INTRODUCTION TO ANDROID

Niv Voskoboynik
Ben-Gurion University
Electrical and Computer Engineering
Advanced computer lab
2015

Contents

• Introduction
• Prior learning
• Download and install Android Studio
• How to create a new project
• Create “Hello World” Application
• Create the User Interface
• View
• MediaPlayer
• Touch Event

• Android manifest
• Activities
• Thread
• LogCat
• Toast Notifications
• Summary
Introduction - What is operating system?

- An operating system is a software bridging between the **hardware**, the **user** and the **software** applications.
- Operating system provides three interfaces:
  - **UI** (User Interface).
  - **Hardware interface** - drivers.
  - **API** (Applications programming interface).
Introduction - What is operating system?

- User (printf)
- System call (standard C library - stdio)
- Kernel Write
- Hardware (UART)

Operating system - Services

- Process Management
- Security
- Networking
- Memory Allocation
- Operating System
- I/O
- Drivers
- Hardware
- API
- Application
Operating system - Android

- Android is a powerful **Linux** based operating system supporting a large number of applications.
- Android applications are written in **Java** programming language.
- Android is available as open source for developers to develop applications which can be further used for selling in Android market.
- There are around 1.6 million applications developed for Android with over 50 billion downloads.

Operating system - Android

- For software development, Android provides **SDK** (Software Development Kit).
- The Android **SDK** provides you the **API** libraries and developer tools necessary to build, test, and debug apps for Android.
- The latest and most advanced Android **IDE** (Integrated Development Environment) is called – Android Studio.
Prior learning

• Object-oriented programming
• You must understand terms like:
   Class
   Method
   Interface
   Instance
   Thread
   Extend/Implement
• Great tutorial can be found in:
   http://docs.oracle.com/javase/tutorial/

Prior learning - Java

```java
package tttemp;
public class Bicycle {
    private int speed;
    public Bicycle(){       speed = 20; }
    public Bicycle(int speed){ this.speed = speed; }
    public int getSpeed(){ return speed; }
}

package tttemp;
public class MountainBike extends Bicycle {
    public int seatHeight = 10;
    public MountainBike(int speed){ super(speed); }
    public int getHeight(){ return seatHeight; }
}
```

• **Permissions** are allowed at the beginning
• Classes are handled through a **pointer**
• Java doesn’t support **multiple** inheritance
• The **values** goes together with the declaration
• **super()** calls the parent constructor
Prior learning - Java

- The **main** is within a class
- Constructor is called with the use of **new**
- **Override** methods is allowed

```java
package ttemp;
public class CreateObject {
    public static void main(String[] args) {
        MountainBike A,B;
        A = new MountainBike(20);
        @Override
        public int getHeight(){ return seatHeight+10; }
        B = new MountainBike(20);
        System.out.println("My speed is "+A.getSpeed()+" and the seat height is "+A.getHeight());
        System.out.println("My speed is "+B.getSpeed()+" and the seat height is "+B.getHeight());
    }
}
```

Download and install Eclipse

- Download **JDK** (Java Development Kit) from:
    In order to be able to run Java code on your machine.
- Download and install **Eclipse and Android ADT** from:
    This download includes the **SDK** and the workspace.
How to create a new project

- Choose a name
  - Application name
  - Project name
  - Activity name
  - Layout name
- Minimum SDK choose
  - API 10
- Blank Activity

Create “Hello World” Application

- Before we can run the application, we need to setup an **AVD** (Android Virtual Device), or emulator, to run it on.
- In order to setup a new **AVD** go to:
  - AVD Manager
- We’re now ready to run our application.
- Right-click on your project in the Package Explorer and choose:
  - Run As -> Android Application.
Create the User Interface

- Android allows you to lay out your user interfaces using a simple XML specification. Begin by creating a new Android XML file:
  - Go to res->layout in the Package Explorer
  - In order to create a new XML file (for future use), Right click and choose new-> Android XML Layout File

![Graphical Layout](image)

Create the User Interface

- In the Graphical Layout tab you can use visual method to add components:
  - TextView - A TextView is simply a component that displays text.
  - EditText - A component to get text from the user.
  - Button
  - ImageView – A component to display pictures.

![Graphical Layout](image)
Create the User Interface – Set properties

- Here are a few useful examples: In order to use them go to the XML code tab of your layout and write them in the component description.
  - Stretch the component to fit a full screen
    ```xml
    android:layout_width="match_parent"
    ```
  - Paint the text
    ```xml
    android:textColor="@android:color/white"
    ```
  - Write text on the component
    ```xml
    android:text="text"
    ```
  - Change the size of the text
    ```xml
    android:textSize="30sp"
    ```
  - Change the style of the text
    ```xml
    android:textStyle="bold"
    ```

Create the User Interface – Example

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/main_layout"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
  <TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="38dp"
    android:text="Welcome to my app"
    android:textColor="#babafa"
    android:textSize="20sp" />
  <EditText
    android:id="@+id/etTextList2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView1"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="38dp"
    android:text="Please enter your name"
    android:hint="Please enter your name" />
  <Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="38dp"
    android:background="#F7F8FA"
    android:text="Start"
    android:textSize="20sp" />
</RelativeLayout>
```
View

• Public class added in API 1. This class represents the basic building block for UI components.
• A View occupies a rectangular area on the screen and is responsible for drawing and event handling.
• View is the base class for widgets, which are used to create interactive UI components (buttons, text fields, etc.).
• For example: after adding an EditText in the XML file, we need to add a new EditText instance in our java code in order to use it.

    EditText text = (EditText)findViewById(R.id.EditText);

View – Using Views

• Set properties: The available properties and the methods that set views will vary among the different subclasses of views. Note that properties that are known at build time can be set in the XML layout files.
• Set up listeners: Views allow clients to set listeners that will be notified when something interesting happens to the view. For example, a Button exposes a listener to notify clients when the button is clicked.
• Set visibility: You can hide or show views.
View – Add a new button

- First, go to the XML file you would like and add a button.

```xml
<Button
    android:id="@+id/my_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="my_button_text"/>
```

- Second, Go to src->….->MainActivity in the Package Explorer. Inside the Activity class and the method onCreate add

```java
Button button = (Button)findViewById(R.id.my_button);
```

- Our button instance in now identified with the Button component we created in the XML file.

View – Set up listener and visibility

- To set up a new listener inside the activity use

```java
Button button= (Button) findViewById(R.id.buttonId);
button.setOnClickListener(new View.OnClickListener()
{
    @Override public void onClick(View v) {
        // perform action on click
    }
});
```

- To set visibility you can use

```java
button.setVisibility(View.VISIBLE);
```

```java
or
```

```java
button.setVisibility(View.INVISIBLE);
```
View – Draw

- View also allows us to draw shapes on the screen.
  - For example: in order to draw rectangle we can use

```java
View view = new View(this)
    Paint paint = new Paint();
    @Override
    protected void onDraw(Canvas canvas){
        canvas.drawRect(30, 30, 80, 80, paint); 
    }
};
setContentView(view);
```

Note that we used Paint and Canvas classes.
- The Canvas class holds the "draw" calls.
- The Paint class holds the style and color information about how to draw e.g. geometries, text, and color.

View - Draw

- Few useful examples: you can use them inside the method onDraw.
  - Draw the specified rectangle using the specified paint.
    ```java
canvas.drawRect(left, top, right, bottom, paint)
```
  - Set the paint's color.
    ```java
    paint.setColor(Color.YELLOW);
    ```
  - Draw the specified circle at origin (x,y) using the specified paint.
    ```java
canvas.drawCircle(x, y, radius, paint);
```
  - Draw the text, with origin at (x,y), using the specified paint.
    ```java
canvas.drawText(text, x, y, paint);
```
  - Set the size of the text.
    ```java
    paint.setTextSize(size);
    ```
View – Draw Example

```java
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        View view = new View(this){
            Paint paint = new Paint();
            protected void onDraw (Canvas canvas){
                paint.setTextSize(30);
                paint.setColor(Color.BLUE);
                canvas.drawText("Hello World", 10, 50, paint);
                paint.setColor(Color.RED);
                canvas.drawCircle(100, 300, 50, paint);
            }
        };
        setContentView(view);
    }
}
```

View – Add pictures

- First, save your picture (PNG or JPG format), in the Package Explorer go to:
  - res -> drawable-hdpi
- Second, use the command
  ```java
  Bitmap pic = BitmapFactory.decodeResource(getResources(), R.drawable.file);
  ```
- Finally, to draw your picture at origin (x,y) use this command in the onDraw method
  ```java
  canvas.drawBitmap(pic, x, y, null);
  ```
MediaPlayer

- Public class added in API 1. MediaPlayer class allows us to play sounds in our application.
- First, save your sound (MP3 format). In the Package Explorer go to:
  - `res -> raw`
  - If the directory does not exist then create it
- Second, to play your sound use

```java
MediaPlayer mp = MediaPlayer.create(this, R.raw.file);
mp.start();
```
Touch Event

- In order to make your application respond to touch events, you must implement the `onTouchEvent()` method on the Activity class. Once a touch is detected, the method `onTouchEvent()` is called. Note that the coordinates of the touch are saved in variables `x` and `y`.

```java
@Override
public boolean onTouchEvent(MotionEvent e){
    float x = e.getX();
    float y = e.getY();
    //Here you can add your commands
    return false;
}
```

Android manifest

- Every application must have an `AndroidManifest.xml` file (with precisely that name) in its root directory. The manifest presents essential information about the application to the Android system, information the system must have before it can run any of the application's code.
  - It describes the components of the application — the activities, services, etc.
  - It declares the permissions the application has, e.g. camera, internet etc.
  - It declares the minimum level of the Android API that the application requires.
Activity

- Public class added in API 1. An activity helps us to interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI.
- The Activity class is an important part of an application’s overall lifecycle, and the way activities are launched and put together is a fundamental part of the platform’s application model.
- In order to set a layout (XML file) into your activity use inside the onCreate method:

```
setContentView(R.layout.activity_main);
```

Activity - Lifecycle

- There are three key loops you may be interested in monitoring within your activity:
- The entire lifetime of an activity happens between the first call to onCreate(Bundle) up to a single final call to onDestroy().
- The visible lifetime of an activity happens between a call to onStart() until a corresponding call to onStop().
- The foreground lifetime of an activity happens between a call to onResume() until a corresponding call to onPause().
- To add this methods: right click in the activity code → Source -> Override/Implemented Methods...
Activity – Create new Activity

- All activity classes must have a corresponding `<activity>` declaration in their package's AndroidManifest.xml.
- For example:

```xml
<activity
    android:name=".MainActivity"
    android:label="@string/title_activity_main"
    android:theme="@android:style/Theme.NoTitleBar">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category
            android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
```
Activity – Create new Activity

- In the line:
  ```xml
  <category android:name="android.intent.category.LAUNCHER"/>
  ```
  We declared our activity to be a launcher, note that for a second activity you should be using
  ```xml
  <category android:name="android.intent.category.DEAFULT"/>
  ```

- Use the command
  ```java
  Intent intent = new Intent(CurrentActivity.this, NextActivity.class);
  startActivity(intent);
  ```
  to start a new activity within a current running activity.

- To close an activity use
  ```java
  finish();
  ```

Activity – Create new Activity

- Passing Arguments between Activities
  ```java
  Intent intent = new Intent(CurrentActivity.this, NextActivity.class);
  int x = 5;
  intent.putExtra("arg",x);
  startActivity(intent);
  ```
  In the new activity use
  ```java
  int x_new = getIntent().getExtras().getInt("arg");
  ```

- To create a new Activity you can also use
  ```java
  File->New->Other…
  ```
  and then expend Android and choose Android Activity
Thread

- Public class added in API 1, implements Runnable. A Thread is a concurrent unit of execution.
- There are two ways to execute code in a new thread. You can either subclass Thread and overriding its run() method, or construct a new Thread and pass a Runnable to the constructor. In either case, the start() method must be called to actually execute the new Thread.

Thread – Create a new thread

- To create a new Thread you can use

```java
Thread thread = new Thread(new Runnable() {
    public void run(){
        try {
            // add commands
            Thread.sleep(1000); // e.g. sleep 1sec
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
});
```

- In order to start the new thread use

```java
thread.start();
```

- Note that you cannot use stop() in android. The Thread will finish when all his commands are done.
Thread - Example

- Next, we would like to demonstrate a full example. This code causes a ball to move diagonally.

  - Class MainActivity is the Activity
  - Class GameView is the View object. Implements onDraw method.
  - Class GameThread is the Thread object.
    Every second we update the center of the ball in the GameView, and then we update the screen using postInvalidate() method.

```java
public class MainActivity extends Activity{

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new GameView(this));
        GameView v = new GameView(this);
        GameThread t = new GameThread(v);
        t.start();
    }
}
```

```java
public class GameView extends View {

    GameView v;
    int x = 1;

    GameThread(GameView v){
        this.v = v;
    }
    
    @Override
    public void run(){
        try{
            x++;
            Thread.sleep(1000);
            v.up(x);
            v.postInvalidate();
        } catch(Exception e){}
    }
}
```

```java
public class GameThread extends Thread {

    GameView v;

    @Override
    public void run(){
        try{
            x++;
            Thread.sleep(1000);
            v.up(x);
            v.postInvalidate();
        } catch(Exception e){}
    }
}
```
LogCat

• Android uses a special debugging framework called LogCat, which will show every message broadcast from every Activity on the Android device in real time

  Add a Debug Log call using Log.d()

  ```java
  Log.d("tag", "message");
  ```

Toast Notifications

• We will now use a second tool for that same kind of feedback on your Android device, only visual: Toast notifications. Toasts create a condensed text popup that appears briefly on screen before disappearing.

  ```java
  Toast.makeText(getApplicationContext(), "Message", Toast.LENGTH_SHORT).show();
  ```
Summary

• References:
  - Java tutorials:
    - http://docs.oracle.com/javase/tutorial
  - Hebrew guide: http://javabook.co.il/wp/
  - Android tutorials:
    - Hebrew guide: http://iandroid.co.il/academy/
    - http://stackoverflow.com

• This tutorial is only an introduction. If you want to learn more the internet is full with great materials.

Niv Voskoboynik