Contents

UNIT 3 - Android Graphics, Databases, and Messaging

CHAPTER 7 – Graphics, Animations, and Multimedia

Structure of Graphics, Animations, and Multimedia

7.1 Learning Outcomes

After the successful completion of this chapter, the student will be able to:

- Describe various graphics and animation techniques in android
- Demonstrate the use of graphics and animations in android with suitable examples
- Describe various persistent data storage concepts used in android

7.2 Graphics

Some android applications make use of graphics and animations. Different techniques are used for drawing graphics depending upon the type of application. For example, the graphics technology used in gaming applications is different from that used in an application that requires the use of static graphics.

Android provides a 2D and 3D graphics library which contains many APIs which are used for drawing graphics in android applications. Android provides android.graphics package which contains the classes required to create graphics in android applications.

Graphics in android can be created either using View object or Canvas.

There are two ways to draw graphics when Canvas is used

- Using View Class
- Using SurfaceView Class

Methods provided by the Canvas class to draw various shapes are

- drawArc()
- drawRect()
- drawCircle()
- drawBitmap()
- drawText() etc

The canvas serves as a surface on which the rectangle and circle are drawn. The onDraw() method is invoked by the android framework. Within the onDraw() method we can invoke various methods like drawRect(), and drawCircle() to draw rectangles and circles on the canvas.

An example of drawing graphics using a customized view class is given here.

```
public class MainActivity extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
         setContentView(new MyCustomView(this));
    }


    public class MyCustomView extends View {
        private Paint paint;
        public MyCustomView(Context context) {
            super(context);
            // create the Paint and set its color
        }


        @Override
        protected void onDraw(Canvas canvas) {
            paint = new Paint();
            canvas.drawColor(Color.LTGRAY);
            paint.setColor(Color.RED);
```

```
            paint.setTextSize(80);
            paint.setTypeface(Typeface.defaultFromStyle(Typeface.BOLD));

            canvas.drawText("Circle",200,150,paint);
            canvas.drawCircle(300, 500, 250, paint);

            paint.setColor(Color.BLUE);
            canvas.drawText("Rectangle",650,150,paint);
            canvas.drawRect(700, 200, 1000, 800, paint);
        }
    }
}
```

Figure 7.1: Snippet of MainActivity.java for drawing of circle and rectangle

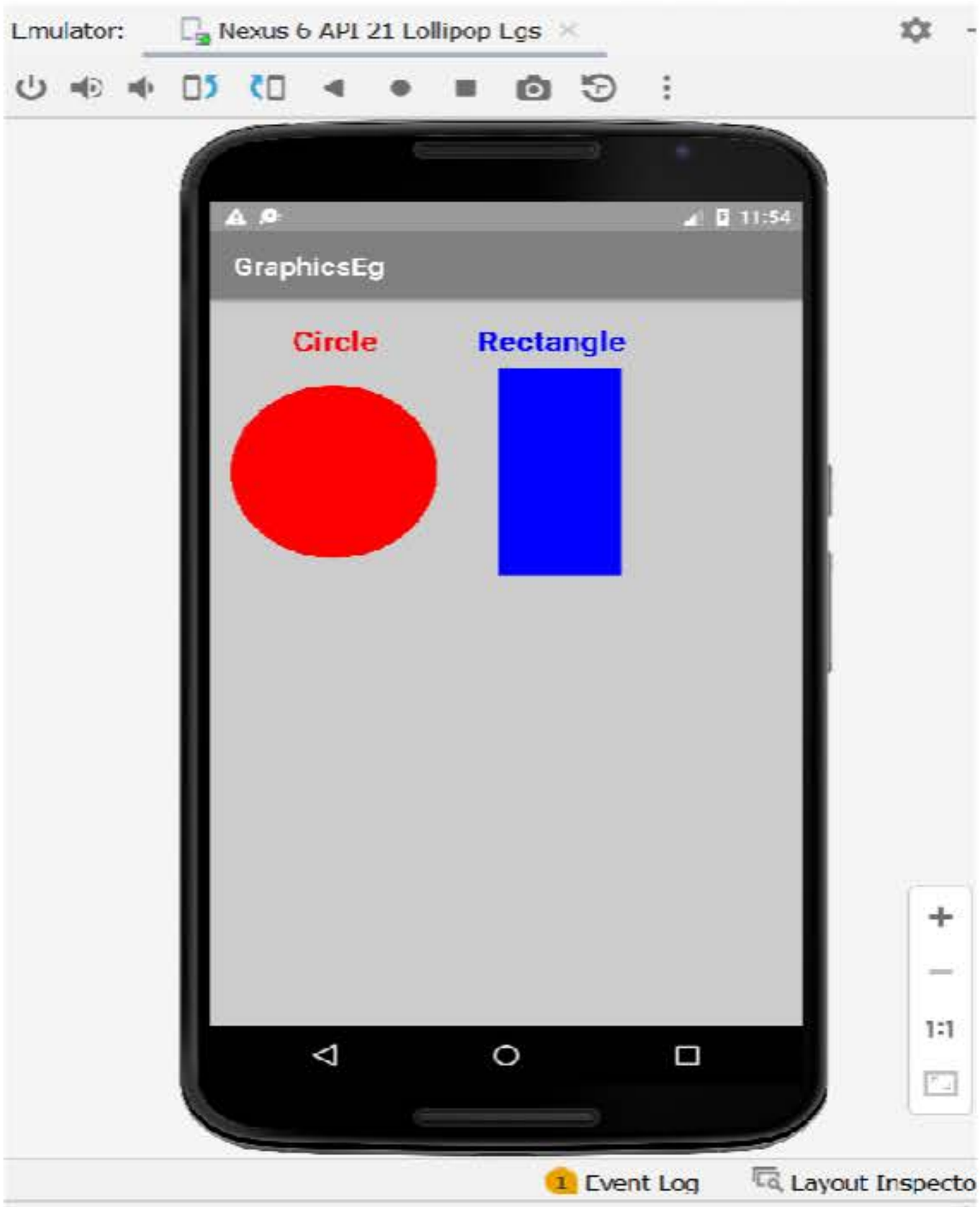

Figure 7.2: Display of Graphics

7.3 Animations

In Android android.view.animation package provides various classes and interfaces which allow implementing animation effects on views such as rotate, scale, fade, move, etc.

There are 3 types of animations mainly used in android

- Property Animation - which allows animating properties of view or non-view objects
- View Animation – used to animate view contents. For e.g. re-sizing, rotation, etc.
- Drawable Animation – used to display a sequence of drawable items/images inside a view object

When using View Animation either an xml file is used for specifying the transformation or hardcode it in the MainActivity.java file. Using xml file is more beneficial as it is more readable and can be re-used by another activity. The XML file should be present in the res/anim folder. A single root element like <set>, <translate>, <scale>, etc .should be used in an XML animation file. The <set> element can embed other elements within it.

Example: Rotating Image using View Animation is given here

Create a project and add an ImageView in the activity_main.xml file as shown below

```
<ImageView
    android:id="@+id/GRElogo1"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_marginTop="200dp"
    android:layout_marginLeft="100dp"
    android:scaleType="fitCenter"
    android:src="@drawable/grelogo"
    android:background="@color/white"/>
```

Figure 7.3: ImageView in activity_main.xml layout file

Create an animation layout file with the following code for rotating the image

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <rotate
        android:fromDegrees="0"
        android:toDegrees="360"
```

```
        android:duration="6000"
        android:repeatCount="infinite" />
</set>
```

Figure 7.4: Animation layout file used for rotating image

Make changes to the MainActivity.java file as shown here and then execute the application

```
public class MainActivity extends AppCompatActivity {
    ..
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        imgview = (ImageView) findViewById(R.id.GRElogo1);
        Animation animobj = AnimationUtils.loadAnimation(this,
                            R.anim.anim_image);
        imgview.startAnimation(animobj);
    }
}
```

Figure 7.5: MainActivity.java file showing code for animating image

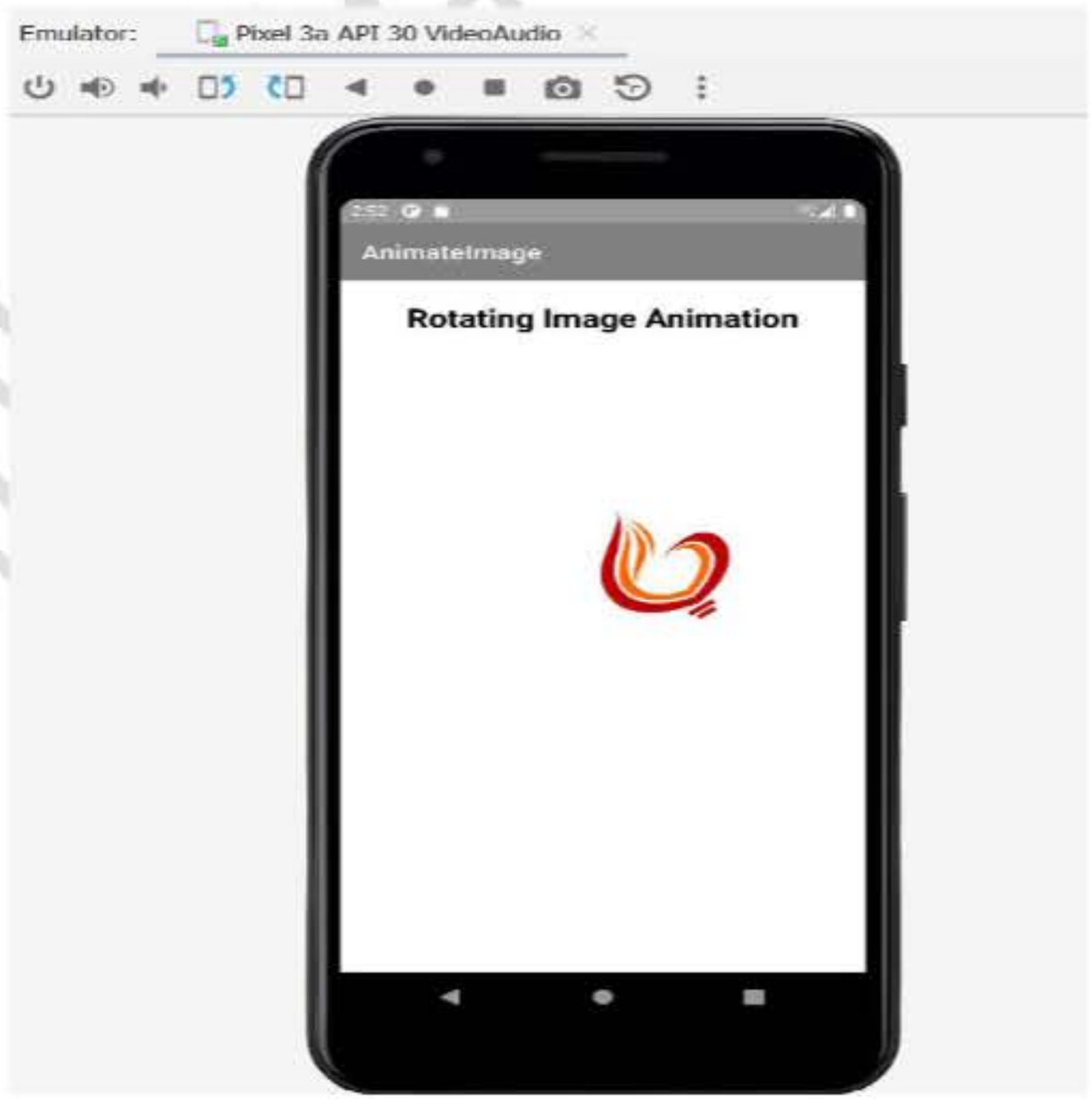

Figure 7.6: Display of Animation - rotating image

7.4 Multimedia

Android provides various media APIs that are used to play audio and video files.

Mp3 format is supported for audio files and mp4 format for video files

MediaPlayer, AudioManager, MediaRecoder.MediaPlayer, MediaController classes are available in android to support playing, and recording audio and video files

7.4.1 Playing audio files

MediaPlayer class is used in android applications to play audio files.

Example: To play an audio file on click of a button

- Create an android project, as usual, and add a button in the activity_main.xml file.
- Add the event handler to the button in the activity_main.xml file to handle the event.
- Make changes in the MainActivity.java file by adding the code to play the audio at the click of a button.
- Add the audio file to be played in the raw folder.

```
public class MainActivity extends AppCompatActivity {

    Button btnplay;
    MediaPlayer mediaplayerobj;

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

    public void playaudio(View view)
    {
       if (mediaplayerobj == null)
       {
           mediaplayerobj = MediaPlayer.create(this,R.raw.audioplaymp3eg);
           mediaplayerobj.start();
       }
    }
}
```

Figure 7.7: MainActivity.java file showing code for playing an audio file

7.4.2 Playing Video Files

MediaController class is used in android applications to play video files.

Example: To play a video file with the click of a button

- Create an android project, as usual, and add a button in the activity_main.xml file.
- Add the event handler to the button in the activity_main.xml file to handle the event.
- Make changes in the MainActivity.java file by adding the code to play the video at the click of a button.
- Add the video file to be played in the raw folder.

```
public class MainActivity extends AppCompatActivity {
..
 @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnplay = (Button) findViewById(R.id.btnplay);
        video1 = (VideoView) findViewById(R.id.video1);
        media1 = new MediaController(this);
        media1.setAnchorView(video1);
    }

    public void playvideo(View view)
    {
        String pathvideo = "android.resource://"+ getPackageName() + "/" +
                                                R.raw.geuhospital;

        Uri uriobj = Uri.parse(pathvideo);
        video1.setVideoURI(uriobj);
        video1.setMediaController(media1);
        video1.requestFocus();
        video1.start();
    }
}
```

Figure 7.8: MainActivity.java file showing code for playing a video file

OnClick of the button the video starts playing. There are also backward and forward buttons added to the video by using the requestFocus() method.

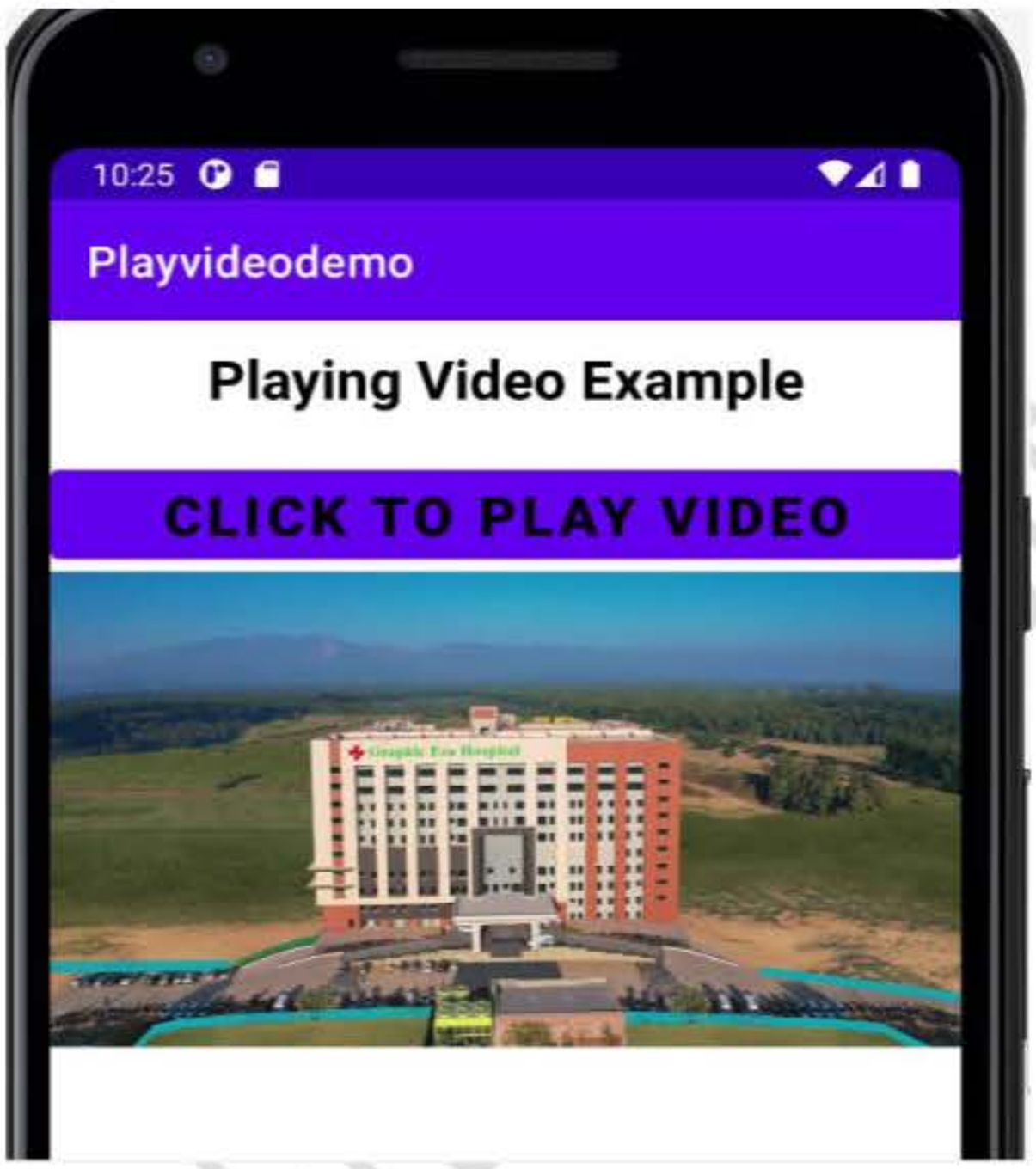

Figure 7.9: Display of playing a video file in an Android project

7.5 Storage of data

It is sometimes required to store the application's data permanently on a device for later use until it is no longer required and needs to be deleted.

Android Application's data can be stored using any of these storage options

- Shared Preferences
- Internal Storage
- External Storage
- SQLite Databases
- Network Connection

The storage option to choose depends on the kind/type of data, and the amount of data. i.e., the space required to store the data, whether the data is private or public, with whom to share, the kind to access to be given to other users, whether the data is to be shared by other applications, etc. Depending upon the user's requirement the storage is to be decided.

7.5.1 Shared Preferences

When the amount of data to store is small/less i.e. less than 100KB and is of primitive type then shared preference is used. The data is stored in the form of key-value pair in an XML file on the user's device. The data is stored in private mode if it is to be used by the application only. If the data is to be shared across other applications then it is stored in public mode. Shared preference stored data is available on the device until it is manually deleted or the application is uninstalled.

7.5.2 Files – Internal Storage

When an android application is installed on a device, the android operating system allocates some internal storage where the private data of the application is stored. Data stored here is not accessible by other applications and is removed when the application is uninstalled. When data is to be stored in files and private internal storage is used.

Some useful methods commonly used for files are

getFilesDir() - To get the absolute path of the directory in the file system
getDir() - To create or open an existing directory
deletefile() - To delete a file
fileList() - It returns a list of all files saved by the application in an array

Sample code to create and write to a file in android

```
String filename = "mydatafile";
String message  = "Welcome to internal storage learning!";
FileOutputStream fileOS = openFileOutput(FILENAME, Context.MODE_PRIVATE);
fileOS.write(message.getBytes());
fileOS.close();
```

Figure 7.10: Snippet for creating and writing in a file in Android

Sample code to read from a file in display the contents read by using Toast

```
String filename = "mydatafile";
FileInputStream fileIS = openFileInput(filename);
int n;
StringBuilder sbobj = new StringBuilder();
```

```
//reading from a file
while ((n = fileIS.read()) != -1)
{
    // appending to StringBuilder object
     sbobj.append((char) n);
}
fileIS.close();

//Using Toast to display the contents of the object
Toast.makeText(getBaseContext(), sbobj.toString(),
                                 Toast.LENGTH_LONG).show();
```

Figure 7.11: Snippet for reading from a file and displaying the contents using Toast in Android

7.5.3 External Storage

Data stored on files using external storage like an SD card is accessible to everyone. Files stored on external storage are accessible even to other apps and don't get deleted even if the app is uninstalled. The files have public access. The user should obtain READ EXTERNAL STORAGE permission to access the external storage data. So any application which has permission can use the data. When a large amount of data is to be stored then external storage should be used. For e.g. video, audio files, etc. Files stored on an external device may not be available always to the application as the user may remove the storage (SD card) at any time. The files will not be deleted unless the user does so.

7.5.4 SQLite database

When data to be stored is in a structured form then databases are used. Android supports SQLite database. The database created by SQLite is specific and accessible only within the application and cannot be used outside it. When data is large and structured then the database is used for storing. Using query language all operations can be performed on data like retrieving, searching, updating, deleting, etc.

7.5.5 Network Connection

When the network is available, it can be used to store and retrieve data.

The packages java.net.*, android.net.* have classes that are used to perform network operations.

7.6 Self-Assessment Questions

Q1. Illustrate the use of graphics in android with a suitable example.[8 marks, L3]

Q2. Illustrate the use of animation in android with a suitable example.[8 marks, L3]

Q3. Demonstrate how to play an audio file with a click of a button and with a suitable code. [8 marks, L3]

Q4. Demonstrate how to play a video file with a click of a button and with a suitable code. [8 marks, L3]

Q5. Discuss how android application data is stored with suitable examples.[8 marks, L2]

Q6. Explain the following storage options in android to store persistence data. [8 marks, L2]

- Shared Preferences
- Internal Storage
- External Storage
- SQLite Databases
- Network Connection

7.7 Self-Assessment Activities

Note: Use an appropriate emulator for the execution of an android application.

A1. Create an Android application, and add appropriate controls and graphics code so that on the execution of the application, the output should be as shown in figure 7.2.

A2. Create an Android application, and add appropriate controls and animation code so that on the execution of the application, the output should be as shown in figure 7.6.

A3. Create an Android application, and add appropriate controls and multimedia code so that during execution with the click of a button an audio file is played.

A4. Create an Android application, and add appropriate controls and multimedia code so that during execution with the click of a button, a video file is played.

7.8 Multiple-Choice Questions

1. The 2 ways to draw graphics using canvas in android are by using_____. [1 mark, L1]
   a) View Class and SurfaceView class
   b) View Class and Canvas Class
   c) Canvas class and SurfaceView Class
   d) None of the above

2. The method invoked by the android framework to draw the graphics is ____. [1 mark, L1]

a) onCreate()
b) onPaint()
c) onDraw()
d) None of the above

3. _____ is not a method of Canvas Class. [1 mark, L1]
   a) drawCircle()
   b) onDraw()
   c) drawRect()
   d) drawText()

4. XML Animation file should be placed in the folder _____. [1 mark, L1]
   a) res/raw
   b) res/layout
   c) res/menu
   d) res/anim

5. The class that is used in android to play an audio file is ____. [1 mark, L1]
   a) MediaPlayer
   b) AudioPlayer
   c) MultimediaPlayer
   d) None of the above

6. The class that is used in android to play video files is _____. [1 mark, L1]
   a) VideoPlayer
   b) MediaController
   c) VideoController
   d) MultimediaPlayer

7. When the amount of data to be stored is less than 100KB, the storage used in Android is ____. [1 mark, L1]
   a) SQLite storage
   b) File storage
   c) Shared preference
   d) None of the above

8. SQLite database data is accessible ____. [1 mark, L1]
   a) from anywhere, even by other applications
   b) from only within the application and cannot be used outside it
   c) as shared data even outside the device
   d) None of the above

9. Data stored on files using external storage like an SD card is accessible ___. [1 mark, L1]
   a) to even other android applications with read external storage permission
   b) Only by the application which created it
   c) to other applications without requiring any permissions
   d) None of the above

10. Data stored on files is not accessible by other applications and is ____. [1 mark, L1]
    a) removed when the application stops executing
    b) removed when the application is uninstalled
    c) not removable

7.9 Key Answers to multiple-choice questions

1. The 2 ways to draw graphics using canvas in android are by using View Class and SurfaceView class.[a]
2. The method invoked by the android framework to draw the graphics is onDraw().[c]
3. onDraw() is not a method of Canvas Class.[b]
4. XML Animation file should be placed in the folder res/anim.[d]
5. The class that is used in android to play an audio file is MediaPlayer.[a]
6. The class that is used in android to play video files is MediaController.[b]
7. When the amount of data to be stored is less than 100KB, the storage used in Android is Shared preference.[c]
8. SQLite database data is accessible from only within the application and cannot be used outside it. [b]
9. Data stored on files using external storage like an SD card is accessible to even other android applications with read external storage permission.[a]
10. Data stored on files is not accessible by other applications and is removed when the application is uninstalled.[b]

7.10 Summary

Some android applications make use of graphics and animations. Different techniques are used for drawing graphics depending upon the type of application. For example, the graphics technology used in gaming applications is different from that used in an application that requires the use of static graphics.

Android provides 2D and 3D graphics library which contains many APIs which are used for drawing graphics in android applications. Android provides android.graphics package which contains the classes required to create graphics in android applications.

Graphics in android can be created either using View object or Canvas.

There are two ways to draw graphics when Canvas is used

- Using View Class
- Using SurfaceView Class

In Android android.view.animation package provides various classes and interfaces which allow implementing animation effects on views such as rotate, scale, fade, move, etc.

There are 3 types of animations mainly used in android

- Property Animation - which allows animating properties of view or non-view objects
- View Animation – used to animate view contents. For e.g. re-sizing, rotation, etc.
- Drawable Animation – used to display a sequence of drawable items/images inside a view object

Android provides various media APIs that are used to play audio and video files.

Mp3 format is supported for audio files and mp4 format for video files

MediaPlayer, AudioManager, MediaRecoder.MediaPlayer, MediaController classes are available in android to support playing, and recording audio and video files

MediaPlayer class is used in android applications to play audio files.

MediaController class is used in android applications to play video files.

It is sometimes required to store the application's data permanently on a device for later use until it is no longer required and needs to be deleted.

Android Application's data can be stored using any of these storage options - Shared Preferences, Internal Storage, External Storage, SQLite Databases and Network Connection.

7.11 Keywords

- Graphics
- Animations
- Multimedia
- Shared Preferences

- Internal Storage
- External Storage
- SQLite Databases
- Network Connection.

7.12 Recommended Resources for Further Reading

Text Book(s)

1. Pradeep Kothari, "Android Application Development" Black-Book, Dreamtech Press,2015
2. Wei-Meng Lee, "Beginning Android Application Development", Wiley 2011.
3. Dawn Griffiths and David Griffiths, "Head First Android Development", O'Reilly, 2017.

References

- https://www.javatpoint.com/android-tutorial
- https://www.tutorialspoint.com/android/index.htm
- https://developer.android.com/guide/components/fundamentals

--*--

Contents

UNIT 3 - Android Graphics, databases, and messaging

CHAPTER 8 – SQLite Database

Structure of SQLite Database

8.1 Learning Outcomes:

After the successful completion of this chapter, the student will be able to:

- Explain the process of creating SQLite database and tables in android
- Illustrate the use of Database Helper Class
- Explain the process of retrieving data from the database and displaying in it RecyclerView
- Develop an android application for creating a database, storing information in it, retrieving the same, and displaying it on the screen in a scrollable manner using RecyclerView.
- Develop an android application for sending SMS and email.

8.2 SQLite Database

Android SQLite database is an open-source, lightweight database available on all android devices. SQLite supports the features of a relational database where data is stored in the form of tables, but on android devices, data is stored in the form of text files. Operations like add, delete, update, and retrieve can be performed on the data. In-built user interface tools for creating SQLite databases are not available in Android. SQLite database does not

support stored procedures. To create a database in android SQLiteOpenHelper class is used.

Advantages of SQLite database

- SQLite is serverless, which means it does not require a server to run
- SQLite is lightweight and can be used in devices like mobile, TV, camera, etc.
- It has better performance as read/write operations are faster
- No installation is required
- It requires less memory
- There are no external dependencies and is self-contained
- It supports almost all operating system

Disadvantages of SQLite database

- Database size is less than 400KB
- The database can be used only on the device on which it is stored.

8.2.1 SQLiteOpenHelper and Database Helper

SQLiteOpenHelper Class - This class is available in android.database.sqlite.SQLite database package. All operations like creating databases and tables, handling database operations like add, delete, update, retrieve, etc., and database versions are managed by SQLiteOpenHelper class.

SQLiteOpenHelper class contains two methods onCreate() and onUpgrade() which have to be overridden by the helper class. SQLiteDatabase class also contains methods to execute SQL commands.

Database Helper :

For working with SQLite database we need to create our own class usually called a Utility or Helper class which extends/inherits from the in-built SQLiteOpenHelper class. In this class, we override the methods onCreate() and onUpgrade() of SQLiteOpenHelper.

onCreate() – In the lifecycle of an application this method is called only once. It is called when a database is created for the first time. So one-time operations like creating tables are performed in this method.

onUpgrade() – Database and table upgradation are usually done in this method. For e.g. dropping a table, altering a table, etc.

Example: Here a user-defined DBHandlerProducts class is created by extending from the in-built SQLiteOpenHelper class.

In the constructor `DBHandlerProducts()`, a call to the superclass creates the Database "**GEUSHOPPING.db**"

In the `onCreate()` method the table **"ProductsInfo"** is created by executing the query.

The `onUpgrade()` contains a call to the `onCreate()` method in case tables have to be re-created or altered.

```
package com.example.geushopapp;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
public class DBHandlerProducts extends SQLiteOpenHelper {
    Context context;
    public static String  DATABASE_NAME = "GEUSHOPPING.db";
    private static final int DATABASE_VERSION = 1;
    private static final String TABLE_NAME = "ProductsInfo";
    public DBHandlerProducts(Context context) {
        super(context, DATABASE_NAME, null,DATABASE_VERSION);
        this.context=context;
    }
    @Override
    public void onCreate(SQLiteDatabase sqlitedb) {
    try {
        String query  = "Create table if not exists " + TABLE_NAME +
                        "(PRODIMGID INTEGER PRIMARY KEY AUTOINCREMENT,
                         PRODNAME Text, PRODIMG BLOB,PRODPRICE Real)";
        sqlitedb.execSQL(query);
        Toast.makeText(context,"ProductsInfo table successfully created
                       inside the database", Toast.LENGTH_SHORT).show();
        }
        catch (Exception e)
        {
         Toast.makeText(context,e.getMessage(),Toast.LENGTH_SHORT).show();
        }
    }
    @Override
    public void onUpgrade(SQLiteDatabase sqlitedb, int i, int i1) {
        onCreate(sqlitedb);
    }
```

Figure 8.1: Snippet of Database Helper class showing code for creating database and table.

8.3 Recycler-View

Recycler-view is a control/widget which is used to efficiently display large datasets in a flexible and scrollable manner. Only a few items/views which fit on the screen are displayed and are visible while other items can be made visible by scrolling. It acts as a container and is considered an advanced version of ListView and GridView.

Before using Recycler-view in the project, update the file build.gradle to add the support library dependency as follows

```
dependencies {
 // support library dependency version has to be matched depending on the version
 // of Android Studio installed,
    compile 'com.android.support:recyclerview-v7:25.3.1'
}
```

Figure 8.2: Support library dependency code for RecyclerView

Components of Recycler-View

- Layout Managers
- View Holder
- RecyclerView.Adapter
- Model Class

8.3.1 Layout Managers

RecyclerView has 3 types of Layout Managers

- LinearLayoutManager – Data items are displayed either vertically or horizontally.
- GridLayoutManager – Data items are displayed in Grid format (rows and columns).
- StaggeredGridLayoutManager – Data items are displayed in the staggered Grid.

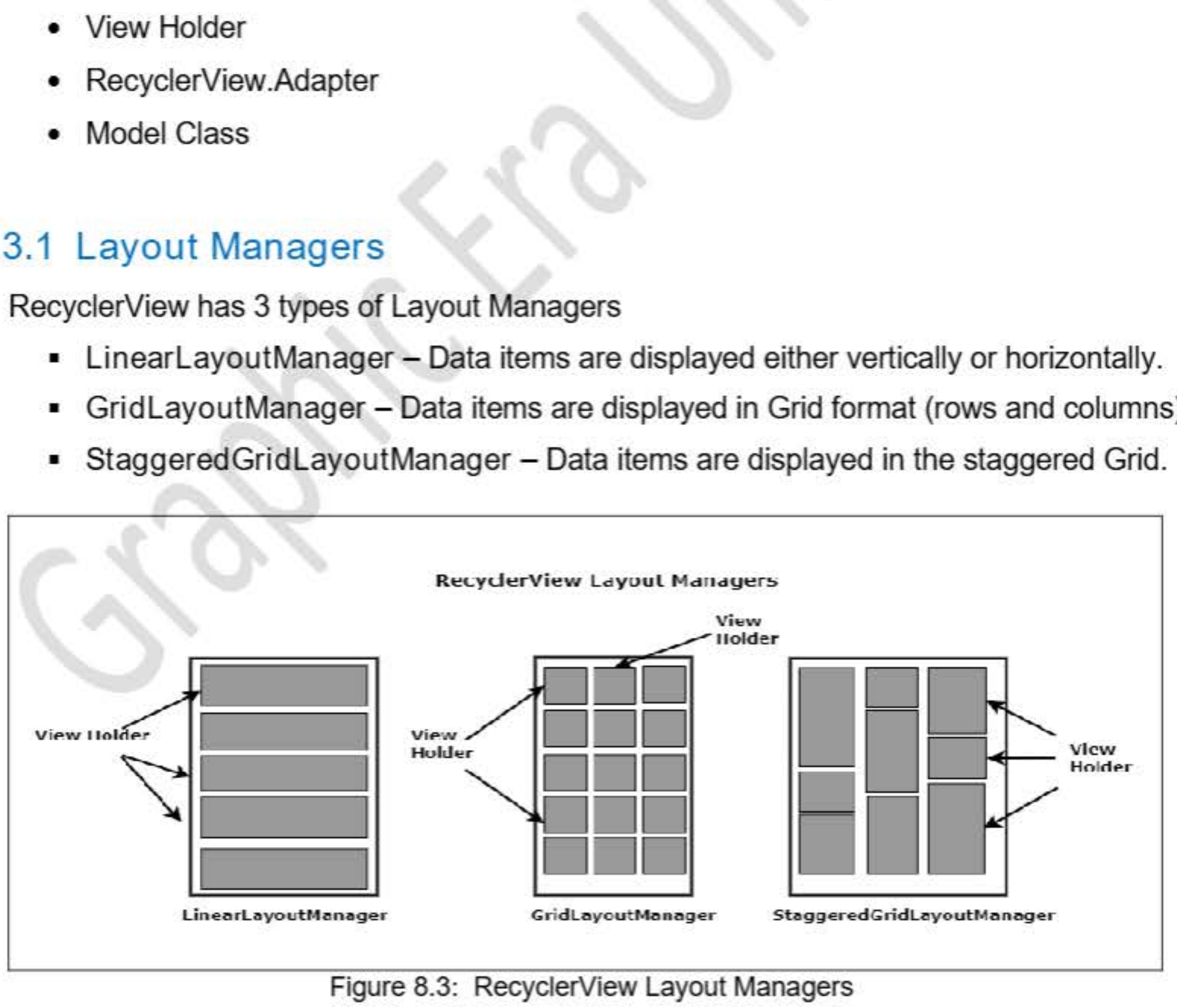

Figure 8.3: RecyclerView Layout Managers

8.3.2 View Holder

It represents an individual item in the ListView or GridView. It acts like a wrapper that contains other views and metadata. It is an object which contains the all information to be displayed as a single item in the list or grid of the RecyclerView. It is mandatory to use a ViewHolder when using a RecyclerView.

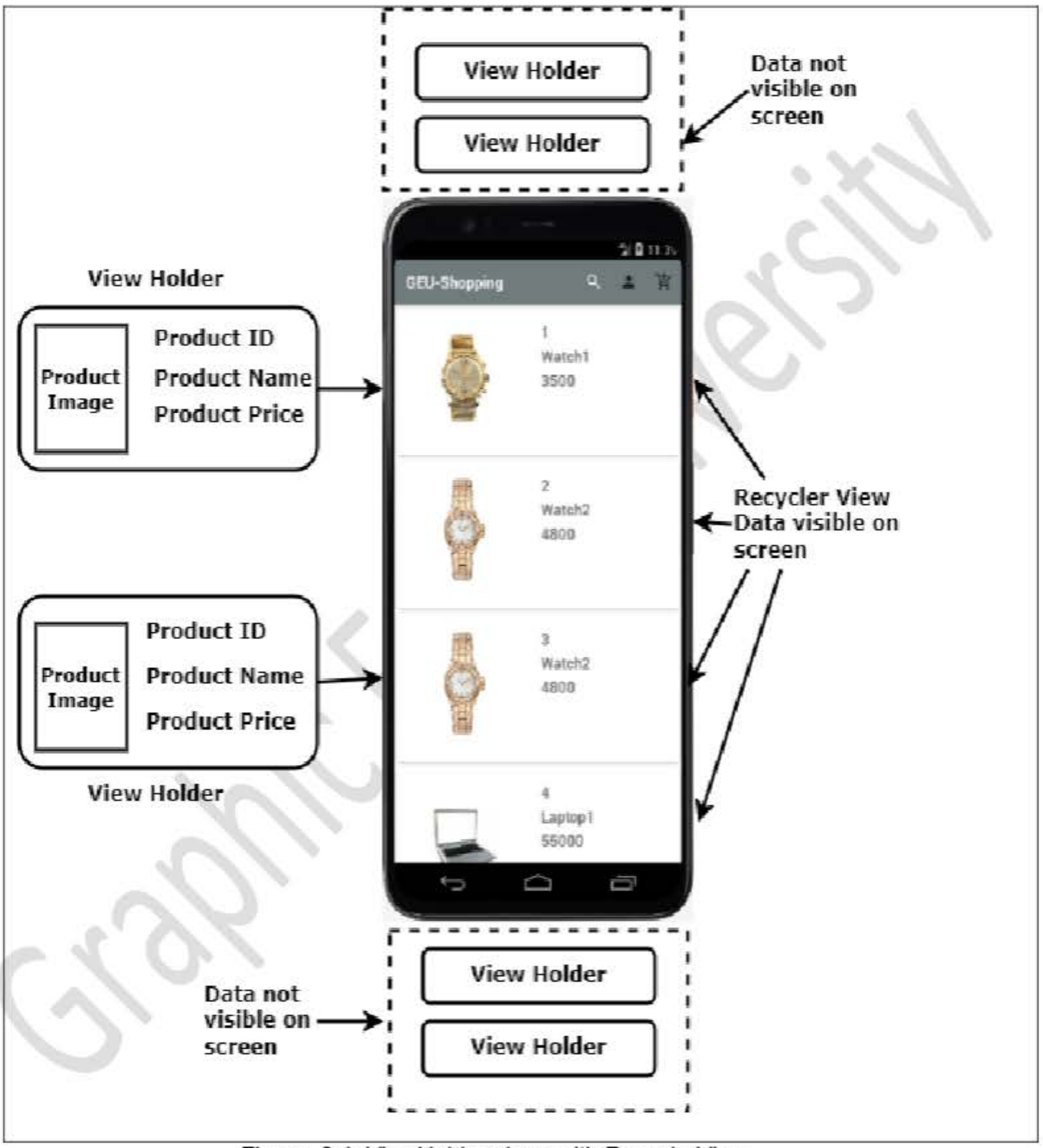

Figure 8.4: ViewHolder along with RecyclerView

For example: In figure 8.4 the view holder contains an ImageView displaying the product image (Watch, Laptop) along with TextView which displays the details of the product like product ID, product name, and product price.

8.3.3 RecyclerView Adapter

It is used to bind the data from the dataset to the ViewHolder that displays it within the RecyclerView.

It contains two methods onCreateViewHolder() and onBindViewHolder() which have to be overridden.

The onCreateViewHolder() method is used to inflate/populate the View and its ViewHolder, while the onBindViewHolder() method is used to bind data to the View.

```
public class RCVAdapter extends RecyclerView.Adapter
                                <RCVAdapter.RCViewHolder Class>
{
   Context context;
   ArrayList<ModelClass> objMCList;
   public RCVAdapter(Context context, ArrayList<ModelClass> objMCList)
  {
   {
        this.context = context;
        this.objMCList = objMCList;
    }

@NonNull
@Override
public RCViewHolderClass onCreateViewHolder(@NonNull ViewGroup parent,
                                                    int i)
{
   LayoutInflater inflater = LayoutInflater.from(parent.getContext());
   View view=inflater.inflate(R.layout.activity_single_row,parent,false);
   return new RCViewHolderClass(view);
}

@Override
public void onBindViewHolder(@NonNull RCViewHolderClass holder,
                                                int position)
 {
   ModelClass objMC = objMCList.get(position);
    holder.TVProdId.setText(objMCList.get(position).getProdId());
    holder.TVProdName.setText(objMCList.get(position).getProdName());
```

```
    holder.IVProdImg.setImageBitmap(objMCList.get(position).getProdImg());
    holder.TVProdPrice.setText(objMCList.get(position).getProdPrice());
}


@Override
public int getItemCount() {
      return objMCList.size();
}

public static class RCViewHolderClass extends RecyclerView.ViewHolder
{
     // Create View objects for reference

     public RCViewHolderClass(@NonNull View itemView)
    {
      super(itemView);

     TVProdId    = (TextView)itemView.findViewById(R.id.sr_TVProdId);
     TVProdName  = (TextView)itemView.findViewById(R.id.sr_TVProdName);
     IVProdImg   = (ImageView)itemView.findViewById(R.id.sr_IVProdImg);
     TVProdPrice = (TextView)itemView.findViewById(R.id.sr_TVProdPrice);
    }
}
```

Figure 8.5: Snippet of RecyclerView Adapter class

8.3.4 Model Class

It is a class that holds the data and business logic. Model class associate's data with view holder.

The Model class contains 3 types of methods

- Constructor: Between the adapter and the model it acts like a bridge
- Setter methods: Property values are set using these methods
- Getter methods: Property values are retrieved using these methods

For each property declared in the Model class, there should be a corresponding getter and setter method

Example of Model class for "ProductsInfo" table

```
public class ModelClass {
    private String ProdId;
    private String ProdName;
    private Bitmap ProdImg;
    private String ProdPrice;
    // constructor
    public ModelClass(String ProdId, String ProdName, Bitmap ProdImg,
                      String ProdPrice) {
        this.ProdId = ProdId ;
        this.ProdName = ProdName;
        this.ProdImg = ProdImg;
        this.ProdPrice = ProdPrice;
    }
    public String getProdId() {                    // getter method
        return ProdId;
    }
    public String getProdName() {                  // getter method
        return ProdName;
    }
    public void setProdName(String ProdName) {     // setter method
        this.ProdName = ProdName;
    }
    public Bitmap getProdImg() {
        return ProdImg;
    }
    public void setProdImg(Bitmap ProdImg) {
        this.ProdImg = ProdImg;
    }
    public String getProdPrice() {
        return ProdPrice;
    }
    public void setProdPrice(String prodPrice) {
        ProdPrice = prodPrice;
    }
}
```

Figure 8.6: Snippet of RecyclerView Adapter class with "ProductsInfo" table

For implementing a RecyclerView we need to decide the following

- How the display should look and accordingly use the appropriate layout. Items can be displayed by using ListView, GridView, etc.
- Design a View holder for placing items and providing functionality
- Design an Adapter i.e., Data Model Class that associates data with the view holder

Figure 8.7 shows how data is retrieved from the database and displayed in the RecyclerView using LinearLayoutManager.

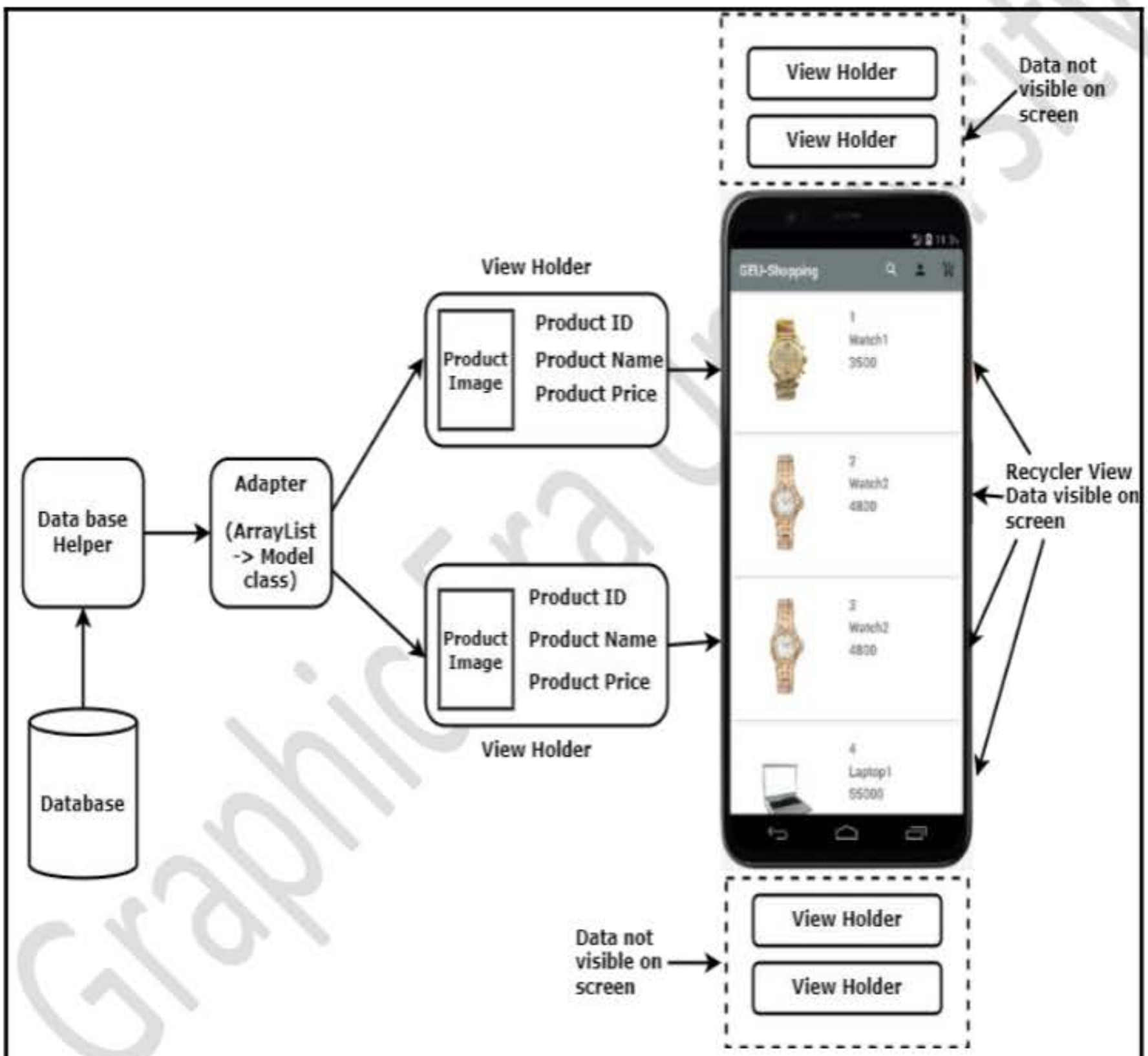

Figure 8.7: Retrieving data from the database and displaying it in RecyclerView

The data is retrieved from the database and stored in an ArrayList. The ArrayList contains objects that are to be displayed in the RecyclerView. Each item in the ArrayList is an object i.e., View Holder that represents a row in the display.

8.4 Messaging

Messaging is one of the capabilities of android that helps to communicate with the outside world. Messaging can be done either by sending an SMS or email.

8.4.1 SMS

In Android, we can send SMS in 2 ways from our application

1. Using SMS Manager API
2. Using Intent i.e., device built-in SMS application

When SmsManager API is used the SMS is sent directly from the application, whereas when Intent is used in-built SMS app is invoked and we need to use proper action (ACTION_VIEW).

In both cases before sending SMS AndroidManifest.xml file has to be updated by adding the following line of code to get permission to send SMS.

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

Figure 8.8: Setting SMS sending permission in the AndroidManifest.xml file

For receiving an SMS we need to add the following line of code in AndroidManifest.xml

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

Figure 8.9: Setting SMS receiving permission in the AndroidManifest.xml file

1. Using SMS Manager API

For sending SMS using SmsManager API

1. Create an object of SmsManager by using the getDefault() method
2. Call sendTextMessage() method.

1st parameter is the destination address i.e., the mobile number to whom the message is to be sent

2nd parameter is the service center by default it is null

3rd parameter is the message "Your bill is Rs. **3000, pay on item delivery"**

4th parameter is null

5th parameter is null

The code to send SMS using SmsManager is given below

```
String mobileno = "8888899999";
String message = " Your bill is Rs. 3000, pay on item delivery";
SmsManager smsobj = SmsManager.getDefault();
snsobj.sendTextMessage(mobileno, null, message, null, null);
```

Figure 8.10: Code implementing SmsManager API for sending SMS

2. Using Intent i.e., device built-in SMS application

For sending SMS using Intent

1. Create an object/instance of Intent by using passing Intent.ACTION_VIEW as a parameter to its constructor
2. Call the putExtra() method to set the "address", bypassing the destination mobile number as 2nd parameter
3. Call putExtra() method to set "sms_body", by passing message as 2nd parameter
4. Call the setType() method by passing "vnd.android-dir/mms-sms" as a parameter
5. Finally call the startActivity() method by passing the intent instance as a parameter

It is mandatory to use parameters "address" and "sms_body" in lowercase letters as they are case-sensitive.

The code to send SMS using Intent is given below

```
String mobileno = "8888899999";
String message = " Your bill is Rs. 3000, pay on item delivery";
Intent intobj = new Intent(Intent.ACTION_VIEW);
intobj.putExtra("address", mobileno);
intobj.putExtra("sms_body", message);
intobj.setType("vnd.android-dir/mms-sms");
startActivity(intobj);
```

Figure 8.11: Using Built-in SMS application for sending SMS

SMS can be sent to more than one mobile number by separating the mobile number by a semi-colon(;).

8.4.2 Email

Sending email in android is done usually by using Intent Object by setting proper action and data. The intent is used to send information from one activity/component to another within or outside an application.

To send an email we need to follow the following steps :

1. Set the action to send data with the Intent object

   ```
   Intent emailobj = new Intent(Intent.ACTION_SEND);
   ```

   To launch an email client on the android device we use the action ACTION_SEND

2. Next we need to set the URI to "mailto:" by using the setData() method and the data type to "text/plain" by using the setType() method

   ```
   emailobj.setData(Uri.parse("mailto:"));
   emailobj.setType("text/plain");
   ```

3. Destination email-Id, copy to, subject of the email, and text message fields are attached to the Intent object by using the putExtra() method

   ```
   String[] to = {"example1@gmail.com"};
   String[] cc = {"example2@gmail.com"};
   emailobj.putExtra(Intent.EXTRA_EMAIL,to);
   emailobj.putExtra(Intent.EXTRA_CC,cc);
   emailobj.putExtra(Intent.EXTRA_SUBJECT,"Hello");
   emailobj.putExtra(Intent.EXTRA_TEXT,"Hello how are you ?");
   ```

4. Start the activity to send emails and finish

   ```
   startActivity(Intent.createChooser(emailobj,"Email"));
   finish();
   ```

In the Androidmanifest.xml file **<action ...>, <category ... >** and **<data ...>** tags have to be added as shown in figure 8.12 as follows

```
<intent-filter>
      . . .
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="message/rfc822"/>
 </intent-filter>
```

Figure 8.12: AndroidManifest.xml file with action, category, and data tags to send email

In the Activity file, the code for sending email is added at the click of a button

```
public class MainActivity extends AppCompatActivity {
    . .
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    // Add a reference to the button
     btnemail.setOnClickListener(new View.OnClickListener() {
       @Override
       public void onClick(View v) {
        Intent emailobj = new Intent(Intent.ACTION_SEND);
        emailobj.setData(Uri.parse("mailto:"));
        emailobj.setType("text/plain");

        String[] to = {"example1@gmail.com"};
        String[] cc = {"example2@gmail.com"};

        emailobj.putExtra(Intent.EXTRA_EMAIL,to);
        emailobj.putExtra(Intent.EXTRA_CC,cc);
        emailobj.putExtra(Intent.EXTRA_SUBJECT,"Hello");
        emailobj.putExtra(Intent.EXTRA_TEXT,"Hello to all");
        startActivity(Intent.createChooser(emailobj,"Email"));
        finish();
       }
    });
}}
```

Figure 8.13: Snippet for sending email in an Android application

In the layout file i.e., activity_main.xml a button is added, so that on click of the button the email is sent.

```
<Button
        android:id="@+id/btn_email"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="Send Mail"
        android:textStyle="bold"
        android:textSize="20sp"
        android:textColor="@color/white" />
```

Figure 8.14: Snippet for creating a Button in an Android application

On executing the application the following screen gets displayed

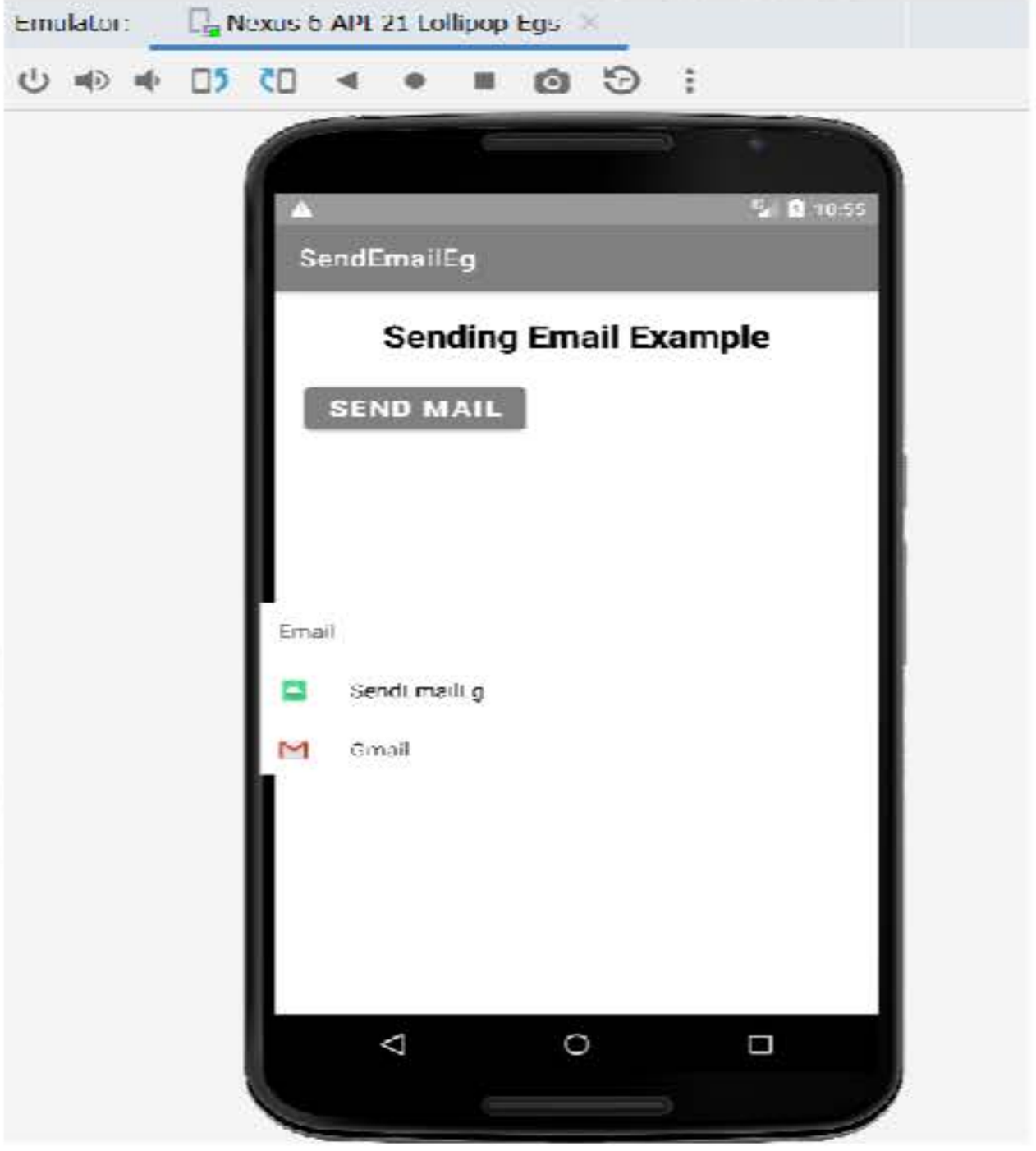

Figure 8.15: Sending email on Click of a button

8.5 Self-Assessment Questions

Q1. Discuss SQLite database in brief. [8 marks, L2]

Q2. Describe the use of SQLiteOpenHelper and Database Helper while using SQLite database.[6 marks, L2]

Q3. What is a RecyclerView? Explain the components of RecyclerView in brief. [8 marks, L2]

Q4. Explain Layout Manager used in RecyclerView. [5 marks, L2]

Q5. Explain the process of retrieving data from the database and displaying it in RecyclerView. [8 marks, L2]

Q6. Explain RecyclerView.Adapter and View Holder. [6 marks, L2]

Q7. Explain the purpose of using Model Class in RecyclerView with a suitable example and code.[8 marks, L2]

Q8. Describe in brief how SMS is sent in Android using SmsManager. [8 marks, L2] [Write the appropriate code used to send SMS]

Q9. Describe in brief how SMS is sent in Android using Intent. [8 marks, L2] [Write the appropriate code used to send SMS]

Q10. Explain with suitable code how emails are sent in Android.[8 marks, l2]

8.6 Self-Assessment Activities

Note: Use an appropriate emulator for the execution of an android application.

A1. Create an Android application, add appropriate controls, and create a database by the name **"GEUSHOPPING.db"** and add the table named **"ProductsInfo"** with fields **(PRODIMGID INTEGER PRIMARY KEY AUTOINCREMENT, PRODNAME Text, PRODIMG BLOB, PRODPRICE Real)** so that on the execution of the application, the database with the table is created.

A2. Add data to the table created in A1, add a recycler view, view holder, recycler-view adapter and a model class and display the information in the recycler view as shown in figure 8.7.

A3. Create an Android application, add appropriate controls, and write the appropriate code for sending SMS using SmsManager class.

A4. Create an Android application, add appropriate controls, and write the appropriate code for sending SMS using Intent.

A5. Create an Android application, add appropriate controls, and write the appropriate code for sending an email using Intent.

8.7 Multiple-Choice Questions

1. Which of the following database is inbuilt into Android Studio? [1 mark, L2]
   a) MySQL
   b) MySQL Server
   c) SQL Server
   d) SQLite

2. Which of the following is not supported by SQLite database? [1 mark, L2]
   a) Query language
   b) Stored procedures
   c) BLOB data type
   d) Real data type

3. SQLite database requires ____. [1 mark, L1]
   a) no installation and is serverless
   b) a server
   c) a server as well as installation
   d) none of the above

4. RecyclerView in Android is used to create a _____. [1 mark, L1]
   a) Static display of items
   b) Scrollable display of items
   c) Recycler to store deleted files
   d) Non-scrollable display of items

5. **`onCreate()`** method of the Database Helper class is called _____ in a lifetime of an application. [1 mark, L1]
   a) Once
   b) Twice
   c) Thrice
   d) Many times

6. View Holder used in RecyclerView can contain ____. [1 mark, L1]
   a) Only Images
   b) Only Text

c) Only integer data
d) Both Images and Text

7. SMS in android is sent by using _____. [1 mark, L1]
   a) SmsManager class
   b) SMSHelper class
   c) SmsDatabase class
   d) None of the above

8. When sending SMS the action to be used while creating Intent Object is ___. [1 mark, L1]
   a) new Intent(Intent.ACTION_SEND);
   b) new Intent(Intent.ACTION_SMS);
   c) new Intent(Intent.ACTION_VIEW);
   d) None of the above

9. When sending an Email the action used while creating Intent Object is____. [1 mark, L1]
   a) new Intent(Intent.ACTION_SEND);
   b) new Intent(Intent.ACTION_SMS);
   c) new Intent(Intent.ACTION_VIEW);
   d) None of the above

10. RecyclerView Layout Managers are ____. [1 mark, L1]
   a) LinearManager, GridManager, StaggeredManager
   b) LinearLayoutManager, GridLayoutManager, StaggeredGridLayoutManager
   c) LinearManagerLayout, GridManagerLayout, StaggereManagerLayout
   d) None of the above

8.8 Key Answers to multiple-choice questions

1. SQLite database is inbuilt into Android Studio.[d]
2. Stored Procedures are not supported by SQLite database [b]
3. SQLite database requires no installation and is serverless.[a]
4. RecyclerView in Android is used to create a Scrollable display of items.[b]
5. **`onCreate()`** method of the Database Helper class is called once in a lifetime of an application.[a]
6. View Holder used in RecyclerView can contain both Images and Text.[d]

7. SMS in android is sent by using SmsManager class.[a]
8. When sending SMS the action to be used while creating Intent Object is new Intent(Intent.ACTION_VIEW).[c]
9. When sending an Email the action used while creating Intent Object is new Intent(Intent.ACTION_SEND).[a]
10. RecyclerView Layout Managers are LinearLayoutManager, GridLayoutManager, and StaggeredGridLayoutManager.[b]

8.9 Summary

Android SQLite database is an open-source, lightweight database available on all android devices. SQLite supports the features of a relational database where data is stored in the form of tables, but on android devices, data is stored in the form of text files. Operations like add, delete, update and retrieve can be performed on the data. In-built user interface tools for creating SQLite databases are not available in Android. SQLite database does not support stored procedures. To create a database in android SQLiteOpenHelper class is used.

SQLiteOpenHelper Class - This class is available in android.database.sqlite.SQLite database package. All operations like creating databases and tables, handling database operations like add, delete, update, retrieve, etc. and database versions are managed by SQLiteOpenHelper class.

SQLiteOpenHelper class contains two methods onCreate() and onUpgrade() which have to be overridden by the helper class.

SQLiteDatabase class also contains methods to execute SQL commands.

Recycler-view is a control/widget which is used to efficiently display large datasets in a flexible and scrollable manner. Only a few items/views which fit on the screen are displayed and are visible while other items can be made visible by scrolling. It acts as a container and is considered an advanced version of ListView and GridView.

Components of Recycler-View

- Layout Managers
- View Holder
- RecyclerView.Adapter
- Model Class

View Holder represents an individual item in the ListView or GridView. It acts like a wrapper that contains other views and metadata.

RecyclerView Adapter is used to bind the data from the dataset to the ViewHolder that displays it within the RecyclerView.

It contains two methods onCreateViewHolder() and onBindViewHolder() which have to be overridden.

Model Class is a class that holds the data and business logic. Model class associate's data with view holder.

Messaging is one of the capabilities of android that helps to communicate with the outside world. Messaging can be done either by sending an SMS or email.

In Android, we can send SMS in 2 ways from an application - Using SMS Manager API and using Intent. Sending email in android is done usually by using Intent Object by setting proper action and data.

8.10 Keywords

- SQLite database
- SQLiteOpenHelper
- Database Helper
- Recycler-view
- View Holder
- Recycler-view Adapter
- Model class
- Messaging

8.11 Recommended Resources for Further Reading

Text Book(s)

1. Pradeep Kothari, "Android Application Development" Black-Book, Dreamtech Press,2015
2. Wei-Meng Lee, "Beginning Android Application Development", Wiley 2011.
3. Dawn Griffiths and David Griffiths, "Head First Android Development", O'Reilly, 2017.

References

- https://www.javatpoint.com/android-tutorial
- https://www.tutorialspoint.com/android/index.htm
- https://developer.android.com/guide/components/fundamentals

--*--

Contents

UNIT 4 - Android Tasks, Services, and Testing

CHAPTER 9 – Android Tasks and Services

Structure of Android Tasks and Services

9.1 Learning Outcomes:

After the successful completion of this chapter, the student will be able to:

- Describe AsynTask Class used in Android for background processing
- Demonstrate the use of AsyncTask in Android with ProgressBar Updates
- Describe the use of AsyncTask to send HTTP requests in the background
- Describe location-based services in Android

9.2 AsyncTask

9.2.1 AsyncTask Overview and methods

In Android, application code statements are executed in the main thread. Sometimes a task may require more time to process which may block the main thread until the operation is performed and completed. In such cases, the tasks that require more time and do not need user interaction can be performed in the background by using another thread. Asynchronous task (AsyncTask) in android allows, execution of some of the instructions in the background using a background thread and later again synchronizing with the main thread. The results of the background thread are later passed to the main thread, this makes the main user interface thread light and the application more responsive.

AsyncTask is a helper class in android used to perform background processing.

An AsyncTask class in android consists of 3 generic types

- TypeOfVarArgParams - It contains the information about the type of parameters sent to the task during execution
- ProgressValue - When the background processing is going on, the updates of it are sent to the user interface. It contains information about the progress
- ResultValue - The result information is stored in it.

The general syntax is as follows. In case only some generic types are used by the asynchronous Task class then it is marked as unused by using the type void.

```
private class MyBGTask extends AsyncTask<Void, Integer, String>
{
    …
 }
```

AsyncTask contains 4 methods which are executed as shown in figure 9.1

1. onPreExecute()
2. doInBackground()
3. onProgressUpdate()
4. onPostExecute()

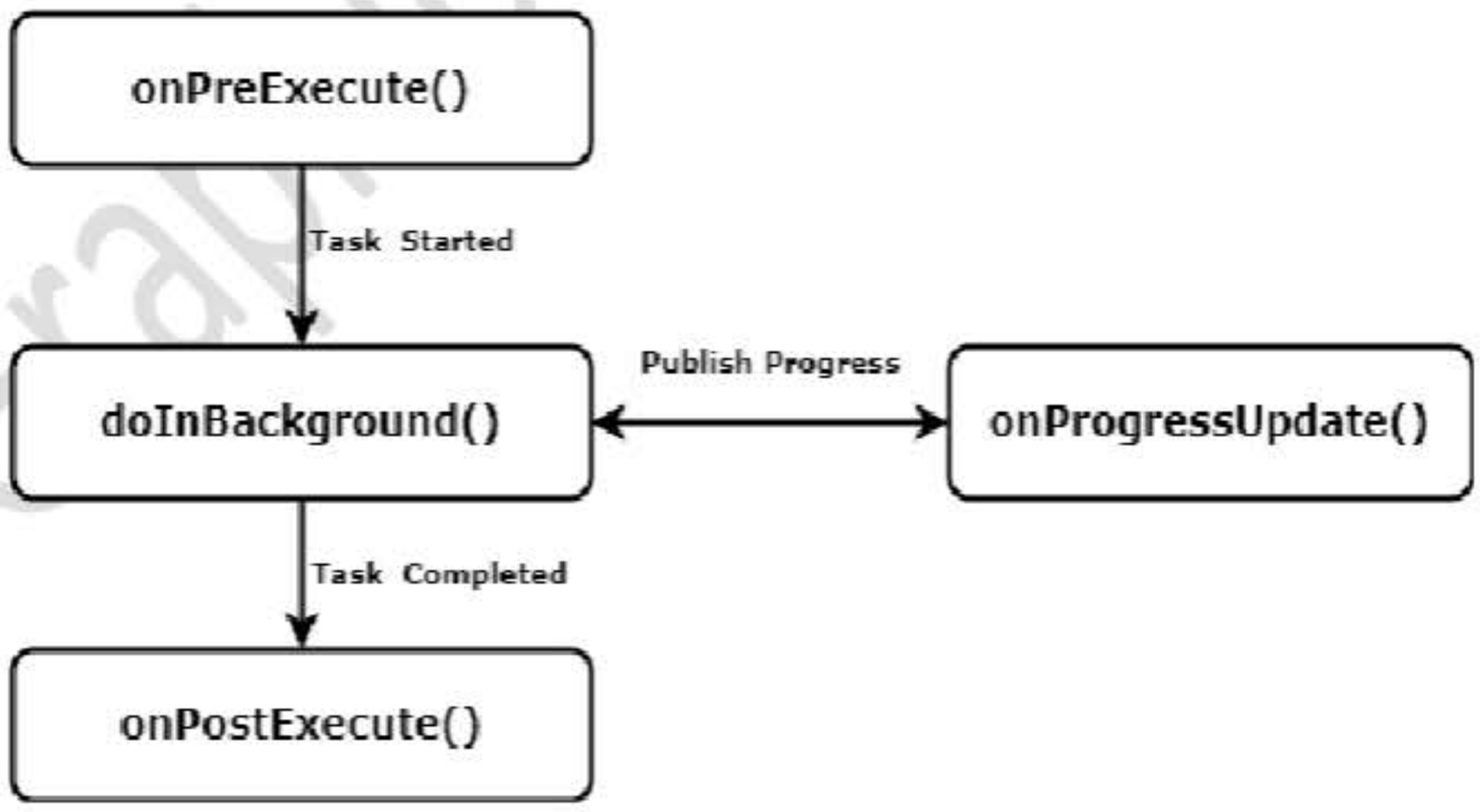

Figure 9.1: AsyncTask methods execution process

1. **onPreExecute()** – This method is invoked before starting the background operation. In this method usually, some instance of ProgressBar/ProgressDialog or animation is created and shown in the user interface to notify the user that some background operation is about to start.
2. **doInBackground()** – In this method background operations instructions are executed in the background thread. This method is executed immediately after onPreExecute()
3. **onProgressUpdate()** – This method is used to update the user interface i.e., the instance of ProgressBar/ProgressDialog or animation set up in the **onPreExecute()** method while the background operations are still going on.
4. **onPostExecute()** – This method is invoked after the background operations are completed. The user interface is updated to show the result.

Sample of Custom class extended from AsyncTask class with methods

```
private class MyBGTask extends AsyncTask<Void, Integer, String>
{
        @Override
        protected void onPreExecute() {
              // Task to be performed before starting background task
         }
        @Override
        protected String doInBackground(Void... voids) {
            // execute a background task pass value to onProgressUpdate()
            publishProgress(integer-value);
             return "xxx";
       }
        @Override
        protected void onProgressUpdate(Integer... values) {
              // show the progress of the background task
        }
        @Override
        protected void onPostExecute(String result) {
               // executed on completion of background task
        }
}
```

Figure 9.2: Sample of Custom class extended from AsyncTask with methods

9.2.2 Implementation of AsynTask with example

Example of AsyncTask with thread executing in the background and showing the progress of the process in ProgressBar

Create a project in Android and add a Button and ProgressBar to the activity_main.xml file as shown here.

```
<ProgressBar
     android:id="@+id/prgbar"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_gravity="center"
     android:minHeight="20dip"
     android:maxHeight="20dip"
     style="@android:style/Widget.ProgressBar.Horizontal" >
</ProgressBar>
```

Figure 9.3: Snippet of activity_main.xml file with ProgressBar

Make changes in the MainActivity.java file to execute the AsyncTask in the background and show the progress of the background process in ProgressBar

```
public class MainActivity extends AppCompatActivity {
   ..
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // get the references to the button and ProgressBar
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
              AsyncTaskExample asyncobj = new AsyncTaskExample();
              asyncobj.execute();
            }
        });
    }
    private final class AsyncTaskExample extends
                                AsyncTask<Void, Integer, String> {
        @Override
```

```
        protected void onPreExecute() {
            super.onPreExecute();
            prgbar.setVisibility(View.VISIBLE);
            prgbar.setMax(100);
            prgbar.setProgress(0);
         }
        @Override
        protected String doInBackground(Void... voids) {
            for (int i = 0; i < 10; i++) {
                try {
                    Thread.sleep(1000);
                    publishProgress(i*10);
                } catch (InterruptedException e) {
                    e.printStackTrace();   // Interrupted
                }
            }
            return "completed";
        }
        @Override
        protected void onProgressUpdate(Integer... values) {
             prgbar.setProgress(values[0]);
        }
        @Override
        protected void onPostExecute(String result) {
            if (result != null) {
                Toast.makeText(MainActivity.this,"Background Task  : "+
                                result, Toast.LENGTH_LONG).show();
            } else
                Toast.makeText(MainActivity.this,"Background Task
                                incomplete", Toast.LENGTH_LONG).show();
                prgbar.setVisibility(View.GONE);
        }
    }}
```

Figure 9.4: Snippet of MainActivity.java with AsynTask implementation

On Executing the status of the ProgressBar is shown in the display

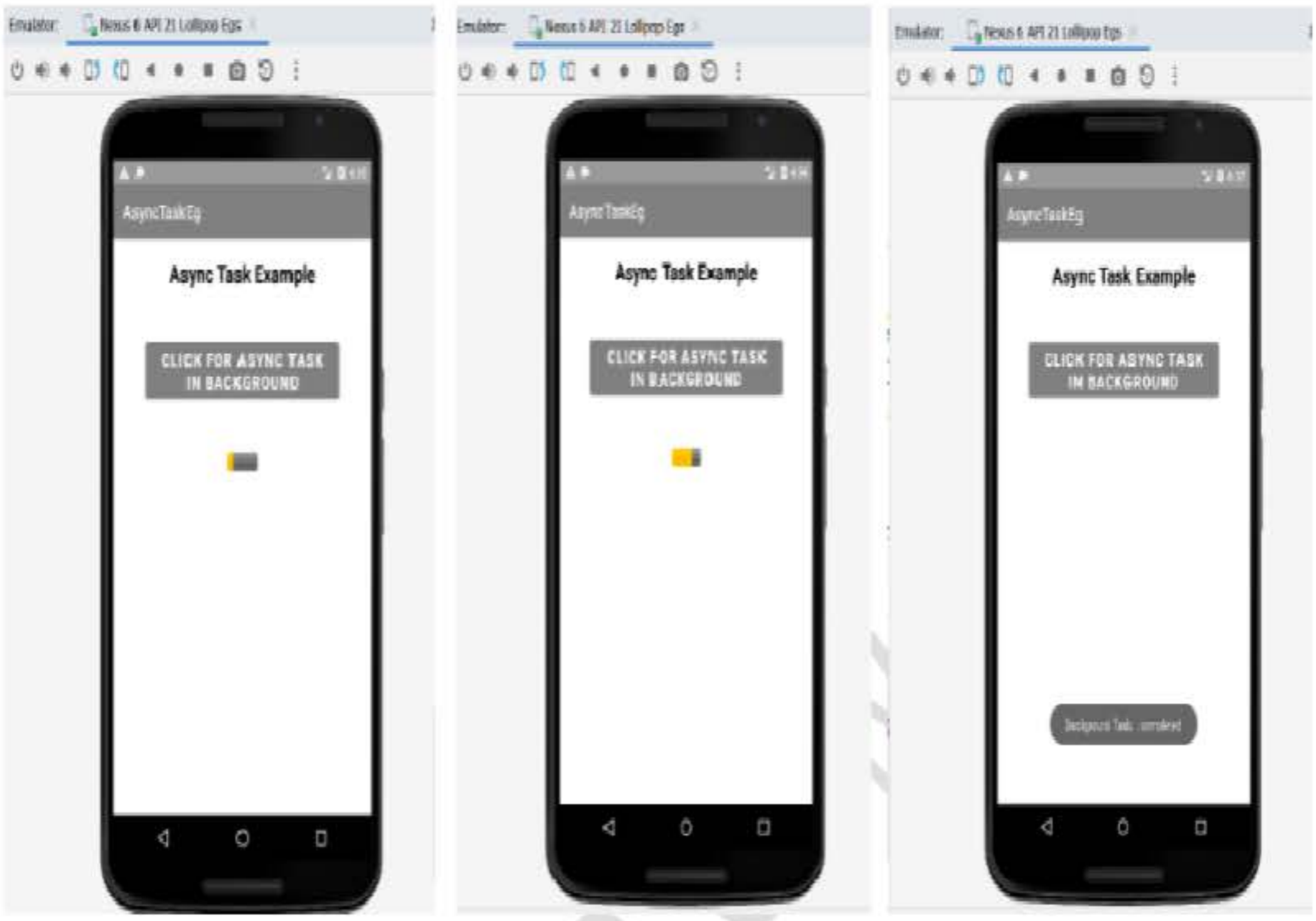

Figure 9.5: AsyncTask with ProgressBar showing progress and completion of Task

9.2.3 AsynTask using HTTP

To send an HTTP request in Android the following classes are used in Android

- HttpClient
- AndroidHttpClient
- HttpURLConnection

To send an HTTP Request we need to

- Declare a URL Connection and create a URL object
- Open the connection by creating an object of HttpUrlConnection class
- Once the connection is successful, download and decode the data based on its type
- Wrap in AsyncTask Class object and execute it in the background thread

Sample Code to download an image from a specific site and display it

1. Declare a URL Connection and create a URL object

```
URL urlobj = new URL("https://xxx.com/imagefile.jpg");
```

2. Open the HTTP connection by creating an object of HttpUrlConnection class

```
HttpURLConnection con = (HttpURLConnection)
                        urlobj.openConnection();
con.connect();

// Open InputStream to connection
InputStream ins = con.getInputStream();
```

3. Download the image and decode using BitmapFactory

```
Bitmap bmpobj = BitmapFactory.decodeStream(ins);
ins.close();
```

4. Insert the downloaded image in the ImageView control for display

```
ImageView imgobj = (ImageView) findViewById(R.id.imageview1);
Imgobj.setImageBitmap(bmpobj);
```

Figure 9.6: Snippet of AsynTask using HTTP Connection to download Image

9.3 Location-Based Services

Location-based services(LBS) in android are very popular nowadays. LBS applications help to track the device location and locate nearby amenities like hospitals, hotels, restaurants, stores, pharmacies, tourist places, etc. They help in finding routes and offer suggestions regarding route planning. Cab drivers nowadays use google maps to find routes. They also provide features of geofencing

By accessing Google Play Services and Location Framework in android, location-based applications can be developed easily.

9.3.1 Displaying Google Maps

For integrating google maps in an android application, Google Play Services are used. The Maps API key has to be obtained by the process of registration and then used in the android application for displaying the map.

Google Maps display the current location, search location, navigation direction, etc.

There are 4 types of google maps

1. Normal - This map displays rivers, roads, mountains, buildings, etc.
2. Hybrid - This map displays data received through satellite photographs along with roads and features that are labeled.
3. Satellite - This map displays only data received through satellite photographs and does not display road labels.
4. Terrain - This map displays photographic information which includes labels, lines, colors, perspective shading, etc.

Google Map API provides methods that are used for getting and customizing google maps

9.3.2 Getting Location data

Location data in android is fetched using GPS receivers, Cellular networks (Cell tower triangulation), and Wi-Fi.

GPS receivers use satellites to easily find the location. But in case of the unclear sky and in places like tunnels through mountains where satellites can't penetrate it becomes a problem to find the location.

Cellular networks work best when the cell towers are located very close to each other. Knowing the cell tower's identity it is easy to get the location and its information.

Wi-Fi can also be used to find the location

For getting the location data the following classes and interfaces are used in an android application

LocationManager class – provides Location Services for the system

LocationListener interface – notifications/updates from LocationManager class are received using this interface

LocationProvider class - provides device location

Location class – It provided information about the location like longitude and latitude etc. For e.g. getLongitude(), getLatitude(), and getAltitude() methods are available in this class.

Geocoders provide 2 types of services for getting location data

Forward Geocoding - translating address information to longitude and latitude

Reverse Geocoding – translating from longitude and latitude to address form

9.4 Self-Assessment Questions

Q1. Give an overview of AsyncTask Class and its methods.[8 marks, L2]

Q2. Illustrate with suitable code the use of AsyncTask to run threads in the background and show the progress of the process in ProgressBar.[8 marks, L2]

Q3. Discuss AsyncTask used with HTTP request in brief. [6 marks, L2]

Q4. Explain location-based services in android. [8 marks, L2]

Q5. Write a note on [5 marks each, L2]

- Google maps in android studio
- Getting location data

9.5 Self-Assessment Activities

Note: Use an appropriate emulator for the execution of an android application.

A1. Create an Android application, and add appropriate controls. Demonstrate the use of AsyncTask with thread executing in the background and showing the progress of the process in ProgressBar. The output should be as shown in figure 9.5.

9.6 Multiple-Choice Questions

1. The class used in Android to perform background processing is _____. [1 mark, L1]
   a) SyncTask
   b) AsyncTask
   c) BackgroundTask
   d) None of the above

2. The method which gets executed before the background process starts is __. [1 mark, L1]
   a) onPreExecute()
   b) PreExecute()
   c) beforestart()
   d) None of the above

3. The method which does not belong to AsyncTask Class is ____. [1 mark, L1]
   a) onPreExecute()
   b) doInBackground()
   c) onEndExecute()
   d) onPostExecute()

4. The AsyncTask method in which background operations instructions are executed in the background thread is ____. [1 mark, L1]
   a) onPreExecute()
   b) doInBackground()
   c) onProgressUpdate()
   d) onPostExecute()

5. To send an HTTP request the class used in Android is ____. [1 mark, L1]
   a) HttpurlConnection()
   b) HTTPClient
   c) HTTPServer
   d) HttpURLConnection

6. To use google maps in the android studio we need to use _____. [1 mark, L1]
   a) Google Maps API
   b) Google APIs
   c) Google Play Services
   d) None of the above

7. The type of map that displays satellite photograph data with typical road maps and feature labels in android is _____. [1 mark, L1]
   a) Hybrid
   b) Normal
   c) Satellite
   d) Terrain

8. The type of map that displays a typical road map, natural features like rivers, and some features built by humans in the android is ____. [1 mark, L1]
   a) Hybrid
   b) Normal
   c) Satellite
   d) Terrain

9. The methods getLongitude() and getLatitude() belong to the class ____. [1 mark, L1]
   a) LocationManager

b) LocationProvider
c) Location
d) None of the above

10. Translating address information to longitude and latitude in android is known as ___. [1 mark, L1]
    a) Forward Geocoding
    b) Reverse Geocoding
    c) Translation
    d) GPS receiver

9.7 Key Answers to multiple-choice questions

1. The class used in Android to perform background processing is AsyncTask. [b]
2. The method which gets executed before the background process starts is onPreExecute(). [a]
3. The method which does not belong to AsyncTask Class is onEndExecute().[c]
4. The AsyncTask method in which background operations instructions are executed in the background thread is doInBackground(). [b]
5. To send an HTTP request the class used in Android is HttpURLConnection.[d]
6. To use google maps in the android studio we need to use Google Play Services.[c]
7. The type of map that displays satellite photograph data with typical road maps and feature labels in android is Hybrid.[a]
8. The type of map that displays a typical road map, natural features like rivers, and some features built by humans in the android is Normal.[b]
9. The methods getLongitude() and getLatitude() belong to the class Location.[c]
10. Translating address information to longitude and latitude in android is known as Forward Geocoding.[a]

9.8 Summary

Asynchronous task (AsyncTask) in android allows, execution of some of the instructions in the background using a background thread and later again synchronizing with the main thread. The results of the background thread are later passed to the main thread, this makes the main user interface thread light and the application more responsive.

AsyncTask is a helper class in android used to perform background processing.

An AsyncTask class in android consists of 3 generic types -TypeOfVarArgParams, ProgressValue, and ResultValue.

AsyncTask contains 4 methods - **onPreExecute(), doInBackground(), onProgressUpdate() and onPostExecute().**

To send an HTTP request in Android the following classes are used in Android – HttpClient, AndroidHttpClient, and HttpURLConnection.

Location-based services(LBS) in android are very popular nowadays. LBS applications help to track the device location and locate nearby amenities like hospitals, hotels, restaurants, stores, pharmacies, tourist places, etc. They help in finding routes and offer suggestions regarding route planning. They also provide features of geofencing.

By accessing Google Play Services and Location Framework in android, location-based applications can be developed easily.

For integrating google maps in an android application, Google Play Services are used.

The Maps API key has to be obtained by the process of registration and then used in the android application for displaying the map.

Google Maps display the current location, search location, navigation direction, etc.

There are 4 types of google maps - Normal, Hybrid, Satellite, and Terrain.

Google Map API provides methods that are used for getting and customizing google maps

Location data in android is fetched using GPS receivers, Cellular networks (Cell tower triangulation), and Wi-Fi.

9.9 Keywords

- AsynTask
- ProgressBar
- HttpURLConnection
- Google maps
- Terrain
- Location-Based Services
- GPS receivers
- Cellular networks (Cell tower triangulation)
- Wi-Fi.
- Forward Geocoding
- Reverse Geocoding

9.10 Recommended Resources for Further Reading

Text Book(s)

1. Pradeep Kothari, "Android Application Development" Black-Book, Dreamtech Press,2015
2. Wei-Meng Lee, "Beginning Android Application Development", Wiley 2011.
3. Dawn Griffiths and David Griffiths, "Head First Android Development", O'Reilly, 2017.

References

- https://www.javatpoint.com/android-tutorial
- https://www.tutorialspoint.com/android/index.htm
- https://developer.android.com/guide/components/fundamentals

--*--