M.C.A. Semester – V
Subject: - Mobile Computing (650003)
Week : 2

1) What is Intent? How it is useful for transitioning between various activities? How intents can be received & broadcasted. (Unit :-2, Chapter :-4)

What is Intent?

- Definition: Intent (android.content.Intent) is an asynchronous message mechanism used by the Android operating system to match task requests with the appropriate Activity or Service (launching it, if necessary) and to dispatch broadcast Intents events to the system at large.
- Package: android.content.Intent.

How it is useful for transitioning between various activities?

- Android app have multiple entry points, No main function.
- One activity is designated as main / launcher activity in manifest file
- To transition from one activity to other after creating instance call **startActivity()** method which has "Intent" instance as argument.
- Activity can be launched in many ways using Intent
  - Launching new activity by class name (Creating simple intent)
    - Create instance of Intent with argument as target activity as

```
startActivity(new Intent(getApplicationContext()MyDrawActivity.class));
```

  - Creating Intent with Action & Data
    - Intent object are composed of two main parts:
      the action to be performed, and **the data** to be acted upon.
    - Most common action types : ACTION_MAIN, ACTION_EDIT
    - Here, intent is saying "Do this" (action) "to that" (data).
  - Launching activity belonging to other application using Intent.
    - Appropriate permission required to access activity of other package.
    - Create instance of Intent with two information : action & data
    - E.g. `Intent dial = new Intent(Intent.ACTION_DIAL, number);`
  - Passing additional data using Inent
    - Use Extras property

        - It stores info. In Bundle object
        - Create instance of Intent, use **putExtra()** method to put some data in form “name/value” pair as two arguments to it.

Receiving & Broadcasting Intent

- Broadcast Intent
    - boradcastIntent() method is used.
    - Purpose: To inform other applications about something interesting, use special intent action types.
    - Special intent action e.g. ACTION_BATTERY_LOW, ACTION_BATTERY_CHANGED. (sdcard state changed, application install/uninstall)
- Receiving Broadcasted Intent
    - Receive the broadcast using BroadcastReceiver

2) Explain various application settings using android manifest file. (Unit :-2, Chapter :-5)

Settings to be done in AndroidManifest.xml

- Application’s identity
    - **Versioning** (<manifest> tag, android:versionCode & android:versionName attributes).
    - **Application name & icon** (<application tag, android:icon & android:label attributes)
- System requirements
    - **Target SDK Version** (<manifest> parent tag, <uses-sdk> child tag, android:minSdkVersion, android:maxSdkVersion, android:targetSdkVersion attributes, values → integer no. specifying api level)
- Application platform requirement
    - **Supported Input Methods** (<manifest> parent tag, <uses-configuration> child tag, android:reqTouchScreen, android:reqKeyboardType, android:reqHardKeyboard, android:reqNavigation, android:reqFiveWayNav attributes, values → true/false)

    - **Required device features** (<manifest> parent tag, <uses-feature> child tag, android:name attributes, values→ appropriate values as per application requirement)
    - **Supported screen size** (<manifest> parent tag, <supports-screen> child tag, android:smallScreens, android:normalScreens, android:largeScreens, android:resizable, android:anyDensity, values→ true/false)
    - **Specifying external library reference** (<manifest> parent tag, <uses-library child tag, android:name attribute, values→ library name that application uses)
- **Registering application's activity, service, broadcast receiver & Intnet-filters**
    - **Activity** (<manifest> parent tag, <activity> child tag, android:name attributes, values → name of the activity class).
    - **Service** (<manifest> parent tag, <service> child tag)
    - **Broadcast Receiver** (<manifest> parent tag, <receiver> child tag)
    - **Intent-filters** (<manifest> parent tag, <intent-filter> child tag, specify <action>, <category> & <data> as sub tags of <intent-filter>)
- **Granting & taking permissions**
    - **Grant permission** (<permission> tag)
    - **Taking/ receiving permission** (<uses-permission tag, android:name attribute)

3) What are dialogs in android? Explain different types of dialogs supported by android. Write in brief lifecycle of a dialog. (Unit :- 3, Chapter :-7) (Ref. pg. no. 165)

Dialogs

- Used to organize information and react to user-driven events.

Types of Dialogs

- Dialog types → (Dialog, AlertDialog, CharacterPickerDialog, DatePickerDialog, ProgressDialog, TimePickerDialog purpose of each & example)
- Lifecycle of Dialog → (showDialog(), dismissDialog(), removeDialog() methods)
    - Provide unique id to dialog, implement onCreateDialog() method, Implement onPrepareDialog(), Launch dialog using showDialog() method)
    - Example of any of the dialog.

4) What are layouts? List built-in layouts supported by android. Explain any two with example. (Unit :- 3, Chapter :- 8)

Layouts

- used to organize screen elements.
- Layouts can be defined programmatically using java code & as resource using xml way.

Built-in Layouts

- **FrameLayout** (Purpose→ to display stack of child view items, list attributes, example, <FrameLayout> parent tag)
- **LinearLayout** (Purpose→ to organized child view items, in single row or single column depending on the orientation (horizontal / vertical, list attributes, example, <LinearLayout> parent tag)
- **RelativeLayout** (Purpose→ to organized child views in relation to each other, list attributes, example, <RelativeLayout> parent tag)
- **TableLayout** (Purpose→ to organize child view into rows, list attributes, example, <TableLayout> parent tag)
- **MultipleLayout** (Layouts can be nested)

5) Write an android application to create database namely "Student", create table "student_info". Write android code to demonstrate insert, update, delete & query data to and from this table. (Unit :- 4, Chapter :- 10)

Create Database (Ref. pg. no. 240)

- Create instance of SQLiteDatabase using openOrCreateDatabase() method with three arguments where 1st arg. Is name of the database ("student_info"), 2nd arg is mode, 3rd arg is factory

  ```
  SQLiteDatabase S = openOrCreateDatabase("Student", SQLiteDatabase.CREATE_IF_NECESSARY, null)
  ```

Creating table

- Write SQL statement to create table, specify table name, column name with data type & constraints: PRIMARY KEY, AUTOINCREMENT, NOT NULL etc. (Note: Foreign key constraint doesn't work with SQLite Database), store it in String variable.
- Use execSQL method to execute above query stored in string variable.

Insert Record

- Create instance of Content Values
- Use put() method with 2 arguments where 1st arg name of the column, 2nd arg. Value to that column)
- Use **insert()** method of SQLiteDatabase instance with **3 arguments**, where 1st arg. Name of the table, 2nd arg. To provide nullable column names or set to null, 3rd arg. Content values object) which return long rowid which is unique for every record.

Update Record

- Create instance of Content values
- Use put() method with 2 arguments where 1st arg name of the column, 2nd arg. Value to that column)
- Use **update()** method of SQLiteDatabase instance with **4 arguments,** where 1st arg. Is name of the table, 2nd arg. Is content value instance, 3rd arg. Where clause, 4th substitute value for where clause '?' parameter)

Deleting Record

- Use **delete()** method of SQLiteDatabase instance with **3 arguments**, where 1st arg. Is name of the table, 2nd arg. Where clause, 3rd substitute value for where clause '?' parameter)

Querying Data

- Create a reference variable of Cursor
- Use **query()** method of SQLiteDatabse instance
- **8 arguments**
  - 1st arg → Name of the table.
  - 2nd arg → List of column names
  - 3rd arg→ where clause
  - 4th arg →where clause substitution for ? parameter
  - 5th arg → GROUP BY clause
  - 6th arg → HAVING clause
  - 7th arg → ORDER BY clause
  - 8th arg → LIMIT clause

(2nd & 4th arg. Are of string array types, rest arguments are of string type)

6) **List default content provider supported by android. Explain CallLog & Broswer content provider with example. (Unit :- 4, Chapter :- 11)**

Built – in Content providers

- **MediaStore →** Audio – visual data on phone and external storage
- **CallLog →** To access & view sent & receive call.
- **Browser →** Browser history & bookmarks
- **ContactsContract →** Phone contact database of phone book
- **Settings →** System – wide device settings and preferences
- **UserDictionary →** A dictionary of user-defined words for use with predictive text input

Call Log Content Provider

- **Purpose →** To access & view sent & receive call.
- **URI →** CallLog.Calls.CONTENT_URI.
- **Columns →** (E.g. to view the name : CallLog.Calls.CACHED_NUMBER_LABEL, Duration : CallLog.Calls.DURATION etc.)
- Use **managedQuery()** method to retrieve info. for various columns with **5 arguments,** with 1st arg. URI, 2nd arg. Required columns, 3rd arg. Where clause to retrieve specific calls only, 4th arg. Values for where clause ? parameter., 5th arg. Defines sorting order.
- Use cursor methods to do various operations on that data.
- Requires permission: <uses-permission> parent tag, android:name attribute, value : android.permission.READ_CONTACTS.

Browser Content Provider

- **Purpose →** For user's browser history and bookmarked websites.
- **URI →** Browser.BOOKMARKS_URI
- **Columns →** (E.g. to view the name of the website url : Browser.Bookmarkcolumns.TITLE, no. of visits to site : Browser.Bookmarkcolumns.VISITS, whether it is bookmarked : Browser.Bookmarkcolumns.BOOKMARK etc.)
- Use **managedQuery()** method to retrieve info. for various columns with **5 arguments,** with 1st arg. URI, 2nd arg. Required columns, 3rd arg. Where clause

to retrieve specific calls only, 4th arg. Values for where clause ? parameter., 5th arg. Defines sorting order.

- Use cursor methods to do various operations on that data.
- Requires permission: <uses-permission> parent tag, android:name attribute, value : android.permission.READ_HISTORY_BOOKMARKS.

7) Write android application to retrieve network status, call state & service state.

To retrieve network status

- Create instance of ConnectivityManager using getSystemService() method, pass Context.CONNECTIVITY_SERVICE.

```
ConnectivityManager cm =
(ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);
```

- Use getNetworkInfo() method of ConnectivityManager instance and receive information in NetworkInfo ref. variable.

- For wifi network:

```
NetworkInfo ni = cm.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
```

- For Cellular network :

```
NetworkInfo ni = cm.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
```

- Use various methods of NetworkInfo class such as,
  - isAvailable(), isConnected() etc.
- Take permission to get network information → <uses-permission> tag, android:name attribute, android.permission.ACCESS_NETWORK_STATE value)

To retrieve call state

- Create instance of TelephonyManager using getSystemService() method, pass Context.TELEPHONY_SERVICE.

```
TelephonyManager telManager = (TelephonyManager)
getSystemService(Context.TELEPHONY_SERVICE);
```

- Use getCallState() method of TelephonyManger class to retrieve call state, Call can be at various state such as
  - `TelephonyManger.CALL_STATE_IDLE`
  - `TelephonyManger.CALL_STATE_OFFHOOK`
  - `TelephonyManger.CALL_STATE_RINGING`

- To continuously listen to the state use listen() method of TelephonyManager class, implement PhoneStateListener() & onCallStateChanged() method.

To retrieve service state

- Create instance of ServiceState.
- Use getState() method of ServiceState instance it can be in
    - ServiceState.STATE_EMERGENCY_ONLY
    - ServiceState.STATE_IN_SERVICE
    - ServiceState.STATE_OUT_OF_SERVICE
    - ServiceState.STATE_POWER_OFF

8) Write an android application that sends text message provided by user to given no. (Unit :-4, Chapter :- 16)

Sending SMS :

- Create SmsManager instance using getDefault() method.
- Use sendTextMessage() method SmsManager with five arguments where 1st arg. Is the phone no., 3rd arg. Is the text message to send.
- Take permission using <uses-permission> tag, android:name attribute, value → android.permission.SEND_SMS.

9) What information are required while packaging android application using Eclipse. Explain steps of packaging android application using eclipse. (Unit :- 5, Chapter :- 29)

- Setting application name & icon.
- Versioning the application
- Verifying the target platform
- Configuring the android manifest
- Disable debugging
- Creating new key, file location, password for managing the keystroke, Preliminary details.

10) Write an android application to design tabs for Gallery, List & Grid. Provide demonstration for all these widget in individual tab.

Creating Tabs:

- Prepare layout file with <TabHost> as parent tag & <TabWidget> as sub tag.
- Main activity class should extend TabActivity.
- Create instance of TabHost using getTabHost() method.
- Use newTabSpec() method to add new tab.
- Use setContent() method to set the content to that tab, pass intent as arguments.
- Use addTab() method of TabHost.
- Create activity for Gallery specify following code :

```
gallery = (Gallery) findViewById(R.id.gallery);
// String array holding the values
String[] text = new String[] { "Hello", "Hi", "Alloha", "Bonjour",
               "Hallo", "¡Hola" };
// Array adapter to display our values in the gallery control
ArrayAdapter<String> arr = new ArrayAdapter<String>(this,
android.R.layout.simple_gallery_item, text);
gallery.setAdapter(arr);
```

- Create activity for List specify the following code :

```
final String[] items = new String[] { "Item1", "Item2", "Item3",
                       "Item4" };
ArrayAdapter ad = new ArrayAdapter(
getApplicationContext(), android.R.layout.simple_list_item_1,
                       items);
list = (ListView) findViewById(R.id.List);
list.setAdapter(ad);
```

- Create activity for Grid specify the following code :

```
GridView gridview = (GridView) findViewById(R.id.gridview);
final String[] items = new String[] { "Item1", "Item2", "Item3",
                       "Item4", "Item5", "Item6", "Item7", "Item8" };

ArrayAdapter<String> ad = new ArrayAdapter<String>(
getApplicationContext(), android.R.layout.simple_list_item_1,
gridview.setBackgroundColor(Color.GRAY);
gridview.setNumColumns(2);
gridview.setGravity(Gravity.CENTER);
gridview.setAdapter(ad);
```