი გამოიყენება სიის ტიპის კომპონენტების დასამატებლად რომელთა ელემენტების რაოდენობა შესაძლოა სრულად არ ეტეოდეს Device ის ეკრანზე და საჭირო იყოს touch(ფიზიკური შეხებით) ით ჩამოყოლა სიის ან ზემოთ ასვლა . სამაგალითოდ შეგვიძლია მოვიყვანოთ შემდეგი xml ფაილის შიგთავსი რომელიც განსაზღვრავს ამ layout ის გამოყენებას:

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <Button
            android:id="@+id/button1"
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button" />

    <Button
        android:id="@+id/button2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button" />

    <Button
        android:id="@+id/button3"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button" />

    <EditText
        android:id="@+id/txt"
        android:layout_width="fill_parent"
        android:layout_height="300px" />

    <Button
        android:id="@+id/button4"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button" />

    <Button
        android:id="@+id/button5"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button" />
 </LinearLayout>

</ScrollView>
```

გაშვებისას მივიღებთ დაახლოებით ასეთ შედეგს:

5554:Android_2.3_Emulator
3:13
Layouts
Button 2
Button 3
5554:Android_2.3_Emulator
3:13
Layouts
Button 4
Button 5

4.1 სხვადასხვა ხედის რეჟიმები

როგორც ცნობილია, უამრავ თანამედროვე device ს აქვს სხვადასხვა დამატებითი ფუნქციების მხარდაჭერა.ერთ–ერთი ასეთი ფუნქცია მდგომარეობს მოწყობილობის ფიზიკური მობრუნების ხდომილების დაჭერაში. მაგალითისთად: დაუშვათ ადამიანს უჭირავს რაიმე device და ამ მოწყობილობაზე გაშვებული აქვს აპლიკაცია. ასეთ შემთხვევაში თუ მოვახდენთ ფიზიკური აპარატის/device ის 180 გრადუსით მოტრიალებას, აპლიკაციაც შესაბამისად უნდა მოიქცეს, ყველაფერი ეს ერთიანდება ე.წ რეჟიმებში რომელთაგანაც გამოიყოფა შემდეგი ორი:

- Portrait - გულისხმობს ე.წ. Anchor ings, რომლის დროსაც აპლიკაციის აქტიური ნაწილის კომპონენტები ახდენენ უბრალოდ მობრუნებას და ისე განლაგებას
- Landscape – წინა რეჟიმისაგან განსხვავებით ახდენს ზომების ცვლილებას და არ შემოიფარგლება მხოლოდ კომპონენტების შემობრუნებით. უნდა გავითვალისწინოთ , რომ ასეთ შემთხვევაში ხშირად რჩება ცარიელი ადგილი მარჯვნივ რაც შეგვიძლია გამოვიყენოთ საჭიროებისამებრ.

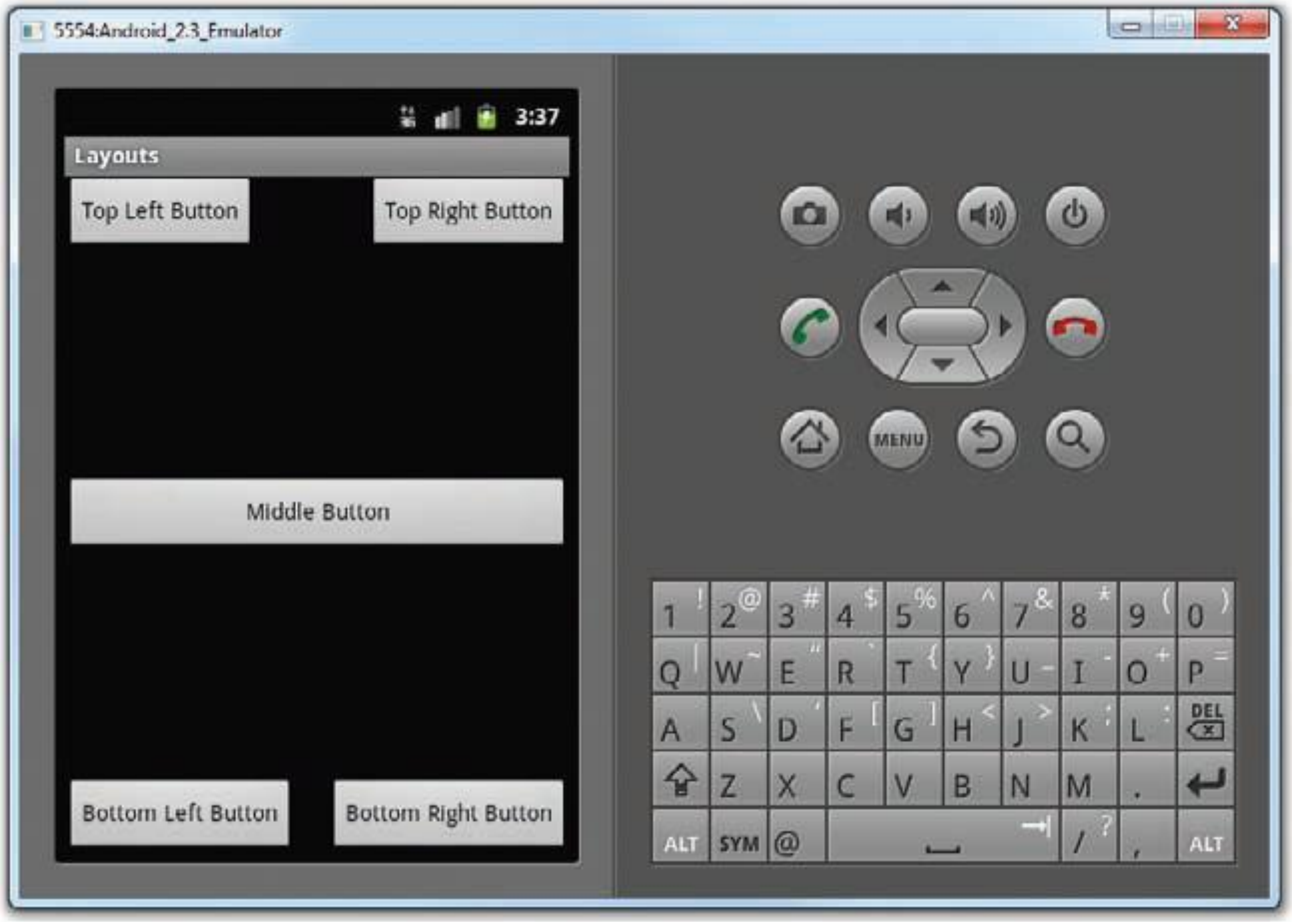
4.2 Anchoring რეჟიმის გამოყენება

ჩვენ განვიხილეთ სხვადასხვა layout ები, Anchoring ი შეგვიძლია კარგად გამოვიყენოთ Relative Layout ის შემთხვევაში , რადგან ეს layout წარმოადგენს ვიზუალური კომპონენტების ისეთ განლაგებას სადაც შესაძლებელია მივუთითოთ რომელი კომპონენტი რომელთან მიმართებაში როგორ უნდა იყოს. აქედან გამომდინარე შეგვიძლია მოვიყვანოთ შესაბამისი main.xml ფაილის მაგალითი რომელზეც განლაგებულია ხუთი ღილაკი:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="Top left button" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:text="Top Right Button" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:text="Bottom left Button" />
```

```xml
<Button
    android:id="@+id/button4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:text="Bottom Right Button" />

<Button
    android:id="@+id/button5"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:text="Middle Button" />

</RelativeLayout>
```

შესაბამისი პროექტის გაშვების შემდეგ დავინახავთ დაახლოებით ასეთ სურათს:



იმისათვის რომ პროექტში ჩადებული იყოს landscape ის მხარდაჭერა, პროექტის ძირეულ დირექტორიაში res, სადაც არის layout საქაღალდე , მის გვერდით დავამატოთ საქაღალდე layout-land და გადავიტანოთ layout საქაღალდიდან შესაბამისი xml ფაილი. ხოლო თუ გვინდა რომ landscape რეჟიმში რაიმე სხვანაირად გამოჩნდეს layout-land ში არსებულ xml ფაილში უნდა მოვახდინოთ შესაბამისი

ცვლილებები და არა layout საქაღალდის xml ფაილში. მაგალითად ჩვენს შემთხვევაში თუ დავამატებთ კიდევ რამოდენიმე ღილაკს landscape რეჟიმისათვის სპეციალურად , შედეგს ექნება შემდეგი სახე:

4.3 მდგომარეობის შენახვა რეჟიმის ცვლილებისას

როდესაც რაიმე აპლიკაციაზე ვმუშაობთ, შეგვყავს მონაცემები და ა.შ მნიშვნელოვანია გავითვალისწინოთ რეჟიმების ცვალებადობა (რაც ზემოთ აღვწერეთ). ამ შემთხვევაში პრობლემა გვხვდება შემდეგ საკითხთან მიმართებაში: როდესაც ხდება ვიზუალური ხედის რეჟიმის ცვლილება, აუცილებლად ხდება მიმდინარე activity ს გაჩერება და განადგურება და თავიდან შექმნა, შესაბამისად ის მდგომარეობა რაც მანამდე კომპონენტებს ჰქონდათ უბრალოდ იკარგება. ასეთ შემთხვევაში როგორღაც უნდა მოვახდინოთ რეჟიმის ცვლილების ხდომილების დაჭერა და საჭირო დროს შევინახოთ ეს მდგომარეობა ,ხოლო საჭირო დროს კი მოვახდინოთ მისი აღდგენა. იმისათვის, რომ დავიჭიროთ ხდომილება რეჟიმის ცვლილების, onCreate() მეთოდს ექნება დაახლოებით ასეთი სახე :

```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.Display;
import android.view.WindowManager;
```

```
public class MainActivity extends Activity {

        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);

                WindowManager wm = getWindowManager();
                Display d = wm.getDefaultDisplay();
                if (d.getWidth() > d.getHeight()) {
                        Log.d("Orientation", "Landscape mode");
                } else {
                        Log.d("Orientation", "Portrait mode");
                }
        }
    }
```

ხოლო იმისათვის რათა მოვახდინოთ რაიმე ობიექტის შენახვა, უნდა გადავფაროთ მეთოდი onRetainNonConfigurationInstance() ხოლო შენახული ობიექტის ამოსაღებად კი უნდა მივმართოთ მეთოდს getLastNonConfigurationInstance(). საბოლოოდ კი ზემოთ დაწერილ კლასს ექნება შემდეგი სახე:

```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.Display;
import android.view.WindowManager;

public class MainActivity extends Activity {

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);

                Object savedObject = getLastNonConfigurationInstance();
                if (savedObject != null) {
                        Log.d("Saved instance", savedObject.toString());
                }

                WindowManager wm = getWindowManager();
                Display d = wm.getDefaultDisplay();
                if (d.getWidth() > d.getHeight()) {
                        Log.d("Orientation", "Landscape mode");

                } else {
                        Log.d("Orientation", "Portrait mode");
                }
        }

        @Override
        public Object onRetainNonConfigurationInstance() {
                // ---rasac davabrunebT aq is iqneba shenaxuli---
                return "kukusha";
        }
    }
```

4.4 Layout ების პროგრამულად დამატება

აქამდე ჩვენ განვიხილეთ სხვადასხვა ტიპის layout ები , რომლის შესახებ ინფორმაციას ვწერდით შესაბამის xml ფაილში. ხშირად არის შემთხვევა როდესაც საჭიროა დინამიურად მოხდეს კომპონენტების გამოჩენა ისე, რომ წინასწარ შეიძლება არც ვიცოდეთ რა და როგორ დაგვჭირდება. Xml ფაილი არის სტატიკური, მასში ვწერთ საწყის მდგომარეობას, ხოლო შემდეგ თვითონ პროგრამული კოდიდან არის შესაძლებლობა ამ ყველაფრის კონტროლის: კომპონენტების დამატება/წაშლა და ა.შ. თუ როგორ კეთდება ეს შეგვიძლია ვნახოთ ქვემოთ მოცემულ მაგალითში:

```
LayoutParams params =
new LinearLayout.LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT);
            // ---create a layout---
            LinearLayout layout = new LinearLayout(this);
            layout.setOrientation(LinearLayout.VERTICAL);
            // ---create a textview---
            TextView tv = new TextView(this);
            tv.setText("This is a TextView");
            tv.setLayoutParams(params);
            // ---create a button---
            Button btn = new Button(this);
            btn.setText("This is a Button");
            btn.setLayoutParams(params);
            // ---adds the textview---
            layout.addView(tv);
            // ---adds the button---
            layout.addView(btn);
            // ---create a layout param for the layout---
            LinearLayout.LayoutParams layoutParam =
new LinearLayout.LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT);
            this.addContentView(layout, layoutParam);
```

ამ ფრაგმენტის საფუძველზე მოხდება LinearLayout ის შექმნა და მასზე ორი კომპონენტის TextView და Button ის დამატება. მნიშვნელოვანია , რომ ასეთ დროს არ უნდა გამოვიძახოთ setContentView() მეთოდი, რათა არ მოხდეს layout ის და ა.შ წაკითხვა შესაბამისი xml ფაილიდან.

4.5 სენსორის ხდომილებების "მოსმენა"

შესაძლებელია device ზე „მოუსმინოთ" შემდეგი ტიპის ხდომილებებს:

- onKeyUp() - იძახება მაშინ როდესაც ხდება გაშვება რაიმე კომპონენტის მაგრამ არ არის დამუშავებული ჯერ (მაგ. ღილაკზე თითის აშვება დაჭერის შემდეგ და ა.შ)
- onKeyDown() – იძახება მაშინ როდესაც ხდება დაჭერა მაგრამ არ არის დამუშავებული ჯერ აღნიშნული ხდომილება
- onMenuItemSelected() – იძახება მენიუს რომელიმე item ის მონიშვნისას
- onMenuOpened() – იძახება მენიუს გახსნისას

აღნიშნული ხდომილებები საჭიროა აპლიკაციის ლოგიკიდან გამომდინარე. იმისათვის რათა პროგრამულად ეს ყველაფერი ხელმისაწვდომი იყოს აუცილებელია გადავფაროთ შესაბამისი მეთოდი, მაგ: onKeyDown()

```
public boolean onKeyDown(int keyCode, KeyEvent event) {
        switch (keyCode) {
        case KeyEvent.KEYCODE_DPAD_CENTER:
                // ...
                break;
        case KeyEvent.KEYCODE_DPAD_LEFT:
                // ...
                break;
        case KeyEvent.KEYCODE_DPAD_RIGHT:
                // ...
                break;
        case KeyEvent.KEYCODE_DPAD_UP:
                // ...
                break;
        case KeyEvent.KEYCODE_DPAD_DOWN:
                // ...
                break;
        }
        return false;
}
```

DPAD_CENTER,RIGHT,LEFT განსაზღვრავს სენსორზე თითის მოძრაობას ცენტრში, მარჯვნივ,მარცხნივ და ა.შ. ხოლო შესაბამისი case ბლოკის ტანში კი იწერება ის ლოგიკა რაც გვინდა რომ შესრულდეს აღნიშნულ ხდომილებაზე.

4.6 ხდომილებების დარეგისტრირება სხვადასხვა View ებისთვის

წინა ქვეთავში განხილული ხდომილებები საკმაოდ სასარგებლოა თუმცა როგორ უნდა მოვიქცეთ მაშინ თუ გვსურს ისეთი ხდომილების დაჭერა როგორიცაა ღილაკზე დაჭერა? რაიმე View ს მონიშვნა და ა.შ? ასეთ შემთხვევაში წინასწარ უნდა დავარეგისტრიროთ Listener ები, ხოლო შემდეგ აღნიშნულ Listener ში მოვახდინოთ ლოგიკის რეალიზაცია (რაც უნდა ხდებოდეს დაჭერისას და ა.შ). ქვემოთ მოცემული მაგალითი გვიჩვენებს ძალიან მარტივ შემთხვევას:

```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity {
	private OnClickListener btnListener = new OnClickListener() {
		public void onClick(View v)
		{
			Log.d("Click Event", ((Button) v).getText() + " was clicked");
		}
	};

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);


		Button btn1 = (Button) findViewById(R.id.btn1);
		btn1.setOnClickListener(btnListener);
		Button btn2 = (Button) findViewById(R.id.btn2);
		btn2.setOnClickListener(btnListener);


	}
}
```

როგორც კი მოხდება btn1 ან btn2 ზე დაჭერა , სისტემა ავტომატურად გამოიძახებს btnListener ობიექტის onClick() მეთოდს საიდანაც ვგებულობთ უშუალოდ რომელ ღილაკზე მოხდა დაჭერა. მანამდე კი თითოეული ღილაკისთვის ვარეგისტრირებთ listener ს setOnClickLister() ის საშუალებით.

5.0 ვიზუალური კომპონენტები (შესავალი)

ვიზუალური კომპონენტები შეგვიძლია პირობითად დავყოთ შემდეგ სხვადასხვა სახის კომპონენტებად , რომელთაგანაც ძირითადები არის შემდეგი:

- Basic Views - განსაზღვრავს ისეთ კომპონენტებს , როგორიცაა ღილაკი, უბრალოდ ტექსტი, სურათი და ა.შ
- Picker Views – განსაზღვრავს ისეთი ტიპის ვიზუალურ კომპონენტებს რომელშიც მომხმარებელს უწევს რაიმე სიიდან ან სხვა საშუალებიდან გარკვეული მნიშვნელობების ამორჩევა.
- Spinner Views – განსაზღვრავს გრძელ სიებს, საიდანაც შესაძლოა მომხმარებელს უწევდეს ერთი ანდ რამოდენიმე ობიექტის ამორჩევა

5.1 ზოგიერთი Basic View

როგორც უკვე აღვნიშნეთ , Basic View ს მიეკუთვნება ისეთი კომპონენტები როგორიცაა ღილაკი, უბრალოდ ტექსტი, სურათი და ა.შ. განვიხილოთ ზოგიერთი მათგანი (კომპონენტების სიმრავლის გამო არ განვიხილავთ ყველა მათგანს).

Basic View დან ერთ ერთი ყველაზე გამოყენებადი არის Button View ანუ ღილაკი, რაც ზოგიერთ მაგალითში უკვე ვაჩვენეთ. ახლა ვნახოთ უფრო კონკრეტულად თუ როგორ შეგვიძლია ამ კომპონენტის გამოყენება. ნებისმიერ Activity ს აუცილებლად უნდა გააჩნდეს layout რომელზეც ვახდენთ კომპონენტების განთავსებას, გამომდინარე იქიდან თუ რა ლოგიკა გვინდა იდოს აპლიკაციის ვიზუალურ გვერდზე, layout საც ვირჩევთ შესაბამისად. დაუშვათ გვაქვს რაიმე layout რომელზეც გვინდა დავდოთ კომპონენტი ღილაკი. ეს შემდეგნაირად შეგვიძლია გავაკეთოთ:

```
<Button
        android:id="@+id/log_in_btn"
        android:text="@string/login_btn"
        android:enabled="false"/>
```

ტეგი <Button/> თავისთავად განსაზღვრავს ღილაკს, მისი ატრიბუტებიდან id განსაზღვრავს უნიკალურ იდენტიფიკატორს, რის საშუალებითაც შეგვიძლია პროგრამულად განვახორციელოთ ამ ობიექტზე წვდომა. text ატრიბუტი განსაზღვრავს იმ ტექსტს რაც ღილაკზე უნდა ეწეროს, ხოლო enabled ატრიბუტი იღებს მნიშვნელობას true ან false იმის მიხედვით აქტიური უნდა იყოს თუ არა ღილაკი. მივაქციოთ ყურადღება შემდეგ ჩანაწერების სინტაქსს : android:id=”@+id/log_in_btn” საიდანაც @+id/ არის სტანდარტული სინტაქსი რომელსაც მოყვება ჩვენს მიერ დარქმეული იდენტიფიკატორი log_in_btn ჩვენს შემთხვევაში. @string/login_btn კი ნიშნავს ,რომ სისტემა ეძებს strings.xml ფაილში ისეთი სტრიქონის მნიშვნელობას, რომლის გასაღებიც არის login_btn .

ერთ–ერთი გამოყენებადი Basic View ასევე არის EditText რომელიც წარმოადგენს რედაქტირებად შემავალ სტრიქონს , მისი ჩაწერის სინტაქსი არის შემდეგი:

```
<EditText
            android:id="@+id/pwd"
                android:inputType="textPassword" />
```

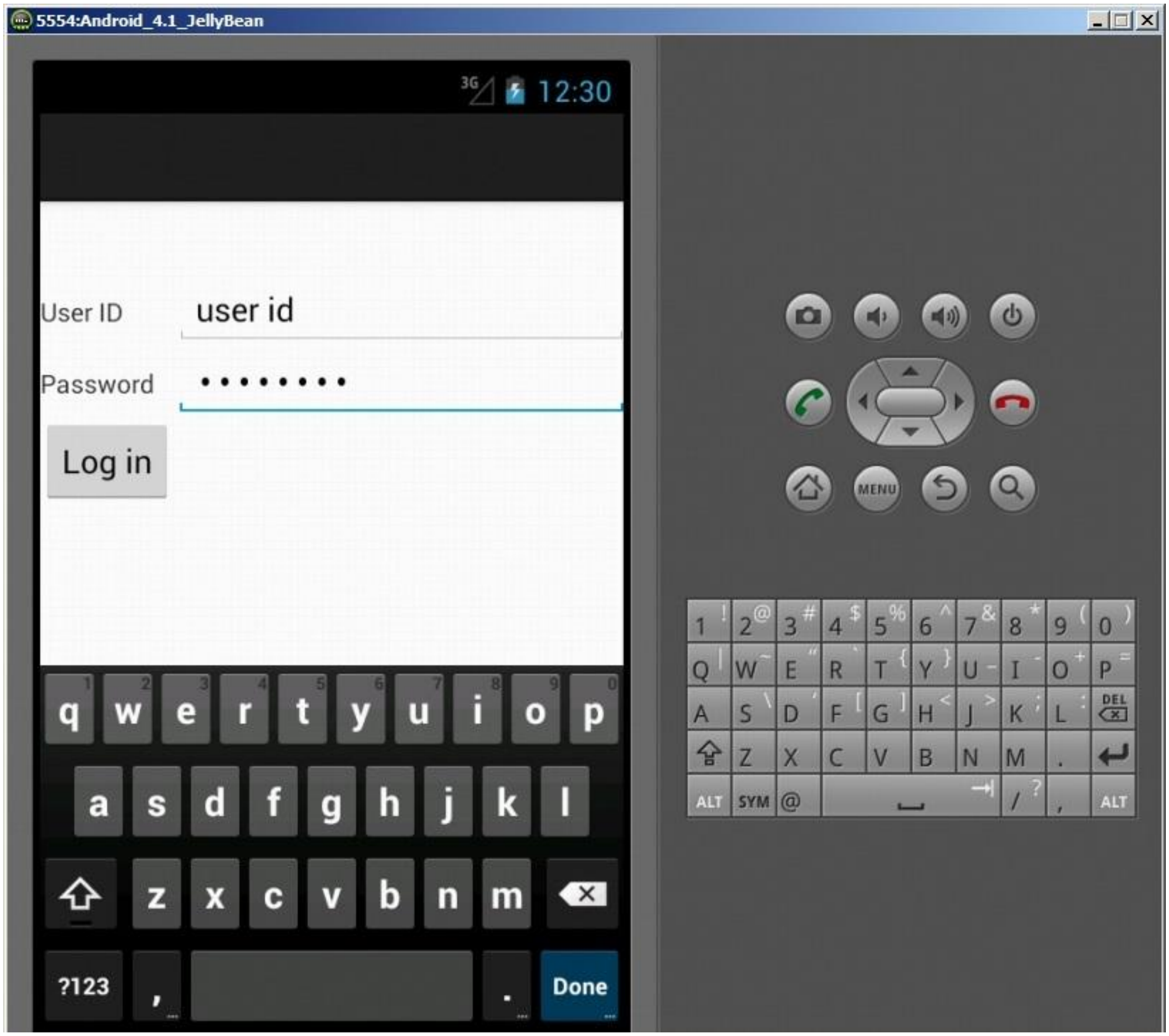
android:id ს აქვს იგივე დანიშნულება რაც წინა განხილული კომპონენტის შემთხვევაში, ხოლო რაც შეეხება android:inputType=”textPassword” ნიშნავს რომ ეს კომპონენტი მონიშნული იყოს როგორც პაროლის ველი.

ყველაზე მარტივი View არის TextView რომელიც წარმოადგენს უბრალო არა რედაქტირებად ტექსტს (ჩვეულებრივ label ს). მისი სინტაქსი არის შემდეგნაირი :

```
<TextView android:text="@string/password" />
```

აღნიშნულ ფრაგმენტში არ გვაქვს იდენტიფიკატორი თუმცა ეს ატრიბუტი ისევე როგორც ყველა სხვა დანარჩენ კომპონენტებს, მასაც გააჩნია.

ზემოთ აღნიშნული კომპონენტები შეგვიძლია გამოვიყენოთ ისეთი ტიპის აპლიკაციებში სადაც გვჭირდება აუდენტიფიკაცია/ავტორიზაცია (სადაც მომხმარებელს უხდება სახელის და პაროლის შეყვანა და ა.შ). მაგალითისთვის შეგვიძლია შევხედოთ შემდეგ სურათს:



ასევე ერთ–ერთი კომპონენტის ე.წ „რადიო ღილაკები“. მისი გამოყენებისათვის ჯერ უნდა აღვწეროთ RadioGroup ხოლო შემდეგ მასში RadioButton. შემდეგი კოდის ფრაგმენტი გვიჩვენებს აღწერის უმარტივეს მაგალითს:

```
<RadioGroup
        android:id="@+id/radio_group"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <RadioButton
            android:id="@+id/male_item"
            android:checked="true"
            android:text="@string/male" />

        <RadioButton
            android:id="@+id/female_item"
            android:text="@string/female" />
    </RadioGroup>
```

ყურადღება მივაქციოთ RadioGroup ტეგის orientation ატრიბუტს, საწყისი მნიშვნელობა ამ კომპონენტს აქვს vertical, შესაბამისად თუკი გვინდა , რომ ღილაკები განლაგდეს ჰორიზონტალურად , აუცილებელია ამ პარატრის გამოყენება და მისთვის horizontal მნიშვნელობის დაყენება. RadioButton ის პარამეტრი checked მიუთითებს რომ მიმდინარე ღილაკი უნდა იყოს მონიშნული. გაშვების შედეგად დავინახავთ დაახლოებით ასეთ სურათს:

5.2 ListView

ხანდახან საჭიროა მთელს ეკრანზე გრძელი სიის ჩვენება რათა მოვნიშნოთ ან ავირჩიოთ სხვადასხვა პარამეტრები, ასეთ შემთხვევაში ვიყენებთ ListView ს. როგორც ვიცით Activity წარმოადგენს იმ ვიზუალურ მხარეს რასთანაც ურთიერთქმედებას ახორციელებს კლიენტი, შესაბამისად თუკი ListView ს გამოტანა გვინდა მთელს ეკრანზე უნდა მოვნიშნოთ ისიც როგორც Activity. განსხვავება ჩვეულებრივი Activity ს რეალიზაციისაგან არის მხოლოდ ის, რომ ჩვენს მიერ დაწერილი კლასი უნდა იყოს არა Activity ს შთამომავალი არამედ ListActivity ს . მაგალითისთვის შეგვიძლია განვიხილოთ ისეთი ListView სადაც მომხმარებელს შეუძლია ერთზე მეტის მონიშვნა. ასეთ შემთხვევაში ჩვენს Activity ს ექნება შემდეგი სახე:

```
package ge.tsu.edu.client;

import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class SubjectListActivity extends ListActivity {
	private final String[] subjects = new String[] { "AAA", "BBB", "CCCC" };

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		ListView currentListView=getListView();
		currentListView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
		setListAdapter(new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_multiple_choice, subjects));
	}

	public void onListItemClick(ListView parent, View v, int position, long id) {
		parent.setItemChecked(position, parent.isItemChecked(position));
		Toast.makeText(this, subjects[position] , Toast.LENGTH_SHORT).show();
	}
}
```

ყურადღება მივაქციოთ იმ ფაქტს რომ არ გვჭირდება setContentView() მეთოდის გამოძახება onCreate() მეთოდში. რა თქმა უნდა ამ კოდში თავისთავად იგულისხმება რომ მომხმარებელი მას იძახებს გარედან შესაბამისი Intent ის საშუალებით. განვიხილოთ კოდის ის ფრაგმენტი სადაც ხდება უშუალოდ სიის პარამეტრების დაყენება:

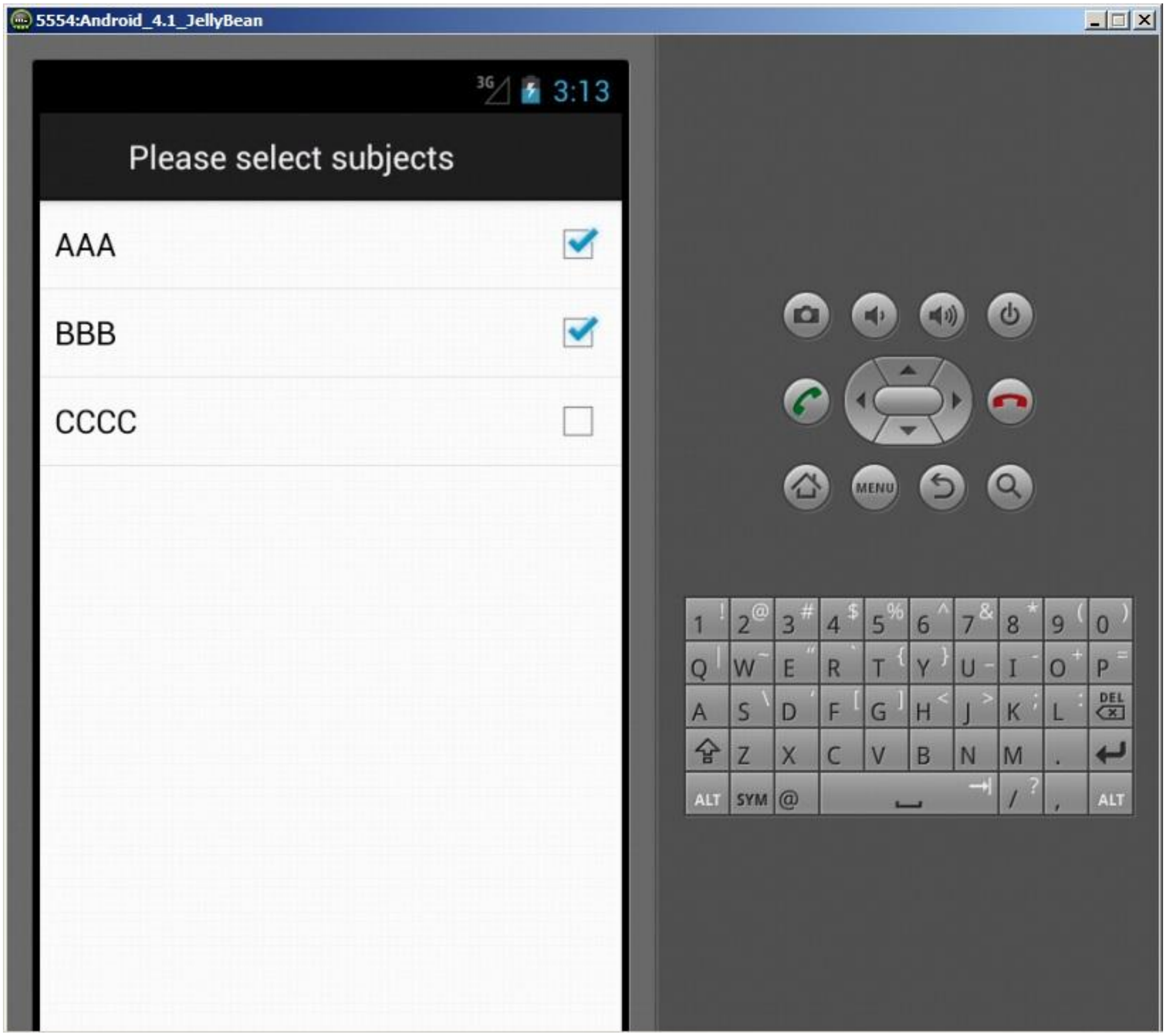
```
		ListView currentListView=getListView();
		currentListView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
		setListAdapter(new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_multiple_choice, subjects));
	}
```

პირველი ხაზის საშუალებით ვიღებთ უბრალოდ მიმდინარე ListView ს ობიექტს, შემდეგ ხდება რეჟიმის დაყენება მეთოდით setChoiceMode() , ჩვენს შემთხვევაში CHOICE_MODE_MULTIPLE ნიშნავს რომ მომხმარებელს შეუძლია ერთზე მეტი პუნქტის მონიშვნა. ბოლოს მეთოდი setListAdapter კი ემსახურება იმ სიის დაყენებას საიდანაც მომხმარებელს უწევს არჩევა (სიის მნიშვნელობების დაყენება), ხოლო პარამეტრი

android.R.layout.*simple_list_item_multiple_choice* კი ეუბნება Adapter ობიექტს , რომ მას გააჩნია მრავალი პუნქტის ერთდროულად მონიშვნის შესაძლებლობა.

მეთოდ onListItemClick() ვფარავთ იმისათვის რომ დავიჭიროთ მონიშვნის ხდომილება, მის ტანში კი დაწერილია რეალიზაცია თუ როგორ ვხვდებით მოცემულ მომენტში მომხმარებელმა რომელი ობიექტი მონიშნა.

SubjectListActivity ს გამოძახებას Intent ით ექნება შემდეგი სახე:



დამატებით რასაკვირველია გვჭირდება AndroidManifest.xml ფაილში შესაბამისი Activity ს აღწერა, ისეთნაირად როგორც ვაკეთებთ ჩვეულებრივი Activity ს აღწერას. მოვიყვანოთ ის ფრაგმენტი AndroidManifest.xml ფაილიდან რომელსაც შეესაბამება ზემოთ აღწერილი ListView:

```
<activity
  android:name=".SubjectListActivity"
  android:label="@string/title_subject_list">
  <intent-filter>
      <action android:name="ge.tsu.edu.client.SUBJECTLIST"/>
      <category android:name="android.intent.category.DEFAULT"/>
  </intent-filter>
 </activity>
```

5.3 SpinnerView

SpinnerView წარმოადგენს ერთ–ერთ ListView ს, ის ListView საგან განსხვავებით განსაზღვრავს ე.წ ჩამოსაშლელ სიას. აღნიშნულ კომპონენტს ვიყენებთ ზუსტად ისეთნაირად როგორც ListView ს, მცირე განსხვავებით. მოვიყვანოთ მისი აღწერის მაგალითი. პირველ რიგში ის უნდა აღვწეროთ შესაბამის xml ფაილში (სადაც ვიზუალურ კომპონენტებს ვწერთ ხოლმე).

```
<Spinner
    android:id="@+id/spinner1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

შემდეგ შესაბამის Activity ში შეგვიძლია ამოვიღოთ აღწერილი ობიექტი შემდეგნაირად:

```
final Spinner spinner = (Spinner) findViewById(R.id.spinner1);
```

ისევე როგორც ListView ს მასაც უნდა დავუყენოთ ArrayAdapter :

```
spinner.setAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_dropdown_item_1line,
new String[] { "A1", "A2", "A3" }));
```

ხოლო არჩევის ხდომილების დასაჭერად უნდა შევქმნატ AdapterView.OnItemSelectedListener ის შესაბამისი ობიექტი (რეალურად OnItemSelectedListener წარმოადგენს ინტერფეისს, როგორც ჯავას კურსიდან არის ცნობილი, მისი შესაბამისი ობიექტის შექმნა შეგვიძლია ორნაირად, 1. დავწეროთ კლასი რომელიც მოახდენს ამ ინტერფეისის რეალიზაციას ან შევქმნათ ანონიმური ობიექტი რის მაგალითსაც გთავაზობთ შემდეგ კოდის ფრაგმენტში)

```
spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

                    @Override
                    public void onItemSelected(AdapterView<?> parent, View v,int position,long  id)
{
                           Toast.makeText(getBaseContext(),
                                          spinner.getAdapter().getItem(position).toString(),
                                          Toast.LENGTH_SHORT).show();

                    }

                    @Override
                    public void onNothingSelected(AdapterView<?> arg0) {
                           // TODO Auto-generated method stub

                    }
             });
```

6.0 Data Persistence

Adndroid პლატფორმას გააჩნია შესაძლებლობა მონაცემთა შენახვა/დამახსოვრების. ამის გაკეთება შესაძლებელია შემდეგნაირად:

- ინფორმაციის შენახვა ე.წ SharedPreference ობიექტში
- ინფორმაციის შენახვა მონაცემთა ბაზაში

სანამ გადავიდოდეთ უშუალოდ იმის აღწერაზე თუ რომელი მექანიზმის გამოყენება როგორ ხდება, მანამდე აუცილებლად უნდა გავეცნოთ იმ ინფორმაციის სახეს და მნიშვნელობას რის შენახვასაც ვაპირებთ. ხშირად აპლიკაციებში საჭიროა რაიმე ინფორმაციის შენახვა და სანამ რომელიმე მექანიზმს გამოვიყენებთ აუცილებელია გავიგოთ თუ რა იგულისხმება ამ რაიმე ინფორმაციის უკან. ძირითადად შეიძლება შეგვხვდეს ინფორმაცია რომელიც მოიცავს მხოლოდ აპლიკაციის გარკვეულ პარამეტრებს. მაგ: შრიფტის ზომა, პაროლის დამახსოვრება და ა.შ. თუკი ასეთი ტიპის ინფორმაციის შენახვა გვსურს რეკომენდირებულია გამოვიყენოთ ე.წ SharedPreference ები. თუმცა შეიძლება ასევე მოვახდინოთ ერთი ტიპის ინფორმაციის ერთად დამახსოვრება, მაგ: საყიდლების სია Shopping ებისათვის და ა.შ. სხვა სიტყვებით რომ ვთქვათ თუ ვაპირებთ ისეთი ინფორმაციის დამახსოვრებას სადაც გვაქვს ერთი ტიპის ინფორმაციის სხვადასხვა ეგზემპლიარები, კარგი იქნებოდა ასეთი ინფორმაცია დავიმახსოვროთ მონაცემთა ბაზაში რომელსაც მოგვიანებით განვიხილავთ.

6.1 SharedPreference ების გამოყენება

აღნიშნული მექანიზმი ითვალისწინებს გარკვეული მონაცემების შენახვას, მას აქვს შესაძლებლობა რომ ეს ინფორმაცია ერთი აპლიკაციის ფარგლებში ხედვადი იყოს ყველა სხვა Activity სათვის. საბოლოო ჯამში ის ინახავს მნიშვნელობებს xml ფაილში რომელიც დევს აპლიკაციაში. პროგრამისტს არ უწევს ამ xml ის ხელით წაკითხვა , პოვნა და რედაქტირება. SharedPreference ების მექანიზმი გვთავაზობს ამ ყოველივეს ავტომატიზაციას.

ახლა მოვიყვანოთ კოდის ფრაგმენტი რომლის საშუალებითაც შევინახავთ სახელს და პაროლს ხოლო სხვა Activity ან იგივე აპლიკაცია მეორე და ა.შ გაშვებაზე წაიკითხავს ამ შენახულ ინფორმაციას და გამოიყენებს აპლიკაციაში ჩადებული ლოგიკის მიხედვით:

```
SharedPreferences preferences = getSharedPreferences("persistenceDemo", MODE_PRIVATE);
            String username = preferences.getString("user_name", "");
            String pwd = preferences.getString("pwd", "");
```

პირველად ვიღებთ SharedPreferences ის შესაბამის ობიექტს, getSharedPreferences() წარმოადგენს სტანდარტულ მეთოდს რომელიც არის განსაზღვრული Activity ში. პირველ პარამეტრად გადაეცემა რესურსის სახელი (თუ ასეთი რესურსი არ არსებობს იგი შეიქმნება ავტომატურად), ხოლო მეორე პარამეტრად კი გადაეცემა რეჟიმი (MODE_PRIVATE ნიშნავს რომ ეს რესურსი წვდომადია მხოლოდ აპლიკაციის დონეზე, შესაბამისად სხვა აპლიკაცია გარედან მას ვერ მიწვდება). კოდის შემდეგ ხაზზე ვიღებთ username სტრიქონს getString() მეთოდით, პირველ პარამეტრად გადაეცემა გასაღები , რომლითაც მანამდე მოვახდინეთ ამ სტრიქონის შენახვა, ხოლო მეორე პარამეტრად კი გადაეცემა ის მნიშვნელობა რაც გვინდა რომ დაგვიბრუნოს თუკი ამ გასაღებით არაფერი მოიძებნა. ასევეა სხვა ტიპის get() მეთოდებში. getBoolean() , getLong() და ა.შ

ახლა განვიხილოთ თუ როგორ ხდება ინფორმაციის შენახვა ამ ობიექტის საშუალებით , მოვიყვანოთ კოდის ფრაგმენტი რომელიც წერს იგივე username da password ს :

```
SharedPreferences preferences = getSharedPreferences("persistenceDemo", MODE_PRIVATE);
                              SharedPreferences.Editor editor = preferences.edit();
                              editor.putString("user_name", userName.getText().toString());
                              editor.putString("pwd", pwdText.getText().toString());
                              editor.commit();
```

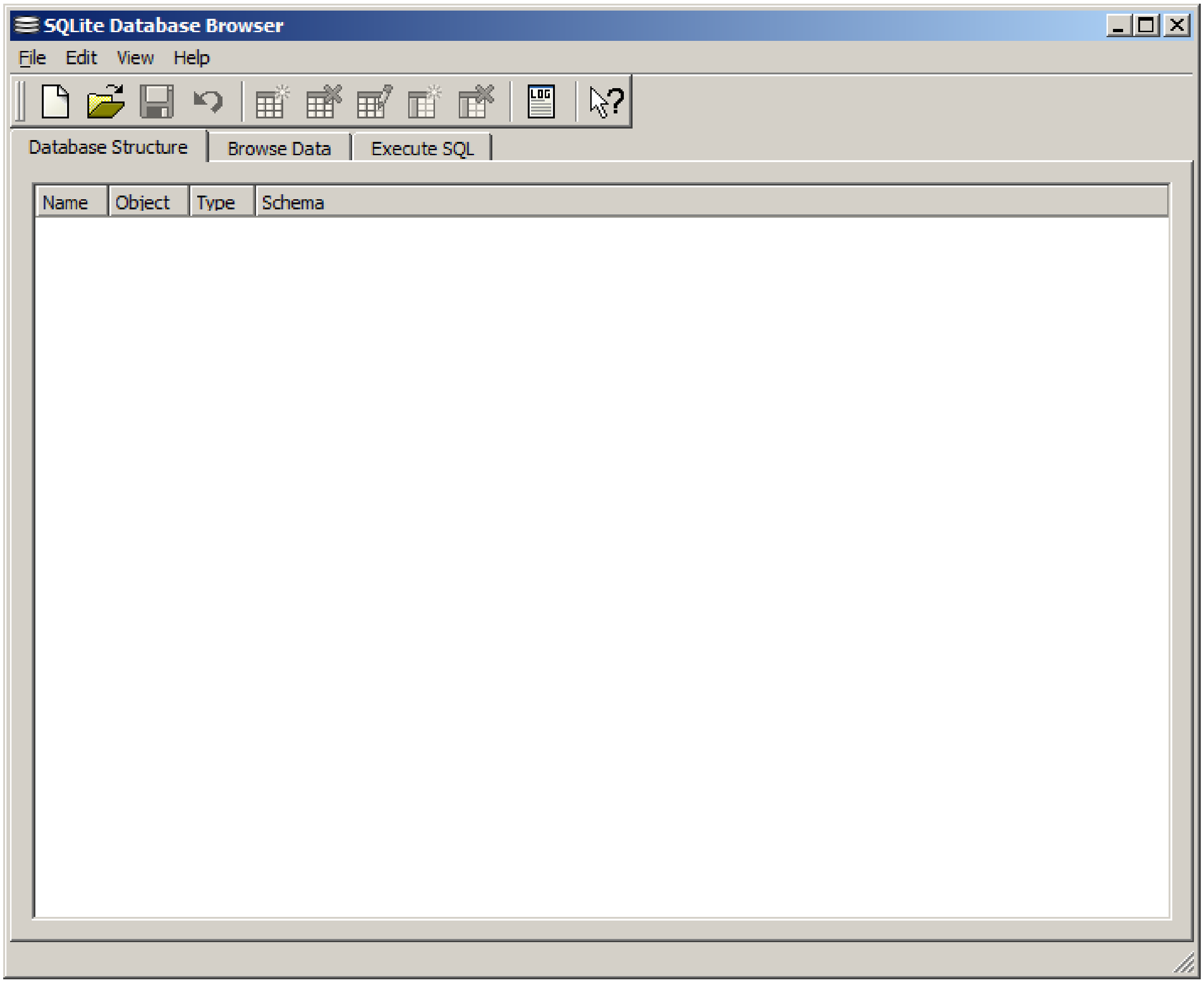
კოდის მეორე ხაზზე ვიღებთ Editor ობიექტს რომელიც წარმოადგენს SharedPreferences ის Inner ინტერფეისს და განკუთვნილია მონაცემთა რედაქტირება/ჩაწერისათვის, შემდეგ ორ ხაზზე ვდებთ ახალ სტრიქონებს გასაღებებით :user_name და pwd , ხოლო ბოლოს ვიძახებთ მეთოდ commit() რომელიც უშუალოდ ახდენს ჩაწერას/რედაქტირებას.

ასევე თუკი გვინდა რომ აღნიშნული რესურსი წვდომადი არ იყოს სხვადასხვა Activity ებში (ანუ მხოლოდ ერთ Activity ში გვინდა მისი გამოყენება) ნაცვლად getSharedPreferences() სა ვიყენებთ უბრალოდ getPreferences() რომელსაც გადაეცემა მხოლოდ რეჟიმი , პარამეტრი სახელი მას არ გააჩნია.

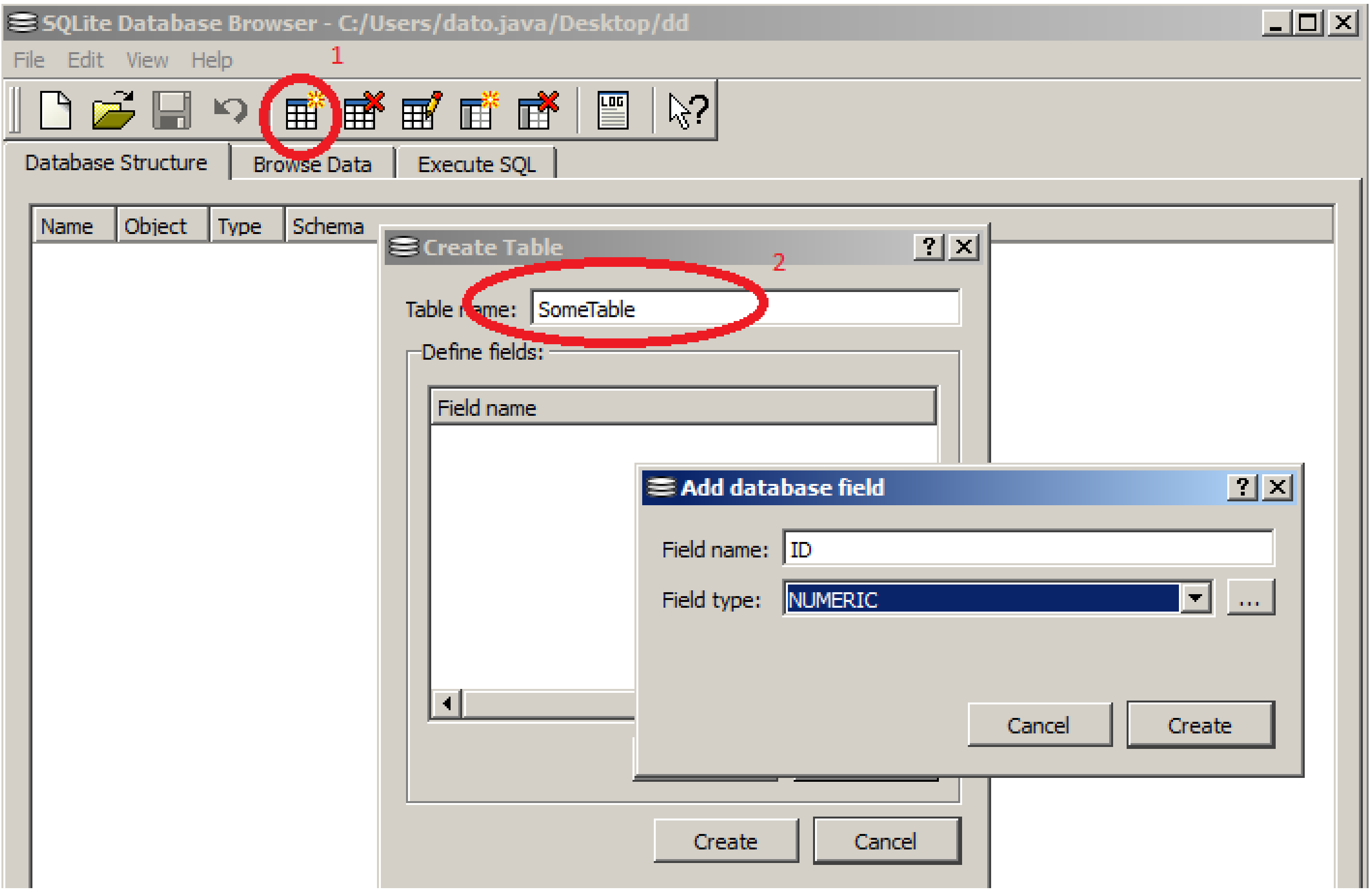
6.2 მონაცემთა ბაზის გამოყენება ინფორმაციის შესანახად

Android ს გააჩნია შესაძლებლობა გამოიყენოს მონაცემთა ბაზები ინფორმაციის შესანახად, იგი იყენებს ე.წ SQLite მონაცემთა ბაზას ამისათვის. სანამ უშუალოდ შევუდგებით მონაცემთა ბაზასთან მუშაობის მექანიზმის განხილვას, საჭიროა გადმოვწეროთ ბაზის კლიენტი რომლის საშუალებითაც შევქმნით/დავარედაქტირებთ მონაცემთა ბაზას. ერთ – ერთი ასეთი პროგრამა არის SQLite browser, რომლის გადმოწერაც შეიძლება ამ მისამართიდან : http://sourceforge.net/projects/sqlitebrowser/files/latest/download

გადმოწერის შემდეგ შეგვიძლია ამოვაარქივოთ და გავუშვათ ერთადერთი გამშვები ფაილი SQLite Database Browser 2.0 b1.exe (მომავალში ვერსია შეიძლება შეიცვალოს და 2.0 b1 ის ნაცვლად იყოს რაიმე სხვა). გაშვების შემდეგ დავინახავთ ასეთ ფანჯარას:



ახლა შევქმნათ ახალი ბაზა და შევინახოთ, ამისათვის დავაჭიროთ File->New Database შემდეგ დავარქვათ სახელი და მივუთითოთ ის დირექტორია სადაც გვინდა რომ შევინახოთ (პირველ ეტაპზე შევინახოთ სადმე, ხოლო შემდეგ უკვე მას მოვათავსებთ android ის პროექტის assets ქვედირექტორიაში). ამის შემდეგ პროგრამა შემოგვთავაზებს ცხრილების დამატებას . თუმცა შეგვიძლია ცხრილები დავამატოთ მენიუს ქვემოთ არსებული ობიექტიდან ისე როგორც ეს სურათზეა ნაჩვენები:



ამის შემდეგ შეიქმნება ცხრილი შესაბამის ბაზაში სახელად SomeTable რომელსაც ექნება ველი ID და კიდევ სხვა ველები, რის დამატებასაც გავაკეთებთ.

Execute SQL ის შესაბამისი ტაბიდან შეგვიძლია გავუშვათ რაიმე SQL Statement ი, ხოლო Browse Data დან კი შეგვიძლია ვნახოთ მონაცემები.

ახლა გადავიდეთ იმ პროგრამული ინტერფეისის აღწერაზე/გამოყენებაზე რაც გვჭირდება მონაცემთა ბაზებთან სამუშაოდ. როგორც წესი მონაცემთა ბაზა ფაილურად დევს ანდროიდის პროექტის assets ქვედირექტორიაში, თუ გვინდა რომ ბაზის არ არსებობის შემთხვევაში მოხდეს ბაზის შექმნა ამაზე თვითონ პროგრამული კოდიდან უნდა ვიზრუნოთ. პირველ ეტაპზე მაინც საჭიროა მოხდეს ბაზის ფაილურად კოპირება შესაბამის ადგილას, ეს ადგილი კი წარმოადგენს შემდეგ დინამიურ გზას: `"/data/data/" + getPackageName() + "/databases/<ბაზის სახელი რა სახელითაც გადაკოპირდეს>"`. getPackageName() აბრუნებს პაკეტის სახელს რომლითაც შევქმენით პროექტი. აქედან გამომდინარე პირველ ეტაპზე უნდა ვიზრუნოთ იმაზე არსებობს თუ არა შესაბამისი ბაზა, შესაბამისი კოდის ფრაგმენტი მოყვანილია ქვემოთ:

```
            String destPath = "/data/data/" + getPackageName() + "/databases/tsudb";
            File file = new File(destPath);
            if (!file.exists())
            {
            try {
            copyDB(getBaseContext().getAssets().open("tsudb"), new FileOutputStream(destPath));
            } catch (Exception ex) {
                    ex.printStackTrace();
                }
            }
```

copyDB() წარმოადგენს ჩვეულებრივ ჯავას მეთოდს რომელიც ერთი InputStream იდან აკოპირებს ფაილის შიგთავსს OutputStream ში. ჩვენს შემთხვევაში პირველად ვამოწმებთ არის თუ არა ასეთი ფაილი და შემდეგ ვიწყებთ კოპირებას თუ ბაზა/ფაილი არ არსებობს.

ახლა განვიხილოთ API რომლის საშუალებითაც ანდროიდის აპლიკაცია ახდენს ოპერაციებს ბაზაზე (ჩაწერა, წაშლა და ა.შ). პირველ რიგში უნდა მოვახდინოთ ბაზის გახსნა, ამისათვის გვჭირდება სტანდარტული SQLiteDatabase ობიექტი, მის შესაქმნელად ვიყენებთ SQLiteDatabaseHelper ტიპის ობიექტს. სრულად მოვიყვანოთ adapter კლასი რომელიც ხსნის და ხურავს კავშირს მონაცემთა ბაზასთან:

```
import android.content.Context;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class TsuDbAdapter {

    private SQLiteDatabase database = null;
    private TsuDbDatabaseHelper databaseHelper;
    private static final String DB_NAME = "tsudb";

    private static final int DB_VERSION = 1;
    private static final String CREATE = "create table faculties(Id INTEGER PRIMARY KEY ,NAME 
TEXT, CODE TEXT)";
    private Context ctx;

    public TsuDbAdapter(Context ctx) {
        this.ctx = ctx;
        databaseHelper = new TsuDbDatabaseHelper(this.ctx);
    }

    private static class TsuDbDatabaseHelper extends SQLiteOpenHelper {

        public TsuDbDatabaseHelper(Context context) {
            super(context, DB_NAME, null, DB_VERSION);
            // TODO Auto-generated constructor stub
        }

        @Override
        public void onCreate(SQLiteDatabase db) {
            try {
                db.execSQL(CREATE);
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
```

```
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
              db.execSQL("DROP TABLE IF EXISTS Faculties");
              onCreate(db);
        }

    }

    public SQLiteDatabase open() {

        if (database != null && database.isOpen())
              return database;
        else {
            database = databaseHelper.getReadableDatabase();
        }
        return database;
    }

    public void close() {

        if (database != null && database.isOpen()) {
        database.close();
                          }

    }
}
```

ჯერ განვიხილოთ open() მეთოდი. ის ამოწმებს არის თუ არა შექმნილი SQLiteDatabase ტიპის ობიექტი და არის თუ არა ღია, წინააღმდეგ შემთხვევაში ვახდენთ database ობიექტის ინიციალიზაციას. რაც შეეხება დახურვას იგი ხორციელდება უშუალოდ SQLiteDatabase ობიექტის close() მეთოდის გამოძახებით. ასევე გვაქვს ჩადგმული კლასი რომელიც არის SQLiteDatabaseHelper კლასის მემკვიდრე და რომლის საშუალებითაც ხდება SQLiteDatabase ობიექტის შექმნა. მას აქვს ორი მეთოდი: onCreate()და onUpgrade() რომელიც იძახება ბაზის განახლების დროს. საინტერესოა რა დროს შეიძლება დაგვჭირდეს ბაზის განახლება? ხშირად აპლიკაციის წერის პროცესში ან როცა უკვე დაწერილია აპლიკაცია, იცვლება მოთხოვნები რაც აისახება მონაცემთა ბაზაზე, სტრუქტურაზე და ა.შ აქედან გამომდინარე რომ მოვახდინოთ ბაზის ავტომატური განახლების ლოგიკის აღწერა ამისათვის გვაქვს მეთოდი onUpgrade() რომელიც ჩვენს შემთხვევაში იძახებს onCreate() ს რათა ყოველ ჯერზე შექმნას თავიდან ფაკულტეტების ცხრილი (ეს არის მოყვანილი უბრალოდ სამაგალითოდ, რეალურ აპლიკაციებში ყოველ ჯერზე ცხრილის შექმნა რასაკვირველია არ ხდება).

ახლა კი მოვიყვანოთ კლასის მაგალითი რომელიც მიმართავს ზემოთ განხილულ კლასს და მოაქვს მონაცემთა ცხრილ Faculty დან მონაცემები და აბრუნებს Faculty ტიპის ობიექტს, რომელიც შეიძლება შემდეგ გამოვიყენოთ სიაში საჩვენებლად:

```java
import ge.tsu.edu.entities.Faculty;

import java.util.ArrayList;
import java.util.List;

import android.database.Cursor;
import android.util.Log;

public class FacultyDao {
	private TsuDbAdapter dbAdapter;
	private static final String TABLE_NAME = "faculties";
	private static final String ID = "id";
	private static final String CODE = "code";
	private static final String NAME = "name";

	public FacultyDao(TsuDbAdapter dbAdapter) {
		this.dbAdapter = dbAdapter;
	}

	public List<Faculty> getAllFaculties() {
		List<Faculty> faculties = new ArrayList<Faculty>();
		Cursor cursor = dbAdapter.open().query(TABLE_NAME, new String[] { ID, NAME, CODE },
null, null, null, null, null);

		while (cursor.moveToNext()) {

			int id = cursor.getInt(cursor.getColumnIndex("Id"));
			String code = cursor.getString(cursor.getColumnIndex("Code"));
			String name = cursor.getString(cursor.getColumnIndex("Name"));
			Faculty faculty = new Faculty(id, code, name);
			faculties.add(faculty);

		}

		dbAdapter.close();
		return faculties;
	}
}
```

როგორც ვხედავთ მეთოდი getAllFaculties() იძახებს open() მეთოდს რომელიც აბრუნებს SQLiteDatabase ობიექტს, ხოლო შემდეგ იმავე ხაზზე იძახებს query() მეთოდს (ეს მეთოდი მრავალპარამეტრიანია, ყოველ ჯერზე შეგვიძლია გამოვიყენოთ ის რომელიც გვჭირდება) რომელსაც ყველანაირი ფილტრის გარეშე მოაქვს მონაცემები Faculties ცხრილიდან და შემდეგ უკვე ხდება ამ მონაცემების დამახსოვრება ArrayList ობიექტში.

რა თქმა უნდა SQLiteDatabase ობიექტს გააჩნია სხვა მეთოდებიც, მაგალითად update() განახლებისათვის, delete() წაშლისთვის და ა.შ.

6.2.1 CRUD ის რეალიზაცია

CRUD ი განისაზღვრება როგორც შექმნა, განახლება , წაშლის ერთობლიობა. მოვიყვანოთ CRUD ის რეალიზაცია ანდროიდის მონაცემთა ბაზაზე (CRUD – Create შექმნა, Update განახლება, Delete წაშლა). როგორც ზემოთ აღვნიშნეთ აუცილებელია ე.წ Adapter კლასი რომელიც უზრუნველჰყოფს კავშირს მონაცემთა ბაზასთან, ასევე კავშირის დახურვასაც და ა.შ. ახლა კი გადავწეროთ ზემოთ მოყვანილი TsuDbAdapter კლასი CRUDAdapter კლასად სადაც გავაერთიანებთ სხვადასხვა ოპერაციებს. მას ექნება დაახლოებით ასეთი სახე:

```
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class CRUDDbAdapter {
	private SQLiteDatabase db;
	private CRUDDbOpenHelper dbOpenHelper;
	private static final String CREATE = "create table Student(id integer primary key 
autoincrement,name text, email text)";
	private Context ctx;

	public CRUDDbAdapter(Context ctx) {
		this.ctx = ctx;
		dbOpenHelper = new CRUDDbOpenHelper(ctx, "CRUD", null, 1);
	}

	class CRUDDbOpenHelper extends SQLiteOpenHelper {

		public CRUDDbOpenHelper(Context context, String name, CursorFactory factory, int 
version) {
			super(context, name, factory, version);
		}

		@Override
		public void onCreate(SQLiteDatabase db) {
			db.execSQL(CREATE);

		}

		@Override
		public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
			db.execSQL("DROP IF EXISTS Student");
			onCreate(db);
		}

	}

	public SQLiteDatabase open() {
		Log.d(AppInfo.TAG, "Opening database");
		if (db == null)
			db = dbOpenHelper.getWritableDatabase();
		Log.d(AppInfo.TAG, "Database opened");
		return db;
```

```java
	}

	public void insert(Student student) {
		Log.d(AppInfo.TAG, "Inserting student " + student);
		ContentValues values = new ContentValues();
		values.put("name", student.getName());
		values.put("email", student.getEmail());
		long id = db.insert("Student", null, values);
		Log.d(AppInfo.TAG, "Inserted and returned id " + id);
	}

	public void delete(int id) {
		Log.d(AppInfo.TAG, "Deleting student with id " + id);
		db.delete("Student", "id=?", new String[] { id + "" });
	}

	public void update(Student student) {

		Log.d(AppInfo.TAG, "Updating student " + student);
		ContentValues values = new ContentValues();
		values.put("name", student.getName());
		values.put("email", student.getEmail());
		int rowsUpdated = db.update("Student", values, "id=? ", new String[] {
student.getId() + "" });
		Log.d(AppInfo.TAG, "Updated " + rowsUpdated + " rows");

	}

	public Student find(int id) {
		Student student = new Student();
		Log.d(AppInfo.TAG, "Searching student with id " + id);
		Cursor selectedStudent = db.query(true, "Student",
							new String[] { "id", "name", "email" },
							"id=" + id, null, null, null, null, null);
		if (selectedStudent.moveToFirst()) {
			student.setId(selectedStudent.getInt(selectedStudent.getColumnIndex("id")));

	student.setName(selectedStudent.getString(selectedStudent.getColumnIndex("name")));

	student.setEmail(selectedStudent.getString(selectedStudent.getColumnIndex("email")));
			Log.d(AppInfo.TAG, "Found student with id " + id);
		} else {
			Log.d(AppInfo.TAG, "Could not find student with id " + id);
		}
		return student;
	}

	public void close() {
		Log.d(AppInfo.TAG, "Closing database");
		if (db != null && db.isOpen()) {
			db.close();
			Log.d(AppInfo.TAG, "Database closed");
		}
	}
}
```

როგორც ვხედავთ გვაქვს ჩადგმული კლასი CRUDDbOpenHelper რომელიც წარმოადგენს SQLiteOpenHelper ის რეალიზაციას და ფარავს შესაბამის მეთოდებს (onCreate() შედის როცა საჭიროა ცხრილის შექმნა, ხოლო onUpdate() ს იყენებს როდესაც ხდება ბაზის სტრუქტურის ცვლილება და საჭიროა მისი მოდიფიცირება ან თავიდან შექმნა, მას აქვს პარამეტრები რომელიც ბაზას აიდენტიფიცირებს ვერსიებად და ამის მიხედვით პროგრამისტი წერს ლოგიკას თუ რა უნდა შეიცვალოს). ასევე გვაქვს შემდეგი მეთოდები:

- Insert() – პირობითად ეს მეთოდი გამოიყენება ახალი ობიექტის/სტრიქონის შესაბამის ცხრილში ჩაწერისათვის, ამ მეთოდში პირდაპირ ვიყენებთ SQLiteDatabase ის insert() მეთოდს, რომელსაც პირველ პარამეტრად გადაეცემა ცხრილის სახელი, მეორე პარამეტრად გადაეცემა ველის სახელი რომელიც გვინდა შევამოწმოთ ხომ არ არის null და მესამე პარამეტრად კი გადაეცემა ContentValues ობიექტი
- Delete() – პირობითად ეს მეთოდი გამოიყენება ობიექტის წაშლისათვის, მას პარამეტრად გადავცემთ ობიექტის იდენტიფიკატორს (გასაღებს) ხოლო ის თავის მხრივ იძახებს SQLiteDatabase ობიექტის delete მეთოდს, რომელსაც პირველ პარამეტრად გადაეცემა რომელი ცხრილიდან წაიშალოს , მეორე პარამეტრად გადაეცემა where პირობა, მესამე პარამეტრად გადაეცემა where პირობის შესაბამისი პარამეტრები მასივის სახით (`db.delete("Student", "id=?", new String[] { id + "" });` ნიშნავს რომ წაშალოს Student ცხრილიდან ის ჩანაწერი რომლის id იც არის გადმოცემული id)
- Update() - პირობითად ეს მეთოდი უზრუნველჰყობს სტრიქონის განახლებას, ბუნებრივია ამ მეთოდმა უნდა იცოდეს რომელი სტრიქონი განაახლოს, აქედან გამომდინარე SQLiteDatabase ს განსაზღვრული აქვს update მეთოდი შემდეგი პარამეტრებით:
    - რომელ ცხრილში ხდება განახლება
    - ახალი მნიშვნელობბები
    - რა პირობით (იგივე Where ის SQL ის)
    - პირობის პარამეტრები
- Find() – პირობითად ეს მეთოდი უზრუნველჰყობს სტრიქონის/ობიექტის მოძიებას გასაღების (ან შეიძლება რაიმე სხვა პირობის/პარამეტრის) მიხედვით. ჩვენს შემთხვევაში მას გადაეცემა id რომელის მიხედვითად ხორციელდება student ობიექტის ძებნა შესაბამის ცხრილში, ვიყენებთ მრავალპარამეტრიან SQLiteDatabase ის Query მეთოდს მონაცემების წამოსაღებად ხოლო შემდეგ ვიღებთ პირველივეს (რადგან find() დაწერილი გვაქვს მხოლოდ ერთი ჩანაწერის პოვნისათვის)

ძალიან მნიშვნელოვანი ობიექტებია : Cursor რომელსაც აბრუნებს Query მეთოდი, ხოლო მას კი განსაზღვრული აქვს იტერაციის მეთოდები თითოეულ სტრიქონზე, რა რაოდენობის ველებია ცხრილში, პირველი სტრიქონის ამოღება და ა.შ. ასევე ერთ ერთი მნიშვნელოვანი კლასი რასაც ვიყენებთ არის ContentValues , მას გააჩნია ე.წ გასაღები=მნიშვნელობა ტიპის მონაცემთა შენახვა. ანდროიდის ბაზის შემთხვევაში ეს გასაღები ყოველთვის უნდა ემთხვეოდეს ცხრილის ველს, ხოლო მნიშვნელობა შეიძლება იყოს ნებისმიერი. ამ კლასის გამოყენება კარგად ჩანს update მეთოდში.

SQLiteDatabase ს ასევე გააჩნია ტრანზაქციების მართვის შესაძლებლობაც რომელსაც ჩვენ არ განვიხილავთ.

მნიშვნელოვანი ასპექტები რაც მონაცემთა ბაზის შესახებ უნდა ვიცოდეთ არის შემდეგი:

1. სანამ რაიმე ოპერაციას გავაკეთებთ ბაზის რომელიმე ობიექტზე, აუცილებელია რომ წვდომა ამ ბაზაზე იყოს გახსნილი (გვქონდეს SQLiteDatabase ობიექტი როგორც writable an readable)
2. ოპერაციის/ოპერაციების ბოლოს აუცილებლად დავხუროთ წვდომა ბაზაზე რათა ამან არ გამოიწვიოს რესურსების დაკავება
3. ბაზა რომელსაც აპლიკაცია იყენებს წვდომადია მხოლოდ ამ აპლიკაციიდან, თუ გვინდა ერთი ბაზის მონაცემებზე სხვა აპლიკაციებმაც მიმართონ ამისათვის უნდა დავწეროთ ContentProvider ი რომელსაც მომდევნო თავში განვიხილავთ.

7.0 Content Provider ები

ხშირად საჭიროა რაიმე ინფორმაციის გაზიარება აპლიკაციებს შორის, წინა თავში ჩვენ განვიხილეთ ამის რამოდენიმე შესაძლებლობა თუმცა არც ერთი მათგანი არ უზრუნველჰყოფს პირდაპირ ინფორმაციის გაზიარებას სხვა აპლიკაციისათვის.

Content Provider ები შეიქმნა ზუსტად ამისათვის. მისი საშუალებით შესაძლებელია სხვა აპლიკაციას მიეცეს წვდომა მიმდინარე აპლიკაციის გარკვეულ რესურსებზე. სხვა სიტყვებით რომ ვთქვათ ის წარმოადგენს ერთ–ერთ Storage ს (მონაცემთა შენახვის ადგილს). მასზე მიმართვა ხდება გარკვეული სინტაქსით და გარკვეული უფლებებით. ძირითადად მისი გამოყენების ორი შესაძლებლობა არსებობს:

1. სტანდარტული Content Provider ები
2. ჩვენს მიერ დაწერილი Content Provider ები

როგორიც არ უნდა იყოს Content Provider ი , ის მაინც ხასიათდება შემდეგი პარამატრებით:

- Uri ობიექტი – წარმოადგენს ობიექტს რომელსაც გადაეცემა მიმართვის/წვდომის სტრიქონი
- მეთოდი query() – წარმოადგენს მეთოდს რომელიც აბრუნებს Cursor ობიექტს, დაბრუნებული ობიექტების იტერაციისათვის
- მეთოდი insert() – ამატებს ახალ ჩანაწერს და აბრუნებს შესაბამის uri ობიექტს
- მეთოდი delete() – შლის ჩანაწერებს და აბრუნებს წაშლილების რაოდენობას
- მეთოდი update() – ანახლებს ჩანაწერებს და აბრუნებს განახლებული სტრიქონების რაოდენობას

7.1 სტანდარტული Content Provider ების გამოყენება

სტანდარტულ Content Provider ებში შედის ისეთები რომელიც უკვე მოყვება Device ს, ეს შეიძლება იყოს მაგალითად , მობ.ტელეფონის შემთხვევაში კონტაქტების შესახებ ინფორმაციის ამოღება, ასევე ბრაუზერის შემთხვევაში Bookmark ების ამოღების შესაძლებლობა და ა.შ. მოვიყვანოთ ფრაგმენტი რომელიც Setting ების სტანდარტულ Content Provider ს მიმართავს :

```
Uri uri = Settings.CONTENT_URI;
Cursor cursor = getContentResolver().query(uri, null, null, null, null);
while(cursor.moveToNext()) {
Log.d("CP",cursor.getString(cursor.getColumnIndex(Settings.ACCOUNT_NAME)));
}
```

Uri ობიექტი გარდა ზემოთ აღწერილისა შეიძლება შეიქმნას როგორც Uri.parse(<parameter>) სადაც პარამეტრად გადაეცეს შემდეგი ტიპის სტრიქონი:

<standard_prefix>://<authority>/<data_path>/<id>

- <standard_prefix> წარმოადგენს ყოველთვის სიტყვა content ს
- <authority> წარმოადგენს რაიმე სახელს რომლის მიხედვითაც ხდება პროვაიდერის იდენტიფიცირება
- <data_path> – წარმოადგენს მონაცემთა სახელს
- <id> – თუ გვინა მონაცემთა იდენტიფიკაცია მაშინ დამატებით ვიყენებთ ამ ატრიბუტსაც

არის კიდევ ერთი მნიშვნელოვანი დეტალი რაც უკავშირდება უფლებებს. იმისათვის რომ გამოვიყენოთ სტანდარტული პროვაიდერები ხშირ შემთხვევაში საჭიროა გვქონდეს აპლიკაციისათვის მინიჭებული ე.წ კონტაქტების კითხვის უფლება. ამის გაკეთება ხდება AndroidManifest.xml ში როგორც ქვემოთ არის ნაჩვენები:

```
    </application>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
</manifest>
```

7.2 საკუთარი Content Provider ების შექმნა

იმისათვის რათა ჩვენი აპლიკაციის გარკვეული რესურსები გახსნილი იყოს სხვა აპლიკაციისათვის და ეს წვდომა ხდებოდეს Content Provider ის საშუალებით, აუცილებელია დავწეროთ ჩვენი პროვაიდერი ამისათვის კი აუცილებელია შემდეგი პირობების დაკმაყოფილება:

1. დავწეროთ კლასი რომელიც აუცილებლად იქნება android.content.ContentProvider აბსტრაქტული კლასის მემკვიდრე და გადაფარავს მის მეთოდებს: onCreate(), query(),insert(),update(), delete(), getType()
2. მოვახდინოთ ჩვენი დაწერილი პროვაიდერის რეგისტრაცია AndroidManifest.xml ში

ამის შემდეგ გამოძახება მოხდება ზუსტად იმავე გზით რა გზითად ვიძახებთ სტანდარტულ პროვაიდერებს , განსხვავება იქნება მხოლოდ Uri ობიექტის შიგთავსი. მოვიყვანოთ ჩვენს მიერ დაწერილი სამაგალითო პროვაიდერი :

```
import java.util.ArrayList;
import java.util.List;

import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.net.Uri;
import android.util.Log;

public class StudentInfoProvider extends ContentProvider {

	private List<StudentInfo> cachedStudents = null;

	public static final String STUDENT_INFO_PROVIDER_NAME = "ge.tsu.cpdemo.StudentInfo";
	public static final Uri CONTENT_URI = Uri.parse("content://" + STUDENT_INFO_PROVIDER_NAME +
"/studentsinfo");

	public static final int STUDENTS = 1;
	public static final int STUDENT_ID = 2;

	public static UriMatcher URI_MATCHER;
	static {
		URI_MATCHER = new UriMatcher(UriMatcher.NO_MATCH);
```

```
        URI_MATCHER.addURI(STUDENT_INFO_PROVIDER_NAME, "studentsinfo", STUDENTS);
        URI_MATCHER.addURI(STUDENT_INFO_PROVIDER_NAME, "studentsinfo/#", STUDENT_ID);
    }

    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        Log.d(AppInfo.TAG_NAME, "Deleting .....");
        int count = 0;
        switch (URI_MATCHER.match(uri)) {
        case STUDENT_ID: {
            for (int i=0;i<cachedStudents.size();i++) {
                if (cachedStudents.get(i).getId() == Integer.parseInt(selectionArgs[0]))
{
                    cachedStudents.remove(cachedStudents.get(i));
                    count++;
                }
            }
            break;
        }
        }
        getContext().getContentResolver().notifyChange(uri, null);
        Log.d(AppInfo.TAG_NAME, "Deleted " + count);
        return count;
    }

    @Override
    public String getType(Uri uri) {
        Log.d(AppInfo.TAG_NAME, "Get Type .....");
        String type = "";
        switch (URI_MATCHER.match(uri)) {
        case STUDENTS: {
            type = "vnd.android.cursor.dir/ge.tsu.cpdemo.StudentsInfo";
            break;
        }
        case STUDENT_ID: {
            type = "vnd.android.cursor.item/ge.tsu.cpdemo.StudentsInfo";
            break;
        }
        }
        return type;
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        Log.d(AppInfo.TAG_NAME, "Inserting .....");
        Uri updatedUri = null;
        StudentInfo info = new StudentInfo();
        info.setId(22);
        info.setFullName(values.getAsString("fullName"));
        info.setAge(values.getAsInteger("age"));

        cachedStudents.add(info);
        updatedUri = ContentUris.withAppendedId(CONTENT_URI, info.getId());
        getContext().getContentResolver().notifyChange(updatedUri, null);
        Log.d(AppInfo.TAG_NAME, "Inserted " + info);
        return updatedUri;
    }

    @Override
```

```
	public boolean onCreate() {
		Log.d(AppInfo.TAG_NAME, "OnCreate() .....");
		cachedStudents = new ArrayList<StudentInfo>();
		for (int i = 0; i < 100; i++) {
			cachedStudents.add(new StudentInfo(i, "fullName of " + i, i % 20));
		}
		Log.d(AppInfo.TAG_NAME, "Students cached");
		return true;
	}

	@Override
	public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
String sortOrder) {
		Log.d(AppInfo.TAG_NAME, "Query .....");
		// TODO Auto-generated method stub
		return null;
	}

	@Override
	public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)
{
		Log.d(AppInfo.TAG_NAME, "Updating .....");
		// TODO Auto-generated method stub
		return 0;
	}
```

ჯერ განვიხილოთ შემდეგი ნაწილი:

```
private List<StudentInfo> cachedStudents = null;

	public static final String STUDENT_INFO_PROVIDER_NAME = "ge.tsu.cpdemo.StudentInfo";
	public static final Uri CONTENT_URI = Uri.parse("content://" + STUDENT_INFO_PROVIDER_NAME +
"/studentsinfo");

	public static final int STUDENTS = 1;
	public static final int STUDENT_ID = 2;

	public static UriMatcher URI_MATCHER;
	static {
		URI_MATCHER = new UriMatcher(UriMatcher.NO_MATCH);
		URI_MATCHER.addURI(STUDENT_INFO_PROVIDER_NAME, "studentsinfo", STUDENTS);
		URI_MATCHER.addURI(STUDENT_INFO_PROVIDER_NAME, "studentsinfo/#", STUDENT_ID);
	}
```

გვაქვს cachedStudents ცვლადი რომელშიც პირობითად ვინახავთ StudentInfo ტიპის ობიექტებს წინასწარ რათა შემდეგ როდესაც მოხდება რაიმე მეთოდის გამოძახება , მოხდეს ამ შევსებულ სტუდენტებზე იტერირება.

თითოეულ პროვაიდერს აუცილებლად უნდა გააჩნდეს სახელი, იგივე authority რომლითაც დარეგისტრირდება AndroidManifest.xml ფაილში. ამ სახელისათვის შემოვიღეთ ცვლადი STUDENT_INFO_PROVIDER_NAME, ხოლო შემდეგი ცვლადი არის Uri ტიპის ობიექტი სადაც სპეციალური სტრუქტურით ვქმნით და ვინახავთ მას. ყურადღება მივაქციოთ data_path ს რომელშიც წერია studentsinfo იგი განსაზღვრავს მონაცემთა ტიპის სახელს, რადგან ჩვენ აქცენტს ვაკეთებთ StudentInfo ტიპის ობიექტებზე ამიტომ ეს სახელიც ლოგიკურად შევარჩიეთ. საინტერესოა შემდეგი ორი რიცხვითი ცვლადი: STUDENTS=1 და STUDENT_ID=2 , ამ ცვლადების საშუალებით შემდეგ მეთოდებში გვექნება შესაძლებლობა

მოვახდინოთ Uri ობიექტის იდენტიფიცირება, რაც იმას ნიშნავს რომ ზუსტად გავიგებთ მომთხოვნს მთელი მონაცემები ჭირდება თუ id პარამეტრის მიხედვით რაიმე ერთი ან რამოდენიმე მონაცემი. ამ ორი შესაძლებლობისათვის მომზადება კარგად ჩანს სტატიკური ინიციალიზაციის ბლოკში.

ახლა კი განვიხილოთ გადაფარული მეთოდები თუ რა ლოგიკით უნდა ავამუშაოთ ისინი. პირველ რიგში აუცილებელია გადავფაროთ onCreate() რომელიც რაიმე ლოგიკით აბრუნებს true ან false ს. ჩვენს შემთხვევაში ვახდენთ StudentInfo ობიექტებით ბმული სიის შევსებას და ვაბრუნებთ true ს, თუკი რაიმე ისე არ წავიდა შეგვიძლია დავაბრუნებინოთ false .

დანარჩენი მეთოდებიდან შეგვიძლია გამოვყოთ:

- getType(Uri uri) – მისი საშუალებით ვახდენთ იდენტიფიკაციას ერთი რაიმე მონაცემი ჭირდება გამომძახებელს თუ მთელი Cursor რომელზე იტერირებასაც შემდეგ შეუდგება. იგი უნდა აბრუნებდებს ე.წ MIME Type ს.
- დანარჩენი მეთოდები მუშაობს დაახლოებით ისეთნაირად როგორც ეს იყო აღწერილი SQLite ის შემთხვევაში, მცირე მოდიფიკაციებით როგორც ეს ზემოთ ნაჩვენებ კოდში კარგად ჩანს.

გარკვეულ მეთოდებში იდენტიფიკაცია კეთდება ე.წ URI_MATHCER.match(uri) მეთოდით, ის აბრუნებს STUDENT ან STUDENT_ID ის იმის მიხედვით ერთ მონაცემზე წვდომა ხორციელდება გამომძახებლის მხრიდან თუ მრავალზე.

როდესაც ხდება შემდეგი მეთოდების გადაფარვა : insert(),updaye() და delete() აუცილებელია მეთოდის ბოლოს გამოვიძახოთ შემდეგი მეთოდი:

```
getContext().getContentResolver().notifyChange(updatedUri, null);
```

რათა მოხდეს ცვლილებების დამახსოვრება.

ასევე უნდა მოვახდინოთ ჩვენს მიერ დაწერილი პროვაიდერის რეგისტრაცია AndroidManifest.xml ში, ეს ჩვენს შემთხვევაში მოხდება შემდეგნაირად:

```
    </activity>

  <provider android:name="StudentInfoProvider" android:authorities="ge.tsu.cpdemo.StudentInfo"/>

  </application>
```

8.0 Messaging და Networking

ანდროიდის პლატფორმის შემთხვევაში, კომუნიკაციის თვალსაზრისით გვაქვს შემდეგი ტიპის კომპონენტები:

- Messaging
- Networking

პირველი განიხილავს შესაძლებლობას გავაგზავნოთ და მივიღოთ SMS შეტყობინება ასევე Email, ხოლო მეორეში შეგვიძლია გავაერთიანოთ ვებ თან კავშირი HTTP პროტოკოლის საშუალებით, ასევე ისეთი მნიშვნელოვანი საკითხები როგორიცაა ვებ სერვერებთან წვდომა, ვებ სერვისებთან წვდომა და მათი გამოძახების შესაძლებლობა. ზემოთ აღნიშნული შესაძლებლობები საკმაოდ წარმატებულს ხდის ანდროიდის აპლიკაციას, რადგან ამ ორ შესაძლებლობას გააჩნია საკმაოდ ფართო გამოყენება რეალური აპლიკაციების შემთხვევაში. თავიდან ბოლომდე აქ არ განვიხილავთ Messaging and Networking ის API ს, რადგან ის საკმაოდ ვრცელი თემაა, უბრალოდ მოვიყვანთ იმ ძირითადი შესაძლებლობების რეალიზებას რაც ყველა ანდროიდ დეველოპერმა უნდა იცოდეს.

8.1 Messaging

Messaging ის შემთხვევაში გვაქვს ძირითადად ორი ფუნქციონალი, რომელთაგან პირველია შეტყობინების გაგზავნა, ხოლო მეორე შეტყობინების მიღება. რა თქმა უნდა აქ საუბარი შესაძლებელია შეეხოს მესიჯების ავტომატური გაგზავნების შესაძლებლობას, მიღებაზე მოსმენის შესაძლებლობას და ა.შ რაც საკმაოდ ფართო თემაა და რასაც აქ არ განვიხილავთ. ახლა გადავიდეთ ძირითად საკითხებზე რაც არის საჭირო იმისათვის , რომ გავაგზავნოთ და მივიღოთ მესიჯი. პირველ რიგში მოვიყვანოთ გაგზავნის მაგალითი, მანამდე კი ჩავთვალოთ რომ გვაქვს აპლიკაცია რომელსაც აქვს ღილაკი „გაგზავნა“ რომელზე დაჭერის შემთხვევაში სისტემამ უნდა გააგზავნოს მითითებულ ნომერზე მესიჯი. უბრალოდ მესიჯის გასაგზავნად გვჭირდება შემდეგი კლასები და პროგრამული რეალიზება:

```
SmsManager smsManager = SmsManager.getDefault();
smsManager.sendTextMessage(phoneNumber, null, message, null, null);
```

პირველ რიგში ვიღებთ SmsManager კლასის ობიექტს ისე როგორც ეს პირველ ხაზზეა ნაჩვენები (და არა პირდაპირ new თი). ხოლო SmsManager კლასს აქვს მეთოდი sendTextMessage() რომელსაც სხვა პარამეტრებთან ერთად გადაეცემა ტელეფონის ნომერი და მესიჯი რაც უნდა გაიგზავნოს.ზემოთ ნაჩვენები ამ ორხაზიანია კოდის შესრულების შემდეგ სისტემა შეეცდება მესიჯის გაგზავნას. ახლა მოვიყვანოთ გაგზავნის მეთოდის პარამეტრები:

1. PhoneNumber - ტელეფონის ნომერი, რომელზეც ვაგზავნით შეტყობინებას
2. ScAddress – სერვის ცენტრის მისამართი
3. Message – მესიჯის ტანი (თავად მესიჯის ტექსტი)
4. SentPendingIntent – PendingIntent ობიექტი რომელიც აფიქსირებს გაიგზავნა თუ არა ან რა სტატუსით გაიგზავნა მესიჯი
5. DeliveredPendingIntent – PendingIntent ობიექტი რომელიც აფიქსირებს მივიდა თუ არა მესიჯი ან რა სტატუსი დააფორმირა „მისვლის“ სერვისმა

საინტერესოა შემდეგი საკითხებზე პასუხის გაცემა: გაიგზავნა თუ არა საერთოდ მესიჯი? თუ არა რატომ? მიიღო თუ არა ადრესატმა მესიჯი? თუ არა რატომ?

ამ კითხვებზე პასუხი რომ გავცეთ უნდა შევქმნათ ორი PendingIntent ობიექტი რომელიც ბოლო ორ პარამეტრად გადაეცემა გაგზავნის მეთოდს. ასევე აუცილებელია დავარეგისტრიროთ გაგზავნის და მიღების ხდომილებები. საბოლოო ჯამში გაგზავნის კოდი მიიღებს შემდეგ სახეს:

```
private final String SMS_SENT = "SMS_SENT";
private final String SMS_DELIVERED = "SMS_DELIVERED";

private void sendSMS(String phoneNumber, String message) {

    PendingIntent sentPI = PendingIntent.getBroadcast(getBaseContext(), 0, new Intent(SMS_SENT), 0);
    PendingIntent deliveredPI = PendingIntent.getBroadcast(getBaseContext(), 0, new Intent(SMS_DELIVERED), 0);
    registerReceiver(new SentSMSBroadCastReceiver(), new IntentFilter(SMS_SENT));
    registerReceiver(new DeleiveredSMSBroadCastReceiver(), new IntentFilter(SMS_DELIVERED));

    SmsManager smsManager = SmsManager.getDefault();
    smsManager.sendTextMessage(phoneNumber, null, message, sentPI, deliveredPI);
}
```

როგორც ვხედავთ გვაქვს ორი კონსტანტური სტრინგი SMS_SENT და SMS_DELIVERED , შემდეგ მეთოდში sendSMS() იქმნება PendingIntent ობიექტები, ეს ობიექტები საჭიროა გაგზავნის და მიღების ხდომილებების მოსასმენად. შემდეგ ორ ხაზზე ვარეგისტრირებთ ორ მიმღებს: პირველს გაგზავნილი მესიჯის სტატუსებისათვის (იმისათვის, რომ გავიგოთ რა მოუვიდა მესიჯს , გაიგზავნა თუ არა საბოლოო ჯამში), მეორეს უკვე გაგზავნილი მესიჯის სტატუსისათვის (იმისათვის, რომ გავიგოთ რა მოუვიდა გაგზავნილ მესიჯს, მიუვიდა თუ არა ადრესატს). მიმღებების რეგისტრაციას გააჩნია ორი პარამეტრი, პირველი არის კლასი რომელიც რეალიზაციას უკეთებს აბსტრაქტულ BroadcastReceiver ს, ხოლო მეორე წარმოადგენს IntentFilter ს რომლითაც ხდება შესაბამისი ინტენტ ობიექტის გადაბმა. BroadcastReceiver აბსტრაქტულ კლასს გააჩნია ერთი აბსტრაქტული მეთოდი onReceive() რომელსაც პირველ პარამეტრად გადაეცემა Context ობიექტი (რომელიც არის Activity კლასის წინაპარი კლასი) , ხოლო მეორე პარამეტრად Intent ობიექტი. საბოლოოდ ჩვენს მიერ დაწერილი SentSMSBroadCastReceiver კლასი ასეთნაირად გამოიყურება:

```java
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.telephony.SmsManager;
import android.widget.Toast;

public class SentSMSBroadCastReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        switch (getResultCode()) {
        case Activity.RESULT_OK: {
            Toast.makeText(context, "SentSMS : OK", Toast.LENGTH_SHORT).show();
            break;
        }
        case Activity.RESULT_CANCELED: {
            Toast.makeText(context, "SentSMS : Canceled", Toast.LENGTH_SHORT).show();
            break;
        }
        case SmsManager.RESULT_ERROR_NO_SERVICE: {
            Toast.makeText(context, "SentSMS : No service", Toast.LENGTH_SHORT).show();
            break;
        }
        }

    }

}
```

ხოლო DeliveredSMSBroadCastReceiver კი შემდეგნაირად:

```java
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.telephony.SmsManager;
import android.widget.Toast;

public class DeleiveredSMSBroadCastReceiver extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {

        switch (getResultCode()) {
        case Activity.RESULT_OK: {
            Toast.makeText(context, "DeliveredSMS: OK", Toast.LENGTH_SHORT).show();
            break;
        }
        case SmsManager.RESULT_ERROR_GENERIC_FAILURE: {
            Toast.makeText(context, "DeliveredSMS: Generic failure", Toast.LENGTH_SHORT).show();
            break;
        }
        case SmsManager.RESULT_ERROR_NO_SERVICE: {
            Toast.makeText(context, "DeliveredSMS: No service", Toast.LENGTH_SHORT).show();
            break;
        }
        case SmsManager.RESULT_ERROR_NULL_PDU: {
            Toast.makeText(context, "DeliveredSMS: Null PDU", Toast.LENGTH_SHORT).show();
            break;
        }
        case SmsManager.RESULT_ERROR_RADIO_OFF: {
            Toast.makeText(context, "DeliveredSMS: Radio off", Toast.LENGTH_SHORT).show();
            break;
        }
        }
    }
}
```

როგორც ვხედავთ ორივე კლასის შემთხვევაში ვფარავთ onReceive() მეთოდს და ვაკეთებთ სტატუსის იდენტიფიკაციას მეთოდით getResultCode(). ძირითადი სტატუსები რაც გვხვდება არის შემდეგი:

- Activity.RESULT_OK - ბრუნდება მაშინ როდესაც გაგზავნა/გაგზავნილის მიღება წარმატებულია
- Activity.RESULT_CANCELED – ბრუნდება მაშინ როდესაც ვერ მოხდა მესიჯის გაგზავნა
- SmsManager.RESULT_ERROR_NO_SERVICE – ბრუნდება მაშინ როდესაც დევაისზე არ არის შესაძლებლობა მესიჯების გაგზავნის, ამის კლასიკური მაგალითია tab რომელზეც აყენია ანდროიდი თუმცა ტელეფონისაგან განსხვავებით მისგან მესიჯის გაგზავნა არ ხდება
- SmsManager.RESULT_ERROR_GENERIC_FAILURE – ბრუნდება მაშინ თუ გაგზავნილი მესიჯის დადასტურება რაიმე მიზეზით ვერ ხერხდება
- SmsManager.RESULT_ERROR_NULL_PDU – (Protocol Description Unit წარმოადგენს მესიჯის ფორმატს) ბრუნდება როდესაც ამ პროტოკოლის შესაბამისად არ არის რიგზე რაიმე
- SmsManager.RESULT_ERROR_RADIO_OFF – ბრუნდება მაშინ როდესაც მიღების ფუნქცია არის გათიშული

არსებობს მეორე გზაც მესიჯის გაგზავნის, ეს გზა გამოიხატება შესაბამისი Intent ობიექტის შექმნით და შესაბამისი Activity ს გამოძახებით. მოვიყვანოთ მაგალითი თუ როგორ ხდება ეს:

```
private void sendSMSIntents(String phoneNumber, String message) {
    Intent smsIntent = new Intent(android.content.Intent.ACTION_VIEW);
    smsIntent.putExtra("address", phoneNumber);
    smsIntent.putExtra("sms_body", message);
    smsIntent.setType("vnd.android-dir/mms-sms");
    startActivity(smsIntent);
}
```

პირველად ვქმნით ჩვეულებრივ Intent ობიექტს, შემდეგ ვდებთ მასში გარკვეულ პარამეტრებს ზუსტად იმ კონსტანტური სახელით რომელიც აღწერილია, ვაყენებთ ე.წ MIME-TYPE ს, და ვიძახებთ მეთოდ startActivity() ს. განსხვავება წინა კოდსა და ამ კოდს შორის არის ის რომ ამ შემთხვევაშ გაიხსნება მესიჯის გაგზავნის აპლიკაცია, ხოლო წინა შემთხვევაში მოხდება პირდაპირ გაგზავნა.

Messaging ს ასევე მიეკუთვნება ემაილის გაგზავნაც , რომლის შესაბამისი კოდი ასე გამოიყურება:

```
private void sendEmail(String email, String message) {
    Intent emailIntent = new Intent(Intent.ACTION_SEND);
    emailIntent.setData(Uri.parse("mailto:"));
    emailIntent.putExtra(Intent.EXTRA_EMAIL, new String[]{email});
    emailIntent.putExtra(Intent.EXTRA_CC, new String[]{"kk@gmail.com"});
    emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Android");
    emailIntent.putExtra(Intent.EXTRA_TEXT, message);
    emailIntent.setType("message/rfc822");
    startActivity(Intent.createChooser(emailIntent, "Email"));
}
```

ეს ყოველივე ხდება Intent ის საშუალებით რომელსაც ვქმნით პირველად (ACTION_SEND კონსტანტით). შემდეგ ვახდენთ setData() მეთოდის გამოძახებას რომელიც მასში გადაცემული Uri ობიექტით ახდენს იდენტიფიკაციას თუ რა უნდა მოხდეს, ჩვენს შემთხვევაში ეს იქნება მეილის გაგზავნა. შემდეგ ვახდენთ მეილისათვის საჭირო პარამეტრების დაყენებას, ასევე ტიპის მითითებას თუ რა სტანდარტით გაიგზავნოს

და ბოლოს ვიძახებთ startActivity მეთოდს რომელიც გვთავაზობს ემაილების ამორჩევას. ეს ამორჩევა შეიძლება მოხდეს დევაისის სტანდარტულ მეილ კლიენტსა და უშუალოდ Account email ს შორის.

შენიშვნა: იმისათვის რათა SMS გაგზავნა მიღებამ გამართულად იმუშაოს აუცილებალია შესაბამისი უფლებების გაწერა AndroidManifest.xml ფაილში <Application/> ტეგის მაღლა , ეს უფლებები არის შემდეგი:

```
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
```

8.2 Networking

Networking ის ერთ–ერთი მთავარი თემაა ვებ რესურსებზე წვდომა, ამ ინფორმაციის წამოღება, კლიენტის მხარეს გაშიფრვა და გარკვეული სახით ჩვენება. გარდა ამისა კიდევ არის ერთ–ერთი უმნიშვნელოვანესი თემა რომელიც ეხება ვებ სერვისებს და რომელსაც აქ არ განვიხილავთ, თუმცა ის შეიძლება ითქვას , რომ იმისათვის რათა მოვახდინოთ საჭირო ვებ სერვისზე მიმართვა აუცილებელია გვესმოდეს ის ლოგიკა რასაც ქვემოთ მოვიყვანთ სხვა მაგალითზე.

დაუშვათ გვინდა განვახორციელოთ წვდომა ინტერნეტ რესურსზე და ჩამოვტვირთოთ იქიდან რაიმე ფაილი, მაგალითად რაიმე სურათი რომელსაც გამოვიყენებთ აპლიკაციაში. პირველ რიგში არ უნდა დაგვავიწყდეს შესაბამისი უფლების გაწერა AndroidManifest.xml ფაილში რომელიც არის შემდეგი:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

ამის შემდეგ უკვე უნდა განვახორციელოთ უშუალოდ მიმართვა ინტერნეტ რესურსებზე და წავიკოთხოთ შესაბამისი შიგთავსი. ეს ყოველივე განვიხილოთ შემდეგ მარტივ მაგალითზე: დაუშვათ გვაქვს ღილაკი რომელზე დაჭერის დროსაც აპლიკაციამ უნდა მიმართოს წინასწარ განსაზღვრულ მისამართს , ჩამოტვირთოს სურათი და ჩასვას ჩვენს Activity ში. ამ ყოველივეს რეალიზაცია გამოიყურება შემდეგნაირად:

```
private void downloadImage(String url) {
    AsyncTask<String, Void, Bitmap> task = new AsyncTask<String, Void, Bitmap>() {
        @Override
        protected Bitmap doInBackground(String... params) {
            Bitmap bitmap = null;
            try {
                for (int i = 0; i < params.length; i++) {
                    java.net.URL urlToConnect = new java.net.URL(params[i]);
                    URLConnection connection = urlToConnect.openConnection();
                    if (connection instanceof HttpURLConnection) {
                        HttpURLConnection httpURLConnection = (HttpURLConnection) connection;
                        httpURLConnection.setAllowUserInteraction(false);
                        httpURLConnection.setInstanceFollowRedirects(true);
                        httpURLConnection.setRequestMethod("GET");
                        httpURLConnection.connect();

                        int responce = httpURLConnection.getResponseCode();
                        if (responce == HttpURLConnection.HTTP_OK) {
                            InputStream is = httpURLConnection.getInputStream();
                            bitmap = BitmapFactory.decodeStream(is);
                            is.close();
                        }
                    }
                }
            } catch (Exception ex) {
                ex.printStackTrace();
            }
            return bitmap;
        }
    }; try {
        task.execute(url);
        Bitmap image = task.get();
        ImageView imageView = (ImageView) findViewById(R.id.some_image);
        imageView.setImageBitmap(image);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

მთავარი მეთოდი ჩვენს შემთხვევაში არის downloadImage რომელსაც პარამეტრად გადაეცემა სურათის მისამართი. შემდეგ გვაქვს AsyncTask<Param,Progress,Result> ობიექტი, რომელიც საჭიროა იმისათვის რათა გადმოწერის პროცესი აუცილებლად გაეშვას ახალ Thread ში, (ახალ ნაკადში და არა მთავარ UI ნაკადში). მას აქვს მეთოდი doInBackground() რომელიც მთავარი ნაკადის პარალელურად მუშაობს და ასრულებს იმ ლოგიკას რაც მეთოდის ტანში წერია. შემდეგ ვახდენთ Url ობიექტის შექმნას და კავშირის ამოღებას რომელიც უნდა დაემთხვეს HttpUrlConnection კლასს. ამ კლასის ობიექტისათის ვახდენთ შემდეგი პარამეტრების დაყენებას: ქონდეს თუ არა მომხმარებელს მასვე უშუალოდ წვდომის საშუალება (false), თუ ლინკს ჭირდება გადამისამართება, გადამისამართების უფლება მივცეთ თუ არა (true) და რომელი HTTP მეთოდით მოვახდინოთ ინფორმაციის წამოღება (GET). ამის შემდეგ ვახდენთ დაკავშირებას და თუ სტატუსი დაბრუნდა HTTP_OK (რომელიც არის იგივე სტატუსი 200) შემდეგ უკვე ვახდენთ სურათის ამოღებას და Bitmap ობიექტის დაბრუნებას. აღნიშნული asynctask ის გაშვება ხდება execute მეთოდით რომელიც სათავეს უდებს ყოველივე ზემოთ აღწერილს, ხოლო შემდეგ მეთოდი get() ელოდება AsyncTask ის შესაბამის ნაკადს, სანამ არ დაასრულებს მუშაობას და აბრუნებს წაკითხულ სურათს ჩვენს შემთხვევაში. ხოლო ამ წაკითხული სურათის დაყენება ხდება ImageView ზე.

აქ მნიშვნელოვანია 2 რამ:

1. **არ შეიძლება კავშირის Thread/ნაკადი გაეშვას ძირითად ნაკადთან ერთად, სწორედ ამიტომ შემოვიღეთ AsyncTask**
2. **არ შეიძლება AsyncTask ნაკადში ინტერფეისის ობიექტებზე მანიპულირება, სწორედ ამიტომ ვიძახებთ get() მეთოდს რომელიც გვიბრუნებს წაკითხულ სურათს რათა ის ჩავსვათ განსაზღვრულ ადგილას.**