Semester-Fall

1. **a. What is Mobile OS? Define android OS structure and architecture. 7**

Answer: An OS is system s/w that manages computer hardware and software resources and provides common services for computer program.

A mobile operating system mobiles (or Mobile OS) is an operating system for smart phones, tablets, PDAs or other mobile devices.

While laptops typically are mobile, the OS usually used on them are not considered mobile, as they were originally designed for bigger desktop computers.

The newer OS that are hybrid made for both users

Mobile OS combine features of a personal computer OS with other features for mobile or handheld use like touch screen, cellular, Bluetooth, Wi-Fi, GPS, mobile camera, speech recognitions, camera, speech recognition, voice recorder, music plater etc.

Some mobile operating system are open source software that means there is no restriction on what you can download on it. Or who can develop

**Build and structure of Mobile OS:**

Android operating system is stack of software components which is roughly divided into five sections and four main layers:

Applications

Home dialer SMS/MMS IM Browser camera calculator etc.

Application Framework

Activity manager, windows manager, content providers, views system, notifications manager, resource managers

Libraries

Surface manager, media framework, SQLite, SGL, SSl, Libc

Android Runtime

Core libraries

Dalvik Virtual Machine

Linux Kernel

Display driver, Camera Driver, Bluetooth driver, USB driver, Keyboard driver, Wifi Driver

Fig: Architecture of Mobile OS

1. **Linux Kernel**

* At the bottom of the layers is Linux kernel.
* Provides a level of abstraction between the device hardware and it contains all essential hardware drivers like camera, keyboard, display etc.
* Also kernel handles networking and a vast array of device drivers for interfacing with peripheral hardware.

1. **Libraries:**

* On the top of the Linux Kernel there is a set of libraries including open source web browser engine, webkit, well known libraries libc, SQlite database which is useful repository for storage and sharing of applications data, libraries to play and record audio, video, SSL libraries for internet security etc.

**Android Libraries:**

* This category encompasses those Java based libraries that are specific to android development.
* android.app: Provides access to the application model and is the corner stone of all android applications.
* android.content: Facilities content access, publishing and messaging between application and application components.
* android.database: used to access data published by content provide and includes SQLite database management classes.
* android.opengl: a java interface to the open GL 3D graphics, randomly API including messages, system services and inter-process communication.
* android.text: used to render and manipulate text on device display
* android.view: fundamental building blocks of application user interface
* android.widget: rich collection of pre-built user interfaces components such as buttons, labels etc.
* android.webkit: a set of classes intended to allow web building capabilities to be built two applications.

1. **Android Runtime:**

* That section of architecture and available on the second layer from bottom.
* Provides key components called Dalvik VM which is a kind of JVM designed and optimized for Android.
* Android runtime also provides a set of core libraries which enable Android application developers to write android applicationusing standard Java PL.

**Android framework:**

Provides many higher level services to applications in the form of Java classes.

* **Activity manager:** Controls all aspects of the application life cycle and activity stock.
* **Content providers:** Publishing and share data with applications
* **Resource managers:** provides access to non-code embedded resources such as strings, color settings and user interface layouts.
* **Notification manger:** displays events and notification to users
* **View system:** creates applications user interface

1. **Application:** All application applications resides at the top layer. E.g. contacts, books, browsers, games etc.

**b. Define android runtime, Jit and AOT in detail. 8**

Answer:

**2. a. Explain JVM. Why java is called platform independent? 7**

Answer: Java is an object-oriented programming language developed by SUN Micro-systems, a company best known for its high-end Unix workstations. The java language was designed to be small, simple and portable across platform and operating systems.

The most compelling reason for Java to be an out-standing language at the moment is its adaptability in any environment making programming task.

Java as a Platform-independent:

Platform independence is one of the most significant advantages that Java has over other programming language, particularly for systems that need to work on many different platform. Java is platform independent at both the source and the binary level.

At the source level, java’s primitive data types have consistent size across all development platforms. Java’s foundation libraries make it easy to write code that can be moved to platform without the need to re-write it to work with that platform.

Platform independence doesn’t stop at the source level, however Java binary files are also platform independent and con take multiple problems without the need to recompile the source. It is because, Java binary files are actually in a form called bytecodes.

A byte codes are a set of instructions that looks a lot like some machine codes, but is not specific to any one process.

Pentium

SPARC

Power PC

Your code

Compiler

Java code

Compiler

Java byte code

Pentium

Power PC

SPARC

Fig (1)

Fig (2)

Figure (1), shows the traditional compiled program in which the code we present are compiled to the machine in which it is compiled into. Accordingly, the binary files are created with respect to the machine, it was compiled.

Figure (2), shows that the java programs in which the code we present are firstly compiled by Java compiler according to the machine it was presented into. The so code is transformed into a java byte code which shows the platform independence. According to the machine it needs compiling, the java interpreter acts and it is compiled.

The only disadvantage of using bytecode is in its execution speed. Because system specific programs run directly on the hardware for which they are compile, they run significantly faster than Java bytecodes which must be processed by the interpreter.

1. b. Write the sample steps to create the sample app in Eclipse IDE.

Answer: Before application is written in Eclipse IDE, it is necessary to install Eclipse.

Assuming the following,

* New Eclipse has been started with default settings
* Familiar with the basic Eclipse workbench mechanisms, such as views and perspectives
* Eclipse IDE with Android Developer Tools (ADT) plugin to build the application and Android Emulator - Android Virtual Device (AVD) to run the application which will draw 'Hello World!' text on the screen.

Let us proceed a simple application which will print “Hello world!” or any text messages.

**Step 1: Create Android project:**

The first step is create a simple android project using Eclipse IDE. To do so,   
Select from the menu File --> New --> Other --> Android --> Android Application Project (say 'AndroidHello!') and click Nextbutton.

**Step 2: Configure Project Settings:**

A new dialog box appears where we have to enter Application name (say AndroidHello), Project Name: (say, AndroidHello), Package Name (Say, com.example.android). Select ‘Minimum Required SDK’ (lowest version of Android that this app supports). ‘Target SDK’ (highest version of Android with which this application has been tested), ‘Compile With’ (Platform version against which this application will be compiled with) and ‘Theme’ (Android UI style) from the corresponding theme. To make it simple we can leave the dropdown value as it si. Click Next button.

With Next button clicked, a new android dialog box will appear stating location to save the file. We can change its location or keep as it is. Click Next to proceed.

**Step 3: Configure App launcher Icon:**

Choose your app icon and configure as per your requirement.

**Step 4: Create Activity**

Choose an activity template (say ‘BlankActivity’) and click Next button.

Enter ‘Activity Name’ (say ‘Helloactivity’) and click Finish Button. If **Finish** button is not enabled and **Next** is enabled that means required dependencies (Supporting library) are not installed. In this case click **Next** button and hit '**Install/Upgrade**' button to install or upgrade required dependencies. Finally click **Finish** button.

**Step 5: Overall Project Structure:**

Android project will be created with some default files. It consists of src (helloActivity.java), gen (R.java), bin, libs, res/layouts (activity\_hello.xml), res/values (strings.xml) and AndroidManifest.xml

Android\_hello.xml (layout) will be opened using ‘Android Common XML editor’. Here we can build UI by simply dragging and dropping UI components from the Palette

**Step 6: Code:**

'hello\_world' resource string contains the message 'Hello world!' which will be shown on launching of the application.

File: strings.xml

|  |  |
| --- | --- |
|  | <?xml version="1.0" encoding="utf-8"?>  <resources>      <string name="app\_name">AndroidHello</string>      <string name="hello\_world">Hello world!</string>      <string name="menu\_settings">Settings</string>   </resources> |

'activity\_hello.xml' is the layout built using '**Android Common XML Editor**'. Instead of using a hard-coded string value ('Hello world!') in '<TextView>' element, the "@string/hello\_world" value refers to a string resource defined in strings.xml.

File : activity\_hello.xml

|  |  |
| --- | --- |
|  | <RelativeLayout xmlns:android="<http://schemas.android.com/apk/res/android>"      xmlns:tools="<http://schemas.android.com/tools>"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent"      tools:context=".HelloActivity" >        <TextView          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:layout\_centerHorizontal="true"          android:layout\_centerVertical="true"          android:text="@string/hello\_world" />   </RelativeLayout> |

For this application we do not require to change anything in the generated activity code.

File : HelloActivity.java

|  |  |
| --- | --- |
|  | package com.example.android;   import android.os.Bundle;  import android.app.Activity;  import android.view.Menu;  public class HelloActivity extends Activity {      @Override      protected void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          setContentView(R.layout.activity\_hello);      }      @Override      public boolean onCreateOptionsMenu(Menu menu) {          // Inflate the menu; this adds items to the action bar if it is present.          getMenuInflater().inflate(R.menu.activity\_hello, menu);          return true;      }  } |

## Step 7:Run Configuration

Right click on the project and from the context menu select 'Run As' --> 'Run Configurations’

If there is no Android Virtual Device (AVD) already created, then click **Manager** Button to create one. Configure AVD with required configuration and click **OK** button.

**Step 8. Run Application**

Right click on the project and from the context menu select 'Run As' --> 'Android Application'.

**Step 9. Output**

Eclipse ADT will start the AVD and launch your application with 'Hello world!' message on the screen.

1. **a. What do you mean by Default and launcher Activity? Explain the primary forms of intent.**

**Answer**: An Android **Intent** is an abstract description of an operation to be performed. It can be used with **startActivity** to launch an Activity, **broadcastIntent** to send it to any interested BroadcastReceiver components, and **startService(Intent)** or**bindService(Intent, ServiceConnection, int)**to communicate with a background Service.

The primary forms of Intents are:

### **Action**

This is mandatory part of the Intent object and is a string naming the action to be performed — or, in the case of broadcast intents, the action that took place and is being reported. The action largely determines how the rest of the intent object is structured . The Intent class defines a number of action constants corresponding to different intents. Here is a list of [Android Intent Standard Actions](http://www.tutorialspoint.com/android/android_intent_standard_actions.htm)

The action in an Intent object can be set by the setAction() method and read by getAction().

### **Data**

Adds a data specification to an intent filter. The specification can be just a data type (the mimeType attribute), just a URI, or both a data type and a URI. A URI is specified by separate attributes for each of its parts −

These attributes that specify the URL format are optional, but also mutually dependent −

* If a scheme is not specified for the intent filter, all the other URI attributes are ignored.
* If a host is not specified for the filter, the port attribute and all the path attributes are ignored.

The setData() method specifies data only as a URI, setType() specifies it only as a MIME type, and setDataAndType() specifies it as both a URI and a MIME type. The URI is read by getData() and the type by getType().

Some examples of action/data pairs are −

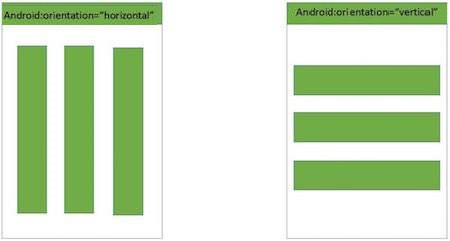
|  |  |
| --- | --- |
| **S.N.** | **Action/Data Pair & Description** |
| 1 | **ACTION\_VIEW content://contacts/people/1**  Display information about the person whose identifier is "1". |
| 2 | **ACTION\_DIAL content://contacts/people/1**  Display the phone dialer with the person filled in. |
| 3 | **ACTION\_VIEW** [**tel:123**](tel:123)  Display the phone dialer with the given number filled in. |
| 4 | **ACTION\_DIAL** [**tel:123**](tel:123)  Display the phone dialer with the given number filled in. |
| 5 | **ACTION\_EDIT content://contacts/people/1**  Edit information about the person whose identifier is "1". |
| 6 | **ACTION\_VIEW content://contacts/people/**  Display a list of people, which the user can browse through. |
| 7 | **ACTION\_SET\_WALLPAPER**  Show settings for choosing wallpaper |
| 8 | **ACTION\_SYNC**  It going to be synchronous the data,Constant Value is**android.intent.action.SYNC** |
| 9 | **ACTION\_SYSTEM\_TUTORIAL**  It will start the platform-defined tutorial(Default tutorial or start up tutorial) |
| 10 | **ACTION\_TIMEZONE\_CHANGED**  It intimates when time zone has changed |
| 11 | **ACTION\_UNINSTALL\_PACKAGE**  It is used to run default uninstaller |

b. **How many types of resources layouts are there in android, describe any two of them with use cases.**

**Answer:** There are various types of resources layouts in android:

They are: Liner layout, Relative layout, Table layout, Absolute layout, Frame layout, List layout and Grid layout.

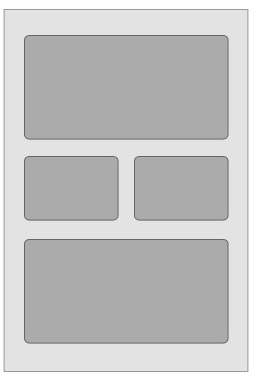
1. **Linear layout:** Android LinerLayout is a view group that aligns all children in vertically or horizontally. Follwing are the important attributes specific to Linear:



|  |  |
| --- | --- |
| **Attribute** | **Description** |
| android:id | This is the ID which uniquely identifies the layout. |
| android:baselineAligned | This must be a boolean value, either "true" or "false" and prevents the layout from aligning its children's baselines. |
| android:orientation | This specifies the direction of arrangement and you will use "horizontal" for a row, "vertical" for a column. The default is horizontal. |
| android:weightSum | Sum up of child weight |
| For more: http://www.tutorialspoint.com/android/android\_linear\_layout.htm | |

1. **Relative Layout**: Android RelativeLayout enables you to specify how child views are positioned relative to each other. The position of each view can be specified as relative to sibling elements or relative to the parent.

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| android:id | This is the ID which uniquely identifies the layout. |
| android:gravity | This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center\_vertical, center\_horizontal etc. |
| android:ignoreGravity | This indicates what view should not be affected by gravity. |

Following are the important attributes specific to RelativeLayout −

Using RelativeLayout, you can align two elements by right border, or make one below another, centered in the screen, centered left, and so on. By default, all child views are drawn at the top-left of the layout, so you must define the position of each view using the various layout properties available from **RelativeLayout.LayoutParams** and few of the important attributes are given below −

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| android:layout\_above | Positions the bottom edge of this view above the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name" |
| android:layout\_alignBottom | Makes the bottom edge of this view match the bottom edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name". |
| android:layout\_alignLeft | Makes the left edge of this view match the left edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name". |
| For more <http://www.tutorialspoint.com/android/android_relative_layout.htm> | |
| For more: http://www.tutorialspoint.com/android/android\_user\_interface\_layouts.htm | |

1. **a. Explain about Android 3rd party UI/UX library.**

**Answer:** Over past few years, the age of mobile development become a fiercely-competitive world of serious money making business. Android developer’s network is growing rapidly. As the network grows, there are many open source libraries available. Although using someone written code is not be recommended always, but sometimes it helps in getting the application quicker to market.

Below are some of the popular android third party libraries.

**1. User Interface Libraries**

|  |  |
| --- | --- |
| **Library** | **Purpose** |
| [ActionBarSherlock](http://actionbarsherlock.com/) | ActionBarSherlock is an extension of the support library designed to facilitate the use of the action bar design pattern across all versions of Android with a single API. ActionBarSherlock was widely used before Google introduced AppCompact. |
| [ActionBar PullToRefresh](https://github.com/chrisbanes/ActionBar-PullToRefresh) | ActionBar PullToRefresh provides an easy way to add a modern version of the pull-to-refresh interaction to your application. |
| [Android PullToRefresh](https://github.com/chrisbanes/Android-PullToRefresh) | This project aims to provide a reusable Pull to Refresh widget for Android. It was originally based on Johan Nilsson's library (mainly for graphics, strings and animations), but these have been replaced since. |
| [View Pager Indicator](http://viewpagerindicator.com/) | Paging indicator widgets compatible with the ViewPager from the Android Support Library and ActionBarSherlock. |
| [Nine Old Android](http://nineoldandroids.com/) | Android library for using the Honeycomb (Android 3.0) animation API on all versions of the platform back to 1.0! Animation prior to Honeycomb was very limited in what it could accomplish so in Android 3.x a new API was written. |
| [A Chart Engine](https://code.google.com/p/achartengine/) | AChartEngine is a charting library for Android applications. It currently supports all major and widely used chart types. |

**2. Android Networking Libraries**

|  |  |
| --- | --- |
| **Library** | **Purpose** |
| [okHttp](http://square.github.io/okhttp/) | HTTP is the way modern applications network. It’s how we exchange data & media. Doing HTTP efficiently makes your stuff load faster and saves bandwidth. |
| [Volley](http://developer.android.com/training/volley/index.html) | Volley is an HTTP library that makes networking for Android apps easier and most importantly, faster. Volley is available through the open AOSP repository. |
| [Android Asynchronous Http Client](http://loopj.com/android-async-http/) | An asynchronous callback-based Http client for Android built on top of Apache’s HttpClient libraries. All requests are made outside of your app’s main UI thread, but any callback logic will be executed on the same thread as the callback was created using Android’s Handler message passing. |
| [Google GSON](https://code.google.com/p/google-gson/) | Gson is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object. Gson can work with arbitrary Java objects including pre-existing objects that you do not have source-code of. |
| [Retrofit](http://square.github.io/retrofit/) | Retrofit is a type-safe REST client for Android and Java. This uses annotation to describe HTTP server calls. This is an elegant solution for managing REST API calls in in android application. |

**3. Image Downloading And Caching Libraries**

|  |  |
| --- | --- |
| **Library** | **Purpose** |
| [Picasso](http://square.github.io/picasso/#download) | A powerful image downloading and caching library for Android. |
| [Universal Image Loader](https://github.com/nostra13/Android-Universal-Image-Loader) | UIL aims to provide a powerful, flexible and highly customizable instrument for image loading, caching and displaying. It provides a lot of configuration options and good control over the image loading and caching process. |

**4. Database Helper Libraries**

|  |  |
| --- | --- |
| **Library** | **Purpose** |
| [ActiveAndroid](http://www.activeandroid.com/) | ActiveAndroid is an active record style ORM (object relational mapper). It allows you to save and retrieve SQLite database records without ever writing a single SQL statement. Each database record is wrapped neatly into a class with methods like save() and delete(). |
| [ORM Lite](http://ormlite.com/) | Object Relational Mapping Lite (ORM Lite) provides some simple, lightweight functionality for persisting Java objects to SQL databases while avoiding the complexity and overhead of more standard ORM packages. |
| [Green DOA](http://greendao-orm.com/) | GreenDAO is an open source project to help Android developers working with data stored in SQLite. GreenDAO does all the SQLite helper work for you. It maps Java objects to database tables (often called ORM). This way you can store, update, delete, and query for Java objects using a simple object oriented API. |

**b. What is alarm Manager? What are the types of alarm?**

**Answer:** Android AlarmManager allows you to access system alarm.

By the help of **Android AlarmManager** in android, you can schedule your application to run at a specific time in the future. It works whether your phone is running or not.

The Android AlarmManager holds a CPU wake lock that provides guarantee not to sleep the phone until broadcast is handled.

Class: AlarmManager.AlarmclockInfo

Interface: AlarmManager.OnAlarmListener

**Types of Alarms**

1. ELAPSED\_REALTIME:

If an alarm of this type is set, it indicates the time after boot up of the device, after which the alarm should go off. Here, if the device is asleep while the alarm is about to be triggered, the device is not woken up and the alarm is fired the next time the device is woken up. Alarm time in SystemClock.eplapsedRealtime(). Data type used is int. Constant value: 3(0x000000003) (8 zeros)

Syntax: int ELAPSED\_REALTIME

1. ELAPSED\_REALTIME\_WAKEUP: This is identical to ELAPSED\_REALTIME alarm, except that the device is woken up when the alarm is about to be triggered. Alarm time in SystemClock.eplapsedRealtime().Data type used is int. Constant value: 2(0x000000002)

Syntax: int ELAPSED\_REALTIME\_WAKEUP

1. RTC: An alarm of this type indicates the exact time at which the alarm should go off. If the device is asleep when the alarm is about to be triggered, the device is not woken up(similar to ELAPSED\_REALTIME) and the alarm is thrown when the device is woken up next. Alarm time in System.currentTimeMillis().Data type used is int. Constant value: 1 (0x000000001)

Syntax: int RTC

1. RTC\_WAKEUP:  It is similar to RTC alarm, except that the device is woken upwhen the alarm goes off. Alarm time in System.currentTimeMillis().Data type used is int. Constant value: 0 (0x00000000) Syntax: int RTC\_WAKEUP

The following is an example of scheduling a repeating alarm every 15 minutes starting from 16th April 22:18 hrs.

  Calendar alarmTime = Calendar.getInstance();  
       *alarmTime.set(Calendar.DAY\_OF\_MONTH*, 16);  
       *alarmTime.set(Calendar.MONTH,3*);  
       *alarmTime.set(Calendar.HOUR\_OF\_DAY*, 22);  
       *alarmTime.set(Calendar.MINUTE*, 18);

        AlarmManager alarms = (AlarmManager) getSystemService(ALARM\_SERVICE);  
        alarms.setRepeating(AlarmManager.RTC\_WAKEUP,alarmTime.getTimeInMillis(),  
                AlarmManager.INTERVAL\_FIFTEEN\_MINUTES, sender);

The following is an example of an alarm that first goes off  10 seconds after boot up of the device and then repeats every 15 minutes thereafter.

   AlarmManager alarms = (AlarmManager) getSystemService(ALARM\_SERVICE);  
        alarms.setRepeating(AlarmManager.ELAPSED\_REALTIME\_WAKEUP,10\*1000,  
                AlarmManager.INTERVAL\_FIFTEEN\_MINUTES, sender);

In both the examples, sender is the name of the PendingIntent.

1. **a. How many types of animation can be done in android? Explain 7**

**Answer:** Animation is the process of creating motion and shape change. Android provides a variety of powerful APIS for applying animation to UI elements and drawing custom 2D and 3D graphics.

Android has three main types of animation. They are:

* Frame based
* View based
* Property based

These three types of animation fall into two categories: frame animations and tweened animations. Unsurprisingly, frame-base animations are a type of frame animation, whereas View- and Property-based animations are types of tweened animations.

Frame Based animation: A frame animation is one of the simplest and most straightforward types of animations. This type of animation is implemented by displaying a succession of slightly changing images one after another. By displaying these images in quick succession we have a resulting smooth animation.

Tweened based animation: The android framework provides two animation systems:

Property Animation:

View animation:

In addition to these two systems we can utilize drawable animation, which allows to load drawable resources and display them one frame after another.

Property animation:

Introduced in Android 3. (API level 11)

Let’s us animate properties of any object including ones that are not rendered to the screen.

Property animation system allows use to change any object property other time, regardless of whether it draws to the screen or not.

To animate something, we specify the object property that we want to animate such as an object’s position on the screen, how long we want to animate if for and what values we want to animate between.

Property animation lets’ us define the following characteristics:

1. Duration: we can specify the length of the animation. The default length is 300ms.
2. Time interpolation: States how a vein for the property are calculated as a function of the animation’s event elapsed time.
3. Repeat count and behavior: Specify whether or not to have an animation respect when it reaches the end of a duration and how many times to repeat the animation.
4. Animator sets: We can group animations into logical sets that play together or sequentially or after specified delays.
5. Frame refresh delay: we can specify how often to refresh frames of our animation. The defaults is set to refresh every 10 ms.

View animation:

View animation is the older system and can be used for views.

Relatively easy to setup and offers enough capabilities to meet many applications needs.

Used to perform Tweened animation in views

Tween animation calculates the animation with information such as the start point, end point, size rotaion and other common aspects of animation.

Tween Animation can perform a series of simple transformations (positions, size, rotation and transparency) on the contents of the view object.

For example:

If we have a Textview object, we can move, rotate, grow or shrink the text

If it has a background image, the background image will be transformed along with the text.

Andtoid has provided us a class called Animation.

In order to perform animation in android, we are going to call a static functions loadAnimation() of the class AnimationUtils.

We receive the result in an instance of Animation Object.

Syntax: Animation animation = AnimationUtils.loadAnimation(getApplicationContext().R.anim.myanimation);

The second parameter is the name of our animation XML file.

We have to create a new folder called anim under res directory and make an XML File under anim Folder

This animation class has many useful functions which are listed below:

|  |  |
| --- | --- |
| S.No | Method and Description |
| 1. | Start(): this method starts the animation |
| 2. | SetDuration(long duration): This methods Sets the duration of a n animation. |
| 3. | getDuration(): this method gets the duration which is set by above method. |
| 4. | end(): This method ends the animation |
| 5. | Cancel(): This method cancels the animation |

In order to apply this animation to an object we will just call the StartAnimation() method of the object. Its syntax is:

ImageView images = (ImageView)findViewBYID (R.id.ImageViews)

Images.StartAnimation(animation);

import android.view.animation.Animation;

import android.view.animation.AnimatioUtils;

b. **Describe processes, threads and broadcastReceiver class in android.**

**Answer:** Processes:

The class Process provides methods for performing input from the process, performing output to the process, waiting for the process to complete, checking the exit status of the process, and destroying (killing) the process.

As of 1.5, [start()](https://developer.android.com/reference/java/lang/ProcessBuilder.html#start()) is the preferred way to create a Process.

Tools for managing OS processes

Namespace: Android.OS

Assembly: Mono.Android (in Mono.Android.dll)

Assembly versions: 0.0.0.0

Since: Added in API level 1

Java.lang.Object: java.lang.Process

public abstract class Process extends Object{…..}

Public constructors: Process ( )

Public properties:

[read-only]static ElapsedCpuTime Int64. Returns elapsed ms of the time, process has run

Protected constructors: Process(IntPtr, JnihandedOwnership): A constructor use when creating managed representaions of JNI objects; called by the runtime

Public Fields:

const BluetoothGid Int32(2000)

const FirstApplicationUid Int32(10000) Defines the start of a range of UIDs(and GIDs), going from this number to Process.LastApplicationUid

const LastApplicationUid Int32(99999). Last of application-specific UIDs starting at

Process.FirstAppliationUid

const PhoneUid Int32(1001). Defines the UID/GID under which the telephony code runs

const SystemUid Int32(1000). Defines the UID/GID under which system code runs

Protected properties:

[read-only]override ThresholdClass IntPtr. This API supports the Mono for Android

Infrastructure and is not intended to be used directly from your code

[read-only]override ThresholdType Type. This API supports the Mono for Android Infrasttucture and is not intended to be used directly from your code

Threads: A *thread* is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently.

Every thread has a priority. Threads with higher priority are executed in preference to threads with lower priority. Each thread may or may not also be marked as a daemon. When code running in some thread creates a new Thread object, the new thread has its priority initially set equal to the priority of the creating thread, and is a daemon thread if and only if the creating thread is a daemon.

There are two ways to create a new thread of execution. One is to declare a class to be a subclass ofThread. This subclass should override the run method of class Thread. An instance of the subclass can then be allocated and started. For example, a thread that computes primes larger than a stated value could be written as follows:

     class PrimeThread extends Thread {  
         long minPrime;  
         PrimeThread(long minPrime) {  
             this.minPrime = minPrime;  
         }  
  
         public void run() {  
             // compute primes larger than minPrime  
              . . .  
         }  
     }

The following code would then create a thread and start it running:

     PrimeThread p = new PrimeThread(143);  
     p.start();

The other way to create a thread is to declare a class that implements the Runnable interface. That class then implements the run method. An instance of the class can then be allocated, passed as an argument when creating Thread, and started. The same example in this other style looks like the following:

     class PrimeRun implements Runnable {  
         long minPrime;  
         PrimeRun(long minPrime) {  
             this.minPrime = minPrime;  
         }  
  
         public void run() {  
             // compute primes larger than minPrime  
              . . .  
         }  
     }

The following code would then create a thread and start it running:

     PrimeRun p = new PrimeRun(143);  
     new Thread(p).start();

Every thread has a name for identification purposes. More than one thread may have the same name. If a name is not specified when a thread is created, a new name is generated for it.

BroadcastReceiver class:

Broadcast receivers simply respond to broadcast messages from other application or from the system. For e.g.: applications can also initiate broadcast to let other applications know that some data has been downloaded to the device and is available for them to use. So broadcast receiver will intercept this communication and will initiate appropriate action.

public class MyReceiver extends BroadcastReceiver {

public void onReceive{context,intent){

…………………}

}

* Broadcast Receiver simply respond to broadcast messages from other applications or from the system itself.
* These messages are sometimes called events or intents.

For example: Applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use.

The important steps to make BroadcastReceiver works for the system Broadcasted Intents:

* Creating the Broadcast Receiver
* Registering Broadcast Receiver

Creating the BroadCastReceiver

Public class MyReceiver extends BroadCastReceiver{

Public void onReceive(Context c, Intent intent){

Toast.makeText(Context,”Intent Detected”, Toast.LENGTH\_long).show();}}

* A broadcast receiver is implemented as a subclass of BroadCastReceiver class and overriding the onReceive() method where each object s received as an intent object parameter.

Registering Broadcast Receiver:

An application listens for specific broadcast intents by registering a broadcast receiver in AnroidManifest.xml for intents to observe

6. b. What is data management? Explain Internal and external storage.

When we develop Android applications, we have a number of options in terms of how we store and manage our data. Depending on the app, we may find ourselves using more than one of these for different data tasks.

Our data storage options one of the following:

1. Shared preferences(Store private primitive data in key value pairs)
2. Internal Storage (Store private data on the device memory)
3. External Storage (Store public data on the shared external storage)
4. SQLite database (Store structured data in a private database)
5. Network Connection (Store data on the web with our own network Server)

## Internal Storage

We can store data items to the internal storage on our devices. The advantage to this is that it does not rely on any external media, but the disadvantage is that some devices have extremely limited amounts of storage space available. Storing to internal memory essentially means saving data files within the internal device directory structure.

The following Java code demonstrates opening a file for data output:

//get a FileOutputStream object by passing the file-name

FileOutputStream dataFileOutput = openFileOutput("datafile", Context.MODE\_PRIVATE);

When you save files to the internal storage, other applications can't access them.

**External Storage (SD cards):**

It is an ultra-small flash memory card designed to provide high-opacity memory in a small size. SD cards are used in many small portable devices. Manu Android users are dependent on external storage as SD cards due to its insufficient internal storage. SD cards offer your apps an increased amount of storage space, but they impose additional considerations. For example, you cannot assume that a user will have an SD card or other external media resource. For this reason your programming code needs to carry out checks before storing data in this way.

if(Environment.getExternalStorageState().equals(Environment.MEDIA\_MOUNTED)) {

//can use the external storage}

It's also worth bearing in mind that files saved to an SD card can be accessed by other apps and by the user directly, so the data is more vulnerable to corruption.

[For more: <http://code.tutsplus.com/tutorials/data-management-options-for-android-applications--mobile-8621>]

7. Write short notes on:

a. Byte code verifier:

When a class loader presents the bytecodes of a newly loaded Java platform class to the virtual machine, these bytecodes are first inspected by a *verifier.* The verifier checks that the instructions cannot perform actions that are obviously damaging. All classes except for system classes are verified. We can, however, deactivate verification with the undocumented -noverify option.

For example,

**java -noverify Hello**

Here are some of the checks that the verifier carries out:

* Variables are initialized before they are used.
* Method calls match the types of object references.
* Rules for accessing private data and methods are not violated.
* Local variable accesses fall within the runtime stack.
* The runtime stack does not overflow.

If any of these checks fails, then the class is considered corrupted and will not be loaded.

[For more: <http://www.informit.com/articles/article.aspx?p=1187967&seqNum=2>

http://www.oracle.com/technetwork/java/security-136118.html]

b. Fragment vs activity

**1. Dealing with device form-factor differences:**

* A single Activity has to provide a lot of special case handling for various form factors
* A separate Activity is created for each form factor with the non-UI details duplicated or otherwise shared across each Activity

Fragments eliminate this problem by taking on the UI details and leaving the other responsibilities to the Activity. This way a separate Fragment can be created for each form factor with the form factor specific UI details being the only responsibilities of each Fragment.

**2. Passing information between app screens:**

Activity has to pass information between screens while with fragment, fragrmetns always exists within the context of a given activity and can always access that activity. By storing the information of interest within the Activity, the Fragment for each screen can simply access the object reference through the Activity.

**3. User interface organization:**

UI metaphors organization gets better than the activity implementation UI. Fragments using allows better look and style enabling less space and more functionality.

**4. Advanced UI metaphors:**

As the use of Fragments matures, they are an increasingly important part of rich UI design and are becoming the foundation of some of the more advanced UI metaphors

[for more: https://www.pluralsight.com/blog/software-development/android-fragments]

1. Multi touch and Gesture:

Multi-touch gesture happens when more than one finger touches the screen at the same time. Android allows us to detect these gestures.

Android system generates the following touch events whenever multiple fingers touches the screen at the same time.

|  |  |
| --- | --- |
| **Sr.No** | **Event & description** |
| 1 | **ACTION\_DOWN**  For the first pointer that touches the screen. This starts the gesture. |
| 2 | **ACTION\_POINTER\_DOWN**  For extra pointers that enter the screen beyond the first. |
| 3 | **ACTION\_MOVE**  A change has happened during a press gesture. |
| 4 | **ACTION\_POINTER\_UP**  Sent when a non-primary pointer goes up. |
| 5 | **ACTION\_UP**  Sent when the last pointer leaves the screen. |

So in order to detect any of the above mention event, we need to override **onTouchEvent()** method and check these events manually. Its syntax is given below −

public boolean onTouchEvent(MotionEvent ev){

final int actionPeformed = ev.getAction();

switch(actionPeformed){

case MotionEvent.ACTION\_DOWN:{

break;

}

case MotionEvent.ACTION\_MOVE:{

break;

}

return true;

}

In these cases, you can perform any calculation you like. For example zooming, shrinking etc. To get X and Y axis, you can call **getX()** and **getY()** method. Its syntax is given below:

final float x = ev.getX();

final float y = ev.getY();

Apart from these methods, there are other methods provided by this MotionEvent class for better dealing with multitouch. These methods are listed below:

|  |  |
| --- | --- |
| **Sr.No** | **Method & description** |
| 1 | **getAction()**  This method returns the kind of action being performed |
| 2 | **getPressure()**  This method returns the current pressure of this event for the first index |
| 3 | **getRawX()**  This method returns the original raw X coordinate of this event |
| 4 | **getRawY()**  This method returns the original raw Y coordinate of this event |
| 5 | **getSize()**  This method returns the size for the first pointer index |
| 6 | **getSource()**  This method gets the source of the event |
| 7 | **getXPrecision()**  This method return the precision of the X coordinates being reported |
| 8 | **getYPrecision()**  This method return the precision of the Y coordinates being reported |

[For more: https://www.tutorialspoint.com//android/android\_multitouch.htm]