1) **Explain following terms in detail**
   a. **AVD**
   b. **DDMS**
   c. **ADB**
   d. **Context**
   e. **Activity**
   f. **Intent**
   g. **Service**

AVD

- An Android Virtual Device (AVD) is an emulator configuration that lets you model an actual device by defining hardware and software options to be emulated by the Android Emulator.

  An AVD consists of:

- **A hardware profile:** Defines the hardware features of the virtual device. For example, you can define whether the device has a camera, whether it uses a physical QWERTY keyboard or a dialing pad, how much memory it has, and so on.
- **A mapping to a system image:** You can define what version of the Android platform will run on the virtual device. You can choose a version of the standard Android platform or the system image packaged with an SDK add-on.
- **Other options:** You can specify the emulator skin you want to use with the AVD, which lets you control the screen dimensions, appearance, and so on. You can also specify the emulated SD card to use with the AVD.
- **A dedicated storage area on your development machine:** the device's user data (installed applications, settings, and so on) and emulated SD card are stored in this area.

DDMS

- Android ships with a debugging tool called the Dalvik Debug Monitor Server (DDMS), which provides port-forwarding services, screen capture on the device, thread and heap information on the device, logcat, process, and radio state information, incoming call and SMS spoofing, location data spoofing, and more.

ADB

- Android Debug Bridge (adb) is a versatile command line tool that lets you communicate with an emulator instance or connected Android-powered device. It is a client-server program that includes three components:
- A client, which runs on your development machine. You can invoke a client from a shell by issuing an adb command. Other Android tools such as the ADT plugin and DDMS also create adb clients.
- A server, which runs as a background process on your development machine. The server manages communication between the client and the adb daemon running on an emulator or device.
- A daemon, which runs as a background process on each emulator or device instance.

Context

- Interface to global information about an application environment.
- It allows access to application-specific resources and classes, as well as up-calls for application-level operations such as launching activities, broadcasting and receiving intents, etc.

Activity

- An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with setContentView(View).

Intent

- An 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 interestedBroadcastReceiver components, and startService(Intent) or bindService(Intent, ServiceConnection, int) to communicate with a background Service.
- An Intent provides a facility for performing late runtime binding between the code in different applications.
- Its most significant use is in the launching of activities, where it can be thought of as the glue between activities.
- It is basically a passive data structure holding an abstract description of an action to be performed.

Service

- A Service is an application component representing either an application's desire to perform a longer-running operation while not interacting with the user or to supply functionality for other applications to use.

- Each service class must have a corresponding <service> declaration in its package's AndroidManifest.xml. Services can be started withContext.startService() and Context.bindService().

---

2) Explain menus provided by android. Explain with example. (Unit :- 2, Chapter :- 6)

Android supports → Options Menu, Context Menu.

Options Menu

- Contains primary functionality that applies globally to the current activity or starts a related activity. It is typically invoked by a user pressing a hard button, often labeled MENU.
- User presses the MENU button to access the Options menu.
- To create options menu:
  - Override onCreateOptionsMenu(Menu menu) method.
  - Use add() method of Menu instance with 4 arguments, where 1st arg. Is group id, 2nd arg. Is item id, 3rd arg. Is order, 4th arg. Is title.
  - To handle click event of menu item override onOptionsItemSelected(MenuItem item) method.

Context Menu

- Contains secondary functionality for the currently selected item. It is typically invoked by a user's touch & hold on an item. Like on the Options menu, the operation can run either in the current or another activity.
- A Context menu is a list of menu items (commands) that can operate on the selected content. The command can either be part of the current activity, or the system can pass the selected content along to an operation in another activity (by way of an intent).
- To create a context menu:
  - Override **onCreateContextMenu**(ContextMenu menu,View v,ContextMenuInfo info) method.
  - To register context menu with any of the widget use **registerForContextMenu()** method with one argument for the instance with whom context menu needs to be registered.
  - Use **add()** method of ContextMenu instance to add items to context menu.
  - To handle click event for context menu item override **onContextItemSelected(MenuItem item).**

3) Explain process of creating & applying style and theme to android application. (Unit :-2, Chapter :- 7) (Ref. Pg. no.168)

Creating & Applying Style

- **Purpose** → To group common View attribute values.
- Style can be applied to individual view controls.
- Style can be created generally in xml way
- Defined in **/res/values/styles.xml**
- **<resource>** parent tag**, <style>** sub tag**, <item>** sub tag.
- Example.
- To apply style to any widget use style attribute with value "@style/test" where test is the name of style defined in xml file.

Creating & Applying Themes

- **Definition** → collection of one or more styles.
- Theme is not applied to specific control, rather it is applied to all view of specific activity.
- Use **setTheme()** method to apply theme to activity.
- To define theme in androidmanifest.xml file use **android:theme** attribute of activity tag.

---

4) List and explain frame-by-frame and tweened animation supported by android. (Unit :- 3, Chapter :- 9)

Frame-by-frame animation (Ref. pg.no. 223)

- Images with minor differences are rotated fast to feel as animation.
- Best used for complicated graphics transformations that are not easily implemented programmatically.
- Create instance of **BitmapDrawable** and get resources from resources folder (All bmp / jpg files)
- Create instance of **AnimationDrawable** and use **addFrame()** method with 2 argument, where 1st arg. Is BitmapDrawable instance & 2nd arg. Is duration.
- Set AnimationDrawable object as backround to imageview using **setBackgroundDrawable()** method.
- Start the animation using **start()** method of AnimationDrawable.

Tweened Animation

- Developer can provide single drawable resource which may be (Bitmap graphics, ShapeDrawable or any other view.)
- Tweened animation can be define both programmatically and as xml resource.
    - Programmatically→ Different transformation available in android.view.animation package, for rotation animation use RotateAnimation class with appropriate arguments, use setDuration() method of RotateAnimation class to define duration of animation.
    - XML Way → Define animation as resource in /res/anim/spin.xml and <set> is a parent tag, <rotate> / <scale> etc. can be sub tag for various transformation, use android:duration attribute to define duration of animation.

---

5) What are application preferences? Explain storing & retrieving data using private & shared preferences. (Unit : - 4, Chapter :- 10) (Ref. pg. no. 231)

Application Preferences

- Light weight data storage mechanism.
- To store information such as application state, simple user information, configuration options and other small information.
- Preference data can be shared across activities of package, not outside the package.
- Preference values are stored as kev/value pair and allows to store Boolean, Float, Integer, Long & String values.
- SharedPreferences interface of android.content package
- Four steps to store data in preference
    - Retrieve instance of SharedPreference,
    - Modify content using SharedPreference.Editor
    - Do changes to the editor
    - Commit changes.

Creating Private Preference

- Retrieve activity's private preference

```
SharedPreferences settingsActivity = getPreferences(MODE_PRIVATE);
```

Creating Shared Preference

- Retrieve preference, provide name to preference, this name can be used in any activity to retrieve value.

```
sharedPreferences settings = getSharedPreferences("MyCustomSharedPreferences",
0);
```

Adding, updating and deleting data to and from preference

- For performing any above operation create SharedPreference.Editor and use various methods such as putLong(), putString(), putInt() to store values into it.

```
SharedPreferences settingsActivity = getPreferences(MODE_PRIVATE);
SharedPreferences.Editor prefEditor = settingsActivity.edit();
prefEditor.putLong("SomeLong", java.lang.Long.MIN_VALUE);
```

Commit changes

```
prefEditor.commit();
```

- To remove data use
  `prefEditor.remove();`

Reading Preference

- getBoolean, getInt(), getString() choose appropriate method of SharedPreference interface to retrieve values.

---

6) Write an android application to insert new contact, allow user to update that contact & delete contact. (Unit :-3, Chapter :- 11) (Ref.pg. no. 267)

Insert new contact

- Create instance of ContentValues.
- Use put method to put key/value pair for new contact.
- Create uri instance to locate default contacts along with value stored inside ContentValues.
- Create uri instance to make new uri that points to this newly created contact.
- Use ContentValues instance to insert further information for new contact using newly created uri.
- Use insert() method of getContentResolver() to insert new contact.

Updating contact

- Create new uri to point to contact that needs to be updated.
- Create instance of ContentValues & use put() method for the columns where data needs to be updated.
- Use update() method of getContentResolver() to update contact.

Deleting Contact

- Use delete() method of getContentResolver() to delete contact.

7) Write an android application to fetch images stored in sdcard and display it in the gallery. (Unit :- 3, Chapter :- 10)

- Use MediaStore.Images.Media.EXTERNAL_CONTENT_URI as 1st arg. to managedQuery() method to retrieve all images from sdcard.
- Create ImageAdapter that extends BaseAdapter to bind images to sdcard.
- Implement getCount(), getItem(), getItemId(), getItem() & getView() method for ImageAdapter class.
- Code written for getView() method will help to display image to gallery.
- Create ImageView instance and display image of sdcard to it, use setImageURI() method of ImageView as argument that defined that image comes from sdcard.
- These ImageView will be repeated in gallery for no. of images.
- Use setAdapter() method of gallery to point to new ImageAdapter class just created.

---

8) Write android application to parse given xml file located on internet. (Unit :-3, Chapter :- 12) (Ref.pg. no. 290)

To parse xml file

- Created URL instance with address of url as argument where xml file is located.
- Create instance of XmlPullParserFactory using newInstance() method.
- Create instance of XmlPullParser() class using newPullParser() method.
- Use setInput() method of XmlPullParser class to open stream.
- Use getEventType() method to get event type, Event type can be START_TAG, END_TAG.
- Loop until the end of xml document and search for starting tag, to retrieve value for attribute getAttributeValue() method.
- Use next() method to read next tag.

---

9) What is AsyncTask? How it is useful to manage heavy task as background process? Explain with example. (Unit :- 3, Chapter :- 13) (Ref. pg. no. 292)

AsyncTask

- Abstarct helper class for managing background activities.

- Implement onPreExecute(), doInBackground(), onPostExecute(), onProgressUpdate(), onCancelled().
- To manage heavy task as background process write the code in doInBackground() method, create a separate thread and do the processing in this thread, which is separate then UI thread.
- To inform about progress to UI thread publishProgress() method periodically.

---

10) List out features of Android platform. Explain in brief these features.

Features of android platform

- Next generation platform for developing application for handheld device.
- Free & open source.
- Familiar & inexpensive development tools.
- Powerful library.
- Rich, secure application integration.
- Easy to publish.
- Free market for publishing application to customer.
- Growing platform, new feature are added with every new version.