Name of paper : **Android Programming**

Course : **B.Sc. (H) Computer Science**

Semester : **III**

Max. Marks : **25**

Website : **https://www.tutorialsduniya.com**

Section A (Compulsory)

Q1. (a) @string refers to a string in the .... .xml file. **[1 mark]**

Ans: strings.xml file.

(b) In Android:layout_column attribute the value starts from .... **[1mark]**

Ans: Value starts from 0.

(c) Name the tag that contains minSdkVersion. **[1mark]**

Ans: <manifest> tag.

(d) What is APK in Android ? **[1 mark]**

Ans: Android Package Kit is a collection of different files (like source code, images, icons, audio, video etc.) compiled and bundled into one file for the distribution and installation purpose.

(e) Name any 2 layouts available is android. **[1mark]**

Ans: Linear Layout and Relative Layout.

(f) What is the XML Tag used for Menu files ? **[1 mark]**

- **(i)** **<main_menu>**
- **(ii)** **<menu>**
- **(iii)** **<menu_layout>**
- **(iv)** **none of these**

Ans: (ii) <menu>

(g) Write the command used for installing a new application to the emulator or actual connected device. **[1 mark]**

Ans: emulator @avd_name [{-option [value]} ...]

(h) Which class contains the onClick() method ? **[1 mark]**

Ans: View.OnClickListener class.

(i) The symbolic constant for Menu Key is KEYCODE_SET_MENU. True or False ? [1 mark]

Ans: False, the symbolic constant for Menu Key is KEYCODE_ MENU.

(j) Android 3.0.X version having API 11 is named as .... [1 mark]

Ans: Honeycomb.

Section B (Attempt any three)

Q2. Explain the different states and methods of the Android Lifecycle stages with diagram and example. [5 marks]

Ans: The different states and methods of the Android Lifecycle stages are as follows:

1) **onCreate():** This method fires when the system creates our activity. Our implementation should initialize the essential components of our activity.

2) **onStart():** As onCreate() exits, the activity enters the Started state, and the activity becomes visible to the user. This callback contains what amounts to the activity's final preparations for coming to the foreground and becoming interactive.

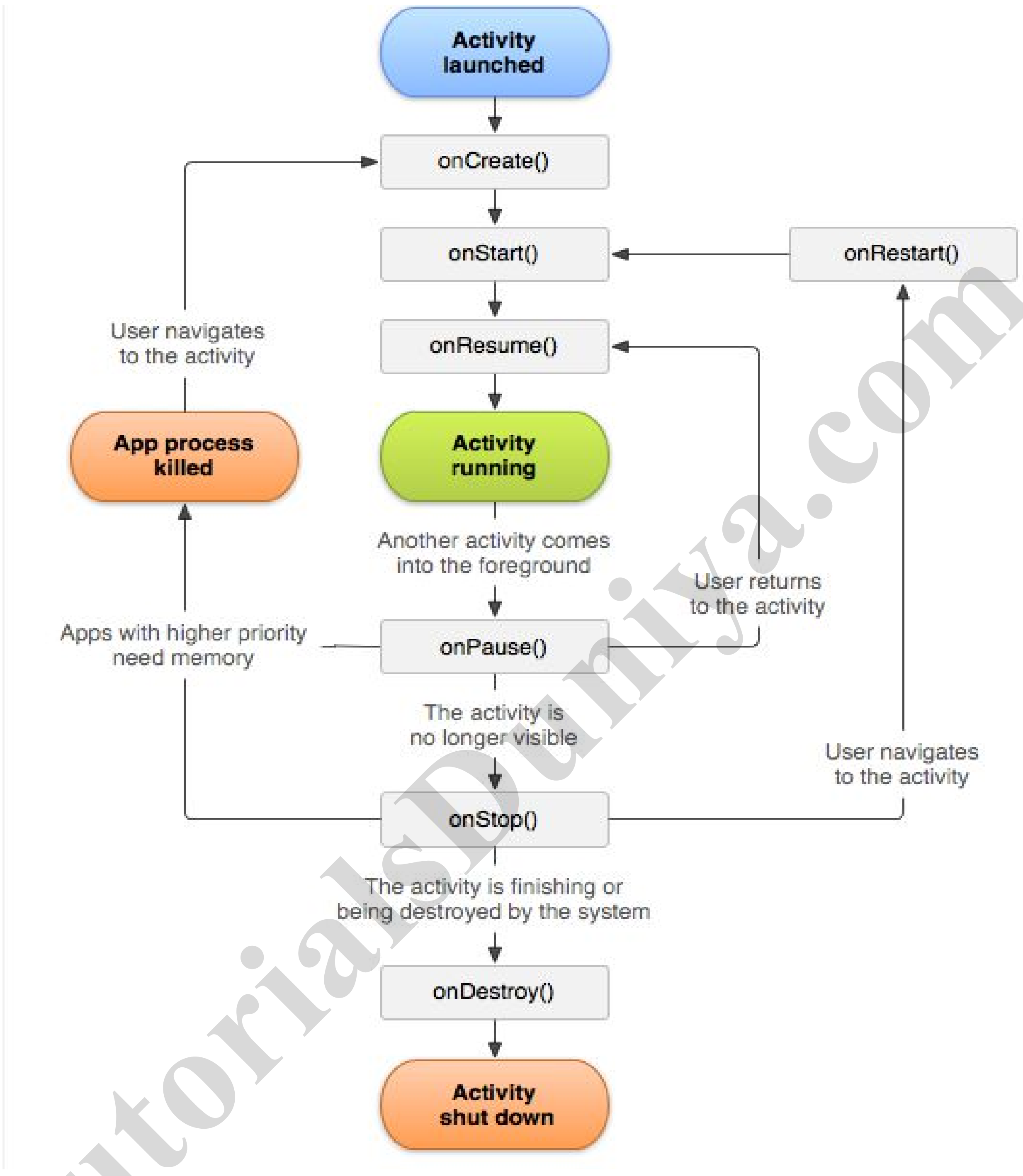
3) **onResume():** The system invokes this callback just before the activity starts interacting with the user. At this point, the activity is at the top of the activity stack, and captures all user input. Most of an app's core functionality is implemented in the onResume() method.

4) **onPause():** The system calls onPause() when the activity loses focus and enters a Paused state. This state occurs when, for example, the user taps the Back or Overlay button. When the system calls onPause() for our activity, it technically means your activity is still partially visible, but is an indication that the user is leaving the activity.

5) **onStop():** The system calls onStop() when the activity is no longer visible to the user. This may happen because the activity is being destroyed, a new activity is starting, or an existing activity is entering a Resumed state and is covering the stopped activity.

6) **onRestart():** The system invokes this callback when an activity in the Stopped state is about to restart. onRestart() restores the state of the activity from the time that it was stopped.

7) **onDestroy():** The system invokes this callback before an activity is destroyed. This callback is the final one that the activity receives. onDestroy() is usually implemented to ensure that all of an activity's resources are released when the activity, or the process containing it, is destroyed.


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

Q3. What are Intents and Intent filters ? How many types of Intents are there in Android system ? How is intent built ? Explain with example. [5 marks]

Ans: Intent is a messaging object that we can use to request an action from another app component.

Intent filter is an expression in an app's manifest file that specifies the type of intents that the component would like to receive.

There are 2 types of Intents in the Android system,

1. Explicit Intent: It specifies the component to start by name (the fully-qualified class name). It is used to start a component in our own app, because we know the class name of the activity or service we want to start.

Example, start a new activity in response to a user action or start a service to download a file in background.

```
Intent i = new Intent(this, TargetActivity.class);

i.putExtra(“Key1”, “Tutorialsduniya”);

startActivity(i);
```

2. Implicit Intent: It does not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it.

Example, if we want to show the user a location on a map, we can use an implicit intent to request that another capable app show a specified location on a map.

```
Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(https://www.tutorialsduniya.com));

startActivity(i);
```

Q4. What is method overriding in OOPs ? Can we make a class both final and abstract at the same time ? Why ? [5 marks]

Ans: If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in Java**. In other words, if a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

It is used to provide the specific implementation of a method which is already provided by its superclass.

Example:

```
class Vehicle{

        void run(){System.out.println("Vehicle is running");}

}

class Bike extends Vehicle{

        public static void main(String args[]){

                Bike obj = new Bike();

                obj.run();
```

```
  }
}
```

No, we cannot make an abstract class final in Java because abstract and final are the *mutual exclusive* concept. An abstract class is incomplete and can only be instantiated by extending a concrete class and implementing all abstract methods, while a final class is considered as complete and cannot be extended further. This means when you make an abstract class final, it cannot be extended hence it cannot be used and that's why Java compiler throws a compile time error when you try to make an abstract class final in Java.

Q5. Explain the types of Layout supported by android system. What is the difference between Linear Layout and Relative Layout ? Give example. [5 marks]

Ans: Layout is the visual structure for a user interface, such as UI for an activity.

There are 6 types of layouts supported by android system,

a. Linear Layout: This layout is a viewgroup that aligns all children in a single direction, vertically or horizontally.

b. Relative Layout: This layout is a viewgroup that displays all the child views in relative positions.

c. Table Layout: This layout is a view that groups views into rows and columns.

d. Absolute Layout: This layout enables us to specify exact location of its children.

e. Grid Layout: This layout places its children in a rectangular *grid*.

f. List Layout: It is a viewgroup that displays a list of scrollable items.

Linear Layout	Relative Layout
This layout is a viewgroup that aligns all children in a single direction, vertically or horizontally.	This layout is a viewgroup that displays all the child views in relative positions.
A vertical LinearLayout will only have one child per row and a horizontal LinearLayout will only have one single row of elements on the screen.	A RelativeLayout can have any number of elements in the row as well as in the column.

Q6. Explain how you can connect to SQLite database ? How can we load data from a file to SQLite database ? [5 marks]

Ans: SQLite is a opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation.

The main package is **android.database.sqlite** that contains the classes to manage your own databases.

In order to create a database we need to call **openOrCreateDatabase** method with our database name and mode as a parameter. It returns an instance of SQLite database which we have to receive in our own object. Its **syntax** is as follows:

SQLiteDatabase myDB = openOrCreateDatabase("DBname", MODE_PRIVATE, null);

Example:

SQLiteDatabase TD = openOrCreateDatabase("TutorialsDuniya", MODE_PRIVATE, null);

We can **load** our data into the database by passing a ContentValues object to the insert() method.

Example:

SQLiteDatabase TD = mDbHelper.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(FeedEntry.COLUMN_NAME_TITLE, title);

values.put(FeedEntry.COLUMN_NAME_TITLE, subtitle);

long newRowId = db.insert(FeedEntry.TABLE_NAME, null, values);