14EE706/14EC706/IT01

Hall Ticket Number:

IV/IV B.Tech (Regular/Supplementary) DEGREE EXAMINATION

November,2018 **Common to ECE & EEE**

Seventh Semester **Mobile Application Development**

Time: Three Hours **Maximum :** 60 Marks

Answer Question No.1 compulsorily. (1X12 = 12 Marks)

Answer ONE question from each unit. (4X12=48 Marks)

1. Answer all questions (1X12=12 Marks)
   a) How to overload a constructor
   b) Mention any Method to initialize your staticvariables.
   c) What is a file stream
   d) What is android
   e) List basic views in android user interface
   f) What is dialog fragment
   g) What is image switching?
   h) What is an intent object?
   i) What is SQLite
   j) What is the role of XML in android applications
   k) Write syntax of selecting a location provider
   l) What is reverse geocoding ?

UNIT I

2. a) Explain the concept of Nested and Inner Classes with examples. 8M
   b) Define a Package. How to Find Packages with CLASSPATH? 4M

(OR)

3. a) Explain Variables in Interfaces 6M
   b) Briefly explain delegation event model. 6M

UNIT II

4. a) Mention the salient features of Android. 6M
   b) Explain Android studio. 6M

(OR)

5. a) Draw and explain Fragment life cycle. 6M
   b) Explain picker views and list views. 6M

UNIT III

6. a) How to pass data using an intent object? 6M
   b) How to receive data using Broadcast Receivers? 6M

(OR)

7 a) Discuss about content values and cursors. 6M
   b) Briefly discuss the working with files. 6M

UNIT IV

8. a) Illustrate sending SMS using intents 6M
   b) Discuss Native android content providers 6M

(OR)

9. a) Illustrate the process of mapping current location 6M
   b) How to create map based activity 6M

14EE706/14EC706/IT01

IV/IV B.Tech (Regular) DEGREE EXAMINATION

NOVEMBER, 2018 **Common to ECE & EEE**
Seventh Semester **Mobile Application Development**

SCHEME OF EVALUATION

1. Answer all questions **(1 X 12=12 Marks)**

a) How to overload a constructor?

Ans) define more than one constructor with different parameter list, in such a way that each constructor performs a different task.

b) Mention any Method to initialize your static variables.

Ans) using static block, you can do static variables initialization in Java.

c) What is a file stream?

Ans) Is a Stream which allows reading/writing from/to a file.

d) What is android?

Ans) *Android is* an open source software stack that includes the operating system, middleware, and key applications along with a set of API libraries for writing mobile applications that can shape the look, feel, and function of mobile handsets.

e) List basic views in android user interface

Ans) TextView, EditText, Button, CheckBox, ToggleButton, ProgressBar, AutoCompleteTextView etc...

f) What is dialog fragment?

Ans) A DialogFragment is a fragment that displays a dialog window, floating on top of its activity's window. This fragment contains a Dialog object, which it displays as appropriate based on the fragment's state. Alternatively, you can create an entirely custom dialog, such as an AlertDialog, with its own content.

g) What is image switching?

Ans) Adds transitions on the images by using a ImageSwitcher view in Android.

h) What is an intent object?

Ans) An Intent object is a bundle of information which is used by the component that receives the intent as well as information used by the Android system.

i) What is SQLite?

Ans) SQLite is a open source SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation. SQLite supports all the relational database features.

j) What is the role of XML in android applications?

Ans) In Android XML is used to define layouts, colors, fonts, styles, strings, arrays, shapes and app configuration manifest file. All the UI and layout of your app is designed using xml.

k) Write syntax of selecting a location provider.

Ans)

```
String providerName = LocationManager.GPS_PROVIDER;
LocationProvider gpsProvider= locationManager.getProvider(providerName);
```

l) What is reverse geocoding ?

Ans) Reverse geocoding returns street addresses for physical locations specified by latitude/longitude pairs.

UNIT I

2 a) Explain the concept of Nested and Inner Classes with examples. **8M**

Explanation = 5M

Ans) It is possible to define a class within another class such classes are known as nested classes. A nested class has access to all of the variables and methods of its outer class. But the outer class doesn't access the variables and methods of nested class. Since the scope of a nested class is bounded by the scope of its enclosing class.

Why Use Nested Classes?

- It is a way of logically grouping classes that are only used in one place.
- It increases encapsulation.
- Nested classes can lead to more readable and maintainable code.

There are two types of nested classes: static and non-static

Static Nested class : A nested class defined with keyword static is known as static Nested class.

- Static nested classes are accessed using the enclosing class name:
  OuterClass.StaticNestedClass
- Static nested classes **do not have access to other members** of the enclosing class.
- to create an object for the static nested class, use this syntax:

OuterClass.StaticNestedClass nestedObject = new OuterClass.StaticNestedClass();

Non Static Nested Class : Non static nested class is also known as "**Inner Class**".

- A nested class is a member of its enclosing class. Non-static nested classes (inner classes) have access to other members of the enclosing class, even if they are declared private.
- To instantiate an inner class, you must first instantiate the outer class. Then, create the inner object within the outer object with this syntax:
  OuterClass.InnerClass innerObject = outerObject.new InnerClass();

Example = 3M

Example:

```
class outer
{
        int out_x=34;
        void test()
        {
        inner in=new inner();
        in.display();
        }
        class inner
        {
        void display()
        {
        System.out.println("value="+out_x);
        }
        }
}
class test
{
        public static void main(String[] args)
        {
        outer y=new outer();
        y.test();
        }
}
```

2 b) Define a Package. How to Find Packages with CLASSPATH? 4M

Ans) Package is a collection of related classes. Each class defines number of methods.

A class path is an environmental variable, which tells the java virtual machine and other java tools (java, javac), where to find the class libraries, including user-defined class libraries.
packages are mirrored by directories. This raises an important question:
How does the Java run-time system know where to look for packages that you create? The answer has three parts. First, by default, the Java run-time system uses the current working directory as its starting point. Thus, if your package is in a subdirectory of the current directory, it will be found. Second, you can specify a directory path or paths by setting the CLASSPATH environmental variable. Third, you can use the -classpath option with java and javac to specify the path to your classes.

For example, consider the following package specification:
package MyPack In order for a program to find MyPack, one of three things must be true. Either the program can be executed from a directory immediately above MyPack, or the CLASSPATH must be set to include the path to MyPack, or the -classpath option must specify the path to MyPack when the program is run via java.
When the second two options are used, the class path must not include MyPack, itself. It must simply specify the path to MyPack. For example, in a Windows environment, if the path to MyPack is
C:\MyPrograms\Java\MyPack
then the class path to MyPack is

C:\MyPrograms\Java

The easiest way to try the examples shown in this book is to simply create the package directories below your current development directory, put the .class files into the appropriate directories, and then execute the programs from the development directory. This is the approach used in the following example.

(OR)

3 a) Explain Variables in Interfaces. **6M**

Explanation = 2M

Ans) You can use interfaces to import shared constants into multiple classes by simply declaring an interface that contains variables that are initialized to the desired values. When you include that interface in a class (that is, when you “implement” the interface), all of those variable names will be in scope as constants. (This is similar to using a header file in C/C++ to create a large number of #defined constants or const declarations.) If an interface contains no methods, then any class that includes such an interface doesn’t actually implement anything. It is as if that class were importing the constant fields into the class name space as final variables. The next example uses this technique to implement an automated “decision maker”:

Program = 4M

Example:

```
import java.util.Random;
interface SharedConstants {
        int NO = 0;
        int YES = 1;
        int MAYBE = 2;
        int LATER = 3;
        int SOON = 4;
        int NEVER = 5;
}

class Question implements SharedConstants {
        Random rand = new Random();
        int ask() {
                int prob = (int) (100 * rand.nextDouble());
                if (prob < 30)

                return NO; // 30%
                else if (prob < 60)
                return YES; // 30%
                else if (prob < 75)
                return LATER; // 15%
                else if (prob < 98)
                return SOON; // 13%
                else
                return NEVER; // 2%
        }
}

class AskMe implements SharedConstants {
        static void answer(int result) {
                switch(result) {
                        case NO:
```

```
                System.out.println("No");
                break;
                case YES:
                System.out.println("Yes");
                break;
                case MAYBE:
                System.out.println("Maybe");
                break;
                case LATER:
                System.out.println("Later");
                break;
                case SOON:
                System.out.println("Soon");
                break;
                case NEVER:
                System.out.println("Never");
                break;
            }
        }

public static void main(String args[]) {
        Question q = new Question();
        answer(q.ask());
        answer(q.ask());
        answer(q.ask());
        answer(q.ask());
        }
}
```

3 b) Briefly explain delegation event model. ***Explanation = 3M*** **6M**

Ans) The modern approach to handling events is based on the delegation event model, which defines standard and consistent mechanisms to generate and process events. Its concept is quite simple:
a source generates an event and sends it to one or more listeners. In this scheme, the listener simply waits until it receives an event. Once an event is received, the listener processes the event and then returns. The advantage of this design is that the application logic that processes events is cleanly separated from the user interface logic that generates those events. Auser interface element is able to "delegate" the processing of an event to a separate piece of code. In the delegation event model, listeners must register with a source in order to receive an event notification. This provides an important benefit: notifications are sent only to listeners that want to receive them. This is a more efficient way to handle events than the design used
by the old Java 1.0 approach. Previously, an event was propagated up the containment hierarchy until it was handled by a component. This required components to receive events that they did not process, and it wasted valuable time. The delegation event model eliminates this overhead.

3 objects = 3M

Events
In the delegation model, an event is an object that describes a state change in a source. It can be generated as a consequence of a person interacting with the elements in a graphical user interface. Some of the activities that cause events to be generated are pressing a button, entering a character via the keyboard, selecting an item in a list, and clicking the mouse. Many other user operations could also be cited as examples.
Events may also occur that are not directly caused by interactions with a user interface. For example, an event may be generated when a timer expires, a counter exceeds a value, a software or hardware failure occurs, or an operation is completed. You are free to define events that are appropriate for your application.
Event Sources

Asource is an object that generates an event. This occurs when the internal state of that object changes in some way. Sources may generate more than one type of event. A source must register listeners in order for the listeners to receive notifications about a specific type of event. Each type of event has its own registration method. Here is the general form:
public void addTypeListener(TypeListener el)

Event Listeners
A listener is an object that is notified when an event occurs. It has two major requirements. First, it must have been registered with one or more sources to receive notifications about specific types of events. Second, it must implement methods to receive and process these notifications.

UNIT II

4 a) Mention the salient features of Android. 6M

Ans) *The following are the most noteworthy Android SDK features:* ***Any 12 points = 6M***

- ✓ Comprehensive APIs for location-based services such as GPS and network-based location detection.
- ✓ Full support for applications that integrate map controls as part of their user interfaces
- ✓ Wi-Fi hardware access and peer-to-peer connections
- ✓ Full multimedia hardware control, including playback and recording with the camera and microphone.
- ✓ Uses SQLite, a light weight relational database, for data storage.
- ✓ Media libraries for playing and recording a variety of audio/video or still-image formats
- ✓ APIs for using sensor hardware, including accelerometers, gyroscopes, barometers, magnetometers, dedicated gaming controls, proximity and pressure sensors, thermometers.
- ✓ Libraries for using Bluetooth and NFC hardware for peer-to-peer data transfer
- ✓ IPC message passing
- ✓ Shared data stores and APIs for contacts, social networking, calendar, and multi-media
- ✓ Background Services, applications, and processes
- ✓ The ability to integrate application search results into the system searches
- ✓ An integrated open-source HTML5 WebKit-based browser
- ✓ Mobile-optimized, hardware-accelerated graphics, including a path-based 2D graphics library and support for 3D graphics using OpenGL ES 2.0
- ✓ Localization through a dynamic resource framework
- ✓ An application framework that encourages the reuse of application components and the replacement of native applications.
- ✓ Touchscreen support
- ✓ Tethering: Supports sharing of Internet connections as a wired/wireless hotspot
- ✓ Video Calling

4 b) Explain Android studio. 6M

Ans) Any 6 windows = 6M

Android Studio includes the following windows:

- **Project –** The project view provides an overview of the file structure that makes up the project allowing for quick navigation between files. Generally, double clicking on a file in the project view will cause that file to be loaded into the appropriate editing tool.

- **Structure –** The structure tool provides a high level view of the structure of the source file currently displayed in the editor. This information includes a list of items such as classes, methods and variables in the file. Selecting an item from the structure list will take you to that location in the source file in the editor window.
- **Favorites –** A variety of project items can be added to the favorites list. Right clicking on a file in the project view, for example, provides access to an Add to Favorites menu option. Similarly, a method in a source file can be added as a favorite by right clicking on it in the Structure tool window. Anything added to a Favorites list can be accessed through this Favorites tool window.
- **Build Variants** – The build variants tool window provides a quick way to configure different build targets for the current application project (for example different builds for debugging and release versions of the application, or multiple builds to target different device categories).
- **TODO –** As the name suggests, this tool provides a place to review items that have yet to be completed on the project. Android Studio compiles this list by scanning the source files that make up the project to look for comments that match specified TODO patterns. These patterns can be reviewed and changed by selecting the File -> Settings… menu option and navigating to the TODO page listed under IDE Settings.
- **Messages –** The messages tool window records output from the Gradle build system (Gradle is the underlying system used by Android Studio for building the various parts of projects into a runnable applications) and can be useful for identifying the causes of build problems when compiling application projects.

- **Terminal** – Provides access to a terminal window on the system on which Android Studio is running. On Windows systems this is the Command Prompt interface, whilst on Linux and Mac OS X systems this takes the form of a Terminal prompt.

  Gradle – The Gradle tool window provides a view onto the Gradle tasks that make up the project build configuration. The window lists the tasks that are involved in compiling the various elements of the project into an executable application. Right-click on a top level Gradle task and select the Open Gradle Config menu option to load the Gradle build file for the current project into the editor. Gradle will be covered in greater detail later in this book.
- **Memory Monitor –** Connects to running Android applications and monitors memory usage statistics in the form of a real-time graph.
- **Designer** – Available when the UI Designer is active, this tool window provides access to the designer's Component Tree and Properties panels.
- **AVD manager** The Android Virtual Device Manager is used to create and manage the virtual devices that will host instances of the Emulator.

▪ **Emulator** with the ability of dialer, sms, networking, gps, maps etc..

▪ **DDMS** (Dalvik Debug Monitoring Service) : Is a debugging tool which includes stack, heap, thread veiwing, process details and screen capture facilities.

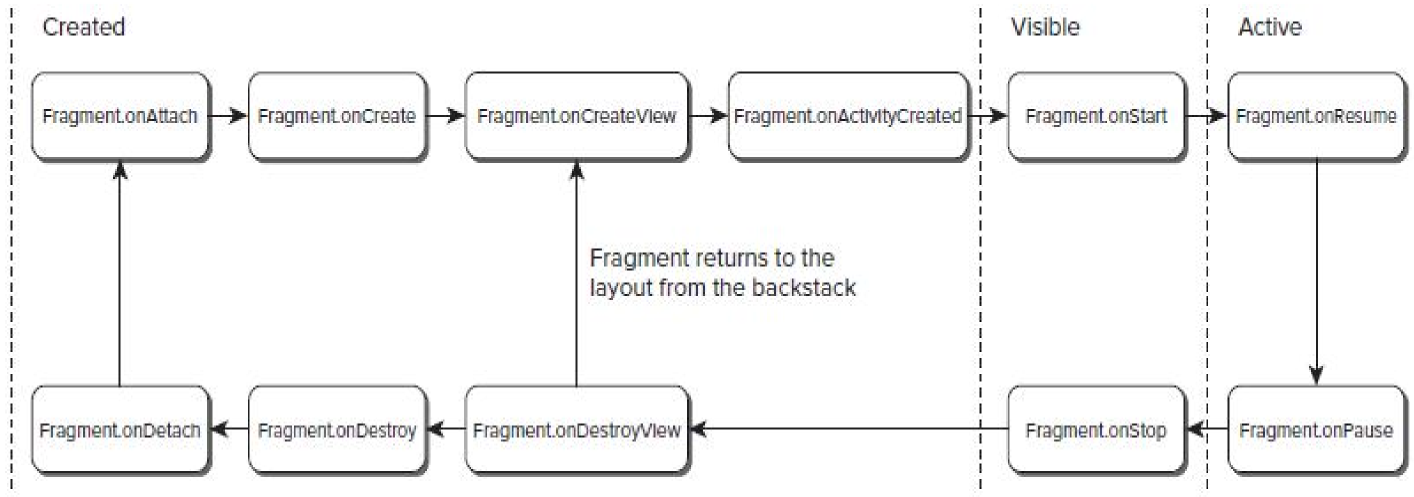
(OR)

5 a) Draw and explain Fragment life cycle. **6M**

Ans)

Diagram = 3M

The following diagram shows the life cycle of a fragment:



Explanation = 3M

The lifecycle events of a Fragment mirror those of its parent Activity; however, after the containing Activity is in its active — resumed — state adding or removing a Fragment will affect its lifecycle independently. Fragments include a series of event handlers that mirror those in the Activity class. They are triggered as the Fragment is created, started, resumed, paused, stopped, and destroyed. Fragments also include a number of additional callbacks that signal binding and unbinding the Fragment from its parent Activity, creation (and destruction) of the Fragment's View hierarchy, and the completion of the creation of the parent Activity.

Fragment life cycle handlers:

- **onAttach( ):** This method is called first even before onCreate() callback and after the fragment has been attached to the activity.
- **onCreate( ):** This method is called when creating the fragment and it takes Bundle as a reference which is used for initialize the essential components and store its value that you want to retain when the fragment ispaused or stopped,then resumed.
- **onCreateView( ):** This method is called when the fragment is added(link the appearance) to an activity for the first time.you must return a view from this method that is the root of your fragment's layout.you can return null if the fragment does not provide a UI.
- **onActivityCreated( ):** This method is called after Activity onCreate() callback has completed it's execution.this method is an indication for the activity has complete its execution before we try to freely access and modify UI elements of the activity.
- **onStart( ):** This method is called once the fragment gets visible in the activity after Activity onStart() callback.
- **onResume( ):** This method is called when the user interacting with the fragments in the activity after Activity onResume() callback.
- **onPause( ):** This method is called when the fragment is no longer interacting with the user either because it's activity is being paused or a fragment operation is modifying it in the activity.

- **onStop( ):** This method is called when the fragment is no longer interacting with the user either because it's activity is being stopped or a fragment operation is modifying it in the activity.
- **onDestroyView( ):** This method is called to allow the fragment to cleanup resources associated with it's view in the activity.
- **onDestroy( ):** This method is called to do final cleanup of the fragment's state.
- **onDetach( ):** This method is called to the fragment no longer being associated with its activity.

5 b) Explain picker views and list views. **6M**

Ans)

Picker views = 3M

Picker views - Views that enable users to select from a list, such as the TimePicker and DatePicker views.
List views - Views that display a long list of items, such as the ListView and the SpinnerView views.
TimePicker: The TimePicker view enables users to select a time of the day, in either 24-hour mode or AM/PM mode.

Use DatePicker setIs24HourView(true) for setting 24 hr format in the TimePicker view. Use setMinute( ), setHour( ), getMinute( ) & getHour( ) methods to set/get the time.

DatePicker: Using the DatePicker, you can enable users to select a particular date on the activity.
Use getMonth( ), getYear( ) and getDayOfMonth( ) methods to get the month, year & day respectively.
Use setFirstDayOfWeek( ) method to set the first day of week.

List views = 3M

List views are views that enable you to display a long list of items. In Android, there are two types of list views: ListView and SpinnerView. Both are useful for displaying long lists of items.

ListView: The ListView displays a list of items in a vertically scrolling list. Use ListAdapter to handle the ListView. onListItemClick( ) method is called whenever you selected an item from the ListView. You can identify the list item selected (position of the item selected is passed as parameter to the handler). Appearance of list items can be customized so that they display checkboxes, radio buttons along with them.

SpinnerView: The ListView displays a long list of items in an activity, but you might want the user interface to display other views, meaning you do not have the additional space for a full-screen view, such as the ListView. In such cases, you should use the SpinnerView. The SpinnerView displays one item at a time from a list and enables users to choose from them.

In the case of SpinnerView, use an ArrayAdapter to handle the SpinnerView items display. Use OnItemSelectedListener to listen for the events in the spinner. Whenever an item is selected from the SpinnerView, onItemSelected( ) method is called.

UNIT III

6 a) How to pass data using an intent object? **6M**

Ans)

Two techniques X 3M = 6M

It is possible to pass data to an activity using Intents.

Data can be placed in intent in two ways:

- **By using Intent's putExtra( ) method.**

Ex:

```
//---use putExtra() to add new name/value pairs---
i.putExtra("str1", "This is a string");
i.putExtra("age1", 25);
```

- **By using putExtras( ) method. (a Bundle object containing data is attached to the Intent)**

Ex:

```
    //---use a Bundle object to add new name/values pairs---
Bundle extras = new Bundle();
extras.putString("str2", "This is another string");
extras.putInt("age2", 35);
//---attach the Bundle object to the Intent object---
i.putExtras(extras);
```

6 b) How to receive data using Broadcast Receivers? **6M**

Ans) To setup a Broadcast receiver for receiving system wide broadcast events or intents, follow the below procedure:

Creation of Receiver = 3M

1. Creating a BroadcastReceiver
2. Registering a BroadcastReceiver

Creating a BroadcastReceiver:

```
public class MyReceiver extends BroadcastReceiver {
   public MyReceiver() {
   }

   @Override
   public void onReceive(Context context, Intent intent) {

      Toast.makeText(context, "Action: " + intent.getAction(), Toast.LENGTH_SHORT).show();
   }
}
```

The onReceive() method is first called on the registered Broadcast Receivers when any event occurs. The intent object is passed with all the additional data. A Context object is also available and is used to start an activity or service using context.startActivity(myIntent); or context.startService(myService); respectively.

Registering a BroadcastReceiver:

Registering Receiver = 3M

A BroadcastReceiver can be registered in two ways.

i) By defining it in the AndroidManifest.xml file as shown below.

```
<receiver android:name=".ConnectionReceiver" >
          <intent-filter>
             <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
          </intent-filter>
</receiver>
```

ii) By defining it programmatically

Following snippet shows a sample example to register broadcast receiver programmatically.

```
IntentFilter filter = new IntentFilter();
intentFilter.addAction(getPackageName() + "android.net.conn.CONNECTIVITY_CHANGE");
MyReceiver myReceiver = new MyReceiver();
registerReceiver(myReceiver, filter);
```

(OR)

7 a) Discuss about content values and cursors. **6M**

ContentValues = 3M

Ans)

ContentValues: Content Values are used to insert new rows into tables. Each ContentValues object represents a single table row as a map of column names to values.

Usage:

```
 public boolean insertData(String name,String surname,String marks) {
SQLiteDatabase db = this.getWritableDatabase();
ContentValues contentValues = new ContentValues();
contentValues.put(COL_2,name);
contentValues.put(COL_3,surname);
contentValues.put(COL_4,marks);
long result = db.insert(TABLE_NAME,null ,contentValues);
if(result == -1)
   return false;
else
   return true;
}
```

Cursors = 3M

Cursors: Database queries are returned as Cursor objects. Rather than extracting and returning a copy of the result values, Cursors are pointers to the result set within the underlying data. Cursors provide a managed way of controlling your position (row) in the result set of a database query.

The Cursor class includes a number of navigation functions, including, but not limited to, the following:

- moveToFirst – Moves the cursor to the fi rst row in the query result
- moveToNext – Moves the cursor to the next row
- moveToPrevious – Moves the cursor to the previous row
- getCount – Returns the number of rows in the result set
- getColumnIndexOrThrow – Returns the zero-based index for the column with the specifi ed
- name (throwing an exception if no column exists with that name)
- getColumnName – Returns the name of the specifi ed column index
- getColumnNames – Returns a string array of all the column names in the current Cursor
- moveToPosition – Moves the cursor to the specifi ed row
- getPosition – Returns the current cursor position

Usage:

➢
```
public void onClick(View v) {
    Cursor res = myDb.getAllData();
    if(res.getCount() == 0) {
        // show message
        showMessage("Error","Nothing found");
        return;
    }

    StringBuffer buffer = new StringBuffer();
    while (res.moveToNext()) {
        buffer.append("Id :"+ res.getString(0)+"\n");
        buffer.append("Name :"+ res.getString(1)+"\n");
        buffer.append("Surname :"+ res.getString(2)+"\n");
        buffer.append("Marks :"+ res.getString(3)+"\n\n");
    }
}
```

7 b) Briefly discuss the working with files. **6M**

Files Explanation = 6M

Ans)

➢ Android supplies some basic file-management tools to help you deal with the fi le system. Many of these utilities are located within the java.io.File package.

Using Application-Specific Folders to Store Files:

➢ There are two options for storing these application-specifi c fi les: internally or externally. Android offers two corresponding methods via the application Context, getDir and getExternalFilesDir, both of which return a File object that contains the path to the internal and external application fi le storage directory, respectively. All fi les stored in these directories or the subfolders will be erased when your application is uninstalled. Both of these methods accept a string parameter that can be used to specify the subdirectory into which you want to place your files. Files stored in the application folders should be specifi c to the parent application and are typically not detected by the media-scanner, and therefore won ’ t be added to the Media Library automatically. If your application downloads or creates fi les that should be added to the Media Library or otherwise made available to other applications

Creating Private Application Files:

Android offers the openFileInput and openFileOutput methods to simplify reading and writing streams from and to fi les stored in the application ’ s sandbox.

```
String FILE_NAME = “tempfile.tmp”;
// Create a new output file stream that’s private to this application.
FileOutputStream fos = openFileOutput(FILE_NAME, Context.MODE_PRIVATE);
// Create a new file input stream.
FileInputStream fis = openFileInput(FILE_NAME);
```

Storing Publicly Readable Files:

Android 2.2 (API level 8) also includes a convenience method, Environment.getExternal StoragePublicDirectory, that can be used to fi nd a path in which to store your application fi les.

The returned location is where users will typically place and manage their own fi les of each type. This is particularly useful for applications that provide functionality that replaces or augments system applications, such as the camera, that store fi les in standard locations.
The getExternalStoragePublicDirectory method accepts a String parameter that determines which subdirectory you want to access using a series of Environment static constants:

- DIRECTORY_ALARMS — Audio fi les that should be available as user-selectable alarm sounds
- DIRECTORY_DCIM — Pictures and videos taken by the device
- DIRECTORY_DOWNLOADS — Files downloaded by the user
- DIRECTORY_MOVIES — Movies
- DIRECTORY_MUSIC — Audio fi les that represent music
- DIRECTORY_NOTIFICATIONS — Audio fi les that should be available as user-selectable notifi -
- cation sounds
- DIRECTORY_PICTURES — Pictures
- DIRECTORY_PODCASTS — Audio fi les that represent podcasts
- DIRECTORY_RINGTONES — Audio fi les that should be available as user-selectable ringtones

Note that if the returned directory doesn't exit, you must create it before writing fi les to the directory, as shown in the following snippet:

```
String FILE_NAME = “MyMusic.mp3”;
File path = Environment.getExternalStoragePublicDirectory(
Environment.DIRECTORY_MUSIC);
File file = new File(path, FILE_NAME);
try {
path.mkdirs();
[... Write Files ...]
} catch (IOException e) {
Log.d(TAG, “Error writing “ + FILE_NAME, e);
```

UNIT IV

8 a) Illustrate sending SMS using intents. **6M**

Ans)

Explanation = 3M

Using the SmsManager class, you can send SMS messages from within your application without the need to involve the built-in Messaging application. However, sometimes it would be easier if you could simply invoke the built-in Messaging application and let it handle sending the message. To activate the built-in Messaging application from within your application, you can use an Intent object with the MIME type "vnd.android-dir/mms-sms", as shown in the following code snippet:

```
Intent i = new Intent(android.content.Intent.ACTION_VIEW);
i.putExtra("address", "5556; 5558; 5560");
i.putExtra("sms_body", "Hello my friends!");
i.setType("vnd.android-dir/mms-sms");
startActivity(i);
```

This code invokes the Messaging application directly. Note that you can send your SMS to multiple recipients by separating each phone number with a semicolon (in the putExtra() method). The numbers are separated using commas in the Messaging application.

Example code = 3M

Example code:

```
package com.example.bh2.sendsmsusingintentex;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Button btnSendSMS = (Button) findViewById(R.id.button1);
      btnSendSMS.setOnClickListener(new View.OnClickListener()
      {
         public void onClick(View v)
         {
            //sendSMS("5556", "Hello my friends!");
            Intent i = new Intent(Intent.ACTION_VIEW);
            i.setData(Uri.parse("smsto:"));
            i.putExtra("address", new String("5556"));
            i.putExtra("sms_body", "Have a nice day");
            i.setType("vnd.android-dir/mms-sms");
            startActivity(i);
            finish();
         }  });   }
}
```

8 b) Discuss Native android content providers. **6M**

Explanation = 3M

Ans)

Android exposes several native Content Providers, which you can access directly using the techniques described earlier in this chapter. Alternatively, the android.provider package includes APIs that can simplify access to many of the most useful Content Providers, including the following:

Media Store – Provides centralized, managed access to the multimedia on your device, including audio, video, and images. You can store your own multimedia within the Media Store and make it globally available, as shown in Chapter 15, " Audio, Video, and Using the Camera. "

Browser – Reads or modifi es browser and browser search history.

Contacts Contract – Retrieves, modifi es, or stores contact details and associated social stream updates.

Calendar – Creates new events, and deletes or updates existing calendar entries. That includes modifying the attendee lists and setting reminders.

Call Log – Views or updates the call history, including incoming and outgoing calls, missed calls, and call details, including caller IDs and call durations.

These Content Providers, with the exception of the Browser and Call Log, are covered in more detail in the following sections. You should use these native Content Providers wherever possible to ensure your application integrates seamlessly with other native and third-party applications.

Using Contacts content provider example code:

Example code = 3M

```
// Create a projection that limits the result Cursor
// to the required columns.
String[] projection = {
ContactsContract.Contacts._ID,
ContactsContract.Contacts.DISPLAY_NAME
};
// Get a Cursor over the Contacts Provider.
Cursor cursor =
getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,
projection, null, null, null);
// Get the index of the columns.
int nameIdx =
cursor.getColumnIndexOrThrow(ContactsContract.Contacts.DISPLAY_NAME);
int idIdx =
cursor.getColumnIndexOrThrow(ContactsContract.Contacts._ID);
// Initialize the result set.
String[] result = new String[cursor.getCount()];
// Iterate over the result Cursor.
while(cursor.moveToNext()) {
// Extract the name.
String name = cursor.getString(nameIdx);
// Extract the unique ID.
String id = cursor.getString(idIdx);
result[cursor.getPosition()] = name + " (" + id + ")";
}
// Close the Cursor.
cursor.close();
```

(OR)

9 a) Illustrate the process of mapping current location. 6M

5 steps = 6M

Ans) In your android app, you can show user's current location on Google map using Google maps android API. The API uses wifi or/and mobile cell data to determine device location. Location enabled map shows blue circle which indicates current location of the device and location button to help user move to the current location on the map.

Google maps android API is part of Google play service API. To make the API available in your android project, you need to follow few steps to setup your project and make it ready for using Google maps android API.

i) First get Google Maps API key to use Google MAPS in your code.
ii) Add location permissions in the manifest file
iii) To use device location, user permissions are need. So, first add android.permission.ACCESS_COARSE_LOCATION and

android.permission.ACCESS_FINE_LOCATION permissions to android manifest file in your project as shown below. And also, in the code, you need to check for the permissions and request them if not granted, see below Activity code to know how to request location permissions in the code.

iv) Add support Map fragment.

v) Enable location in Activity.

Create an activity implementing OnMapReadyCallback interface and adding callback handler to SupportMapFragment object in onCreate method. In onMapReady callback method, check for location permissions, if not granted, request them, and enable location by calling setMyLocationEnabled on GoogleMap object.
You can add OnMyLocationButtonClickListener and OnMyLocationClickListener location listeners to GoogleMap object by calling setOnMyLocationButtonClickListener and setOnMyLocationClickListener method on it.
Callback method onMyLocationButtonClick of OnMyLocationButtonClickListener is called when location button is clicked. Callback method of OnMyLocationClickListener listener is called when location is clicked. In these callback methods, you can do some customization such as adding shapes to map or changing zoom, etc. to suit your application requirements.

9 b) How to create map based activity? **6M**

Explanation = 4M

Ans) To use maps in your applications you need to extend MapActivity. The layout for the new class must then include a MapView to display a Google Maps interface element. The Android maps library is not a standard Android package; as an optional API, it must be explicitly included in the application manifest before it can be used. Add the library to your manifest using a uses-library tag within the application node, as shown in the following XML snippet:

```
<uses-library android:name="com.google.android.maps"/>
```

The maps package as described here is not part of the standard Android open-source project. It is provided within the Android SDK by Google and is available on most Android devices. However, be aware that because it is a nonstandard package, an Android device may not feature this particular library.
Google Maps downloads the map tiles on demand; as a result, it implicitly requires permission to use the Internet. To see map tiles in your Map View you need to add a <uses-permission> tag to your application manifest for internet, as shown here:

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

Once you've added the library and configured your permission, you're ready to create your new map-based Activity.
MapView controls can be used only within an Activity that extends MapActivity. Override the onCreate method to lay out the screen that includes a MapView, and override isRouteDisplayed to return true if the Activity will be displaying routing information (such as traffic directions).

Skeleton code for creating a map based activity:

Sample code = 2M

```
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import android.os.Bundle;
public class MyMapActivity extends MapActivity {
private MapView mapView;
private MapController mapController;
@Override public void onCreate(Bundle icicle)
 {
super.onCreate(icicle);
setContentView(R.layout.map_layout);
mapView = (MapView)findViewById(R.id.map_view);
}
 @Override protected boolean isRouteDisplayed()
 {}
}
```

Signature of the Internal Examiner	**Signature of the HOD**	**Signature of the External Examiner**