CSE 4027 - Mobile Application Development

Spinner, ListView

resources/strings.xml

```
<resources>
    <string name="app_name">AppWithSpinnerDemo</
        string>
    <string-array name="states">
        <item>Assam</item>
        <item>Bihar</item>
        <item>Chattisgadh</item>
        <item>Delhi</item>
        <item>Goa</item>
    </string-array>
</resources>
```

activity_main.xml

Include Spinner UI component in the XML file.

```
<Spinner

        android:id="@+id/spinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```

MainActivity.java I

```
package com.xyz.appwithspinnerdemo;

import android.widget.ArrayAdapter;
import android.widget.Spinner;
public class MainActivity extends AppCompatActivity {
  @Override
    protected void onCreate(Bundle savedInstanceState
        ) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sp=(Spinner)findViewById(R.id.spinner);
        states=getResources().getStringArray(R.array.
            states);

        aa=new ArrayAdapter<String>(this,android.R.
            layout.simple_spinner_item,states);
        aa.setDropDownViewResource(android.R.layout.
            simple_spinner_dropdown_item);
        sp.setAdapter(aa);
```

MainActivity.java II

```
}
public void onClick(View v)
{
    Toast.makeText(this,"Your state is "+sp.
        getSelectedItem(), Toast.LENGTH_LONG).show
        ();
}
```

ListView

activity_main.xml

Include Spinner UI component in the XML file.

```
<ListView
        android:id="@+id/spinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:divider="@color/green"
        android:listSelector="@color/green"
        android:dividerHeight="2dp"/>

    <Button
        android:onClick="onClick"
        android:text="Register"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />
```

MainActivity.java I

```
package com.xyz.spinnerdemo;

import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity
    implements AdapterView.OnItemClickListener {

    private String countries[]={"India","Russia","
        Nepal","Sri Lanka"};
    private ListView sp;
    private ArrayAdapter<String> aa;
    @Override
    protected void onCreate(Bundle savedInstanceState
        ) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

MainActivity.java II

```
        sp=(ListView)findViewById(R.id.spinner);
        aa=new ArrayAdapter<String>(this,android.R.
            layout.simple_spinner_item,getResources().
            getStringArray(R.array.countries));
        //aa.setDropDownViewResource(android.R.layout
            .simple_spinner_dropdown_item);
        sp.setAdapter(aa);
        sp.setOnItemClickListener(this);
    }
    protected void onClick(View v) {
        Toast.makeText(this,"Selected item is "+sp.
            getSelectedItem(),Toast.LENGTH_LONG).show
            ();
    }
    public void onItemClick(AdapterView<>v,View v2,
        int i,long l){

    }
}
```

Threads

Threads

Threads

- Application creates a thread of execution called main thread

Threads

- Application creates a thread of execution called main thread
- it is also called UI thread because through this thread system interacts with UI components

Threads

- Application creates a thread of execution called main thread
- it is also called UI thread because through this thread system interacts with UI components
- system does not create separate threads for components

Time Consuming tasks

When performing intensive work single thread model can yield poor performance

Time Consuming tasks

When performing intensive work single thread model can yield poor performance

Performing long operations such as network access or database queries will block the whole UI.

Time Consuming tasks

When performing intensive work single thread model can yield poor performance

Performing long operations such as network access or database queries will block the whole UI.

When the thread is blocked, no events can be dispatched

Time Consuming tasks

When performing intensive work single thread model can yield poor performance

Performing long operations such as network access or database queries will block the whole UI.

When the thread is blocked, no events can be dispatched

From the user's perspective application appears to hang

Time Consuming tasks

When performing intensive work single thread model can yield poor performance

Performing long operations such as network access or database queries will block the whole UI.

When the thread is blocked, no events can be dispatched

From the user's perspective application appears to hang

The user might quit and uninstall

An example

```
public void buttonClick(View view)
    {
      long endTime = System.currentTimeMillis() +
          20*1000;

    while (System.currentTimeMillis() < endTime) {
      synchronized (this) {
      try {
           wait(endTime - System.currentTimeMillis())
               ;
           } catch (Exception e) {
        }
      }
    }
    TextView myTextView = (TextView)findViewById(R.id
        .myTextView);
    myTextView.setText("Button Pressed");
    }
```

Two simple rules

Android UI toolkit is not thread safe.

Two simple rules

Android UI toolkit is not thread safe.

Should not manipulate UI from a worker thread.

Two simple rules

Android UI toolkit is not thread safe.

Should not manipulate UI from a worker thread.

Two simple rules

Two simple rules

Android UI toolkit is not thread safe.

Should not manipulate UI from a worker thread.

Two simple rules

1. Do not block the UI thread

Two simple rules

Android UI toolkit is not thread safe.

Should not manipulate UI from a worker thread.

Two simple rules

1. Do not block the UI thread
2. Do not access the Android UI toolkit from outside the UI thread

Worker Threads

```
public void onClick(View v) {
    new Thread(new Runnable() {
        public void run() {
            Bitmap b = loadImageFromNetwork("http://
                example.com/image.png");
            mImageView.setImageBitmap(b);
        }
    }).start();
}
```

It violates the second rule.

Solution

To fix this problem, Android offers several ways to access the UI thread from other threads. Here is a list of methods that can help:

Solution

To fix this problem, Android offers several ways to access the UI thread from other threads. Here is a list of methods that can help:

Activity.runOnUiThread(Runnable)

Solution

To fix this problem, Android offers several ways to access the UI thread from other threads. Here is a list of methods that can help:

Activity.runOnUiThread(Runnable)
View.post(Runnable)

Solution

To fix this problem, Android offers several ways to access the UI thread from other threads. Here is a list of methods that can help:

Activity.runOnUiThread(Runnable)
View.post(Runnable)
View.postDelayed(Runnable, long)

Fix

For example, you can fix the above code by using the View.post(Runnable) method:

```
public void onClick(View v) {
    new Thread(new Runnable() {
        public void run() {
            final Bitmap bitmap =
                    loadImageFromNetwork("http://
                        example.com/image.png");
            mImageView.post(new Runnable() {
                public void run() {
                    mImageView.setImageBitmap(bitmap)
                        ;
                }
            });
        }
    }).start();
}
```

Handler

Handler Object

```
import android.os.Handler;
import android.os.Message;

public class ThreadExample extends Activity {

  Handler handler = new Handler() {
      @Override
      public void handleMessage(Message msg) {
        TextView myTextView =
                        (TextView)findViewById(R.id.
                            myTextView);
        myTextView.setText("Button Pressed");
         }
     };
```

```
public void buttonClick(View view)
{

        Runnable runnable = new Runnable() {
            public void run() {

              long endTime = System.currentTimeMillis()
                   + 20*1000;

              while (System.currentTimeMillis() <
                  endTime) {
                synchronized (this) {
                  try {
                    wait(endTime - System.
                        currentTimeMillis());
                  } catch (Exception e) {}
                }

            }
        handler.sendEmptyMessage(0);
        }
        };
```

AyncTask

AyncTask

AyncTask

- AsyncTask enables proper and easy use of the UI thread.

AyncTask

- AsyncTask enables proper and easy use of the UI thread.
- allows to perform background operations and publish results on the UI thread

AyncTask

- AsyncTask enables proper and easy use of the UI thread.
- allows to perform background operations and publish results on the UI thread
- No manipulation of Threads / handlers

AsyncTask

AsyncTask

- computation runs in worker thread

AsyncTask

- computation runs in worker thread
- result is published on the UI thread

AsyncTask

- computation runs in worker thread
- result is published on the UI thread
- defined by three generic types called Params, Progress and Result

AsyncTask

- computation runs in worker thread
- result is published on the UI thread
- defined by three generic types called Params, Progress and Result
- four steps called onPreExecute, doInBackground,onProgressUpdate and onPostExecute

An Example

```
public void onClick(View v) {
    new DownloadImageTask().execute("http://example.
        com/image.png");
}

private class DownloadImageTask extends AsyncTask<
    String, Void, Bitmap> {
    /** The system calls this to perform work in a
        worker thread and
      * delivers it the parameters given to AsyncTask
          .execute() */
    protected Bitmap doInBackground(String... urls) {
        return loadImageFromNetwork(urls[0]);
    }

    /** The system calls this to perform work in the
        UI thread and delivers
      * the result from doInBackground() */
    protected void onPostExecute(Bitmap result) {
        mImageView.setImageBitmap(result);
    }
}
```

Another Example

```
private class DownloadFilesTask extends AsyncTask<URL
    , Integer, Long> {
     protected Long doInBackground(URL... urls) {
         int count = urls.length;
         long totalSize = 0;
         for (int i = 0; i < count; i++) {
             totalSize += Downloader.downloadFile(
                 urls[i]);
             publishProgress((int) ((i / (float)
                 count) * 100));
             // Escape early if cancel() is called
             if (isCancelled()) break;
         }
         return totalSize;
     }

     protected void onProgressUpdate(Integer...
         progress) {
         setProgressPercent(progress[0]);
     }
```

Menus

Menus

- menus are common user interface components
- Beginning with Android 3.0 (API level 11), Android-powered devices are no longer required to provide a dedicated Menu button.

Types of menus

- Option menu and app bar
  The options menu is the primary collection of menu items for an activity. It's where you should place actions that have a global impact on the app, such as "Search," "Compose email," and "Settings."
- Context menu and contextual action mode
  A context menu is a floating menu that appears when the user performs a long-click on an element. It provides actions that affect the selected content or context frame. The contextual action mode displays action items that affect the selected content in a bar at the top of the screen and allows the user to select multiple items.

Types of menus

- Popup menu
  A popup menu displays a list of items in a vertical list that's anchored to the view that invoked the menu. It's good for providing an overflow of actions that relate to specific content or to provide options for a second part of a command. Actions in a popup menu should not directly affect the corresponding content-that's what contextual actions are for. Rather, the popup menu is for extended actions that relate to regions of content in your activity.

Defining a menu in XML

For all menu types, Android provides a standard XML format to define menu items. Instead of building a menu in your activity's code, you should define a menu and all its items in an XML menu resource. You can then inflate the menu resource (load it as a Menu object) in your activity

Menus in XML

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/
    res/android">
    <item android:id="@+id/new_game"
          android:icon="@drawable/ic_new_game"
          android:title="@string/new_game"
          android:showAsAction="ifRoom"/>
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"
          android:title="@string/help" />
</menu>
```

Adding Submenus

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/
    res/android">
    <item android:id="@+id/file"
          android:title="@string/file" >
        <!-- "file" submenu -->
        <menu>
            <item android:id="@+id/create_new"
                  android:title="@string/create_new"
                  />
            <item android:id="@+id/open"
                  android:title="@string/open" />
        </menu>
    </item>
</menu>
```

Adding menu to Activity

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.game_menu, menu);
    return true;
}
```

handling Option menu

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection
    switch (item.getItemId()) {
        case R.id.new_game:
            newGame();
            return true;
        case R.id.help:
            showHelp();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Context Menu

Creating a Context Menu

1. Register the View to which the context menu should be associated by calling registerForContextMenu() and pass it the View.
   If your activity uses a ListView or GridView and you want each item to provide the same context menu, register all items for a context menu by passing the ListView or GridView to registerForContextMenu().

2. ```
   @Override
   public void onCreateContextMenu(ContextMenu menu,
        View v,
                                   ContextMenuInfo
                                       menuInfo) {
       super.onCreateContextMenu(menu, v, menuInfo);
       MenuInflater inflater = getMenuInflater();
       inflater.inflate(R.menu.context_menu, menu);
   }
   ```
   ```

3.
```
@Override
public boolean onContextItemSelected(MenuItem
    item) {
    AdapterContextMenuInfo info = (
        AdapterContextMenuInfo) item.getMenuInfo
        ();
    switch (item.getItemId()) {
        case R.id.edit:
            editNote(info.id);
            return true;
        case R.id.delete:
            deleteNote(info.id);
            return true;
        default:
            return super.onContextItemSelected(
                item);
    }
}
```

Notifications

Notifications

A notification is a message one can display to the user outside of application's normal UI.

Notifications

A notification is a message one can display to the user outside of application's normal UI.

When you tell the system to issues a notification, it first appears as an icon in the notification area.

Notifications

A notification is a message one can display to the user outside of application's normal UI.

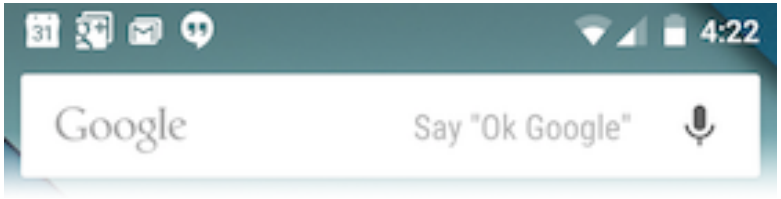
When you tell the system to issues a notification, it first appears as an icon in the notification area.

To see the details of the notification, the user opens the notification drawer.

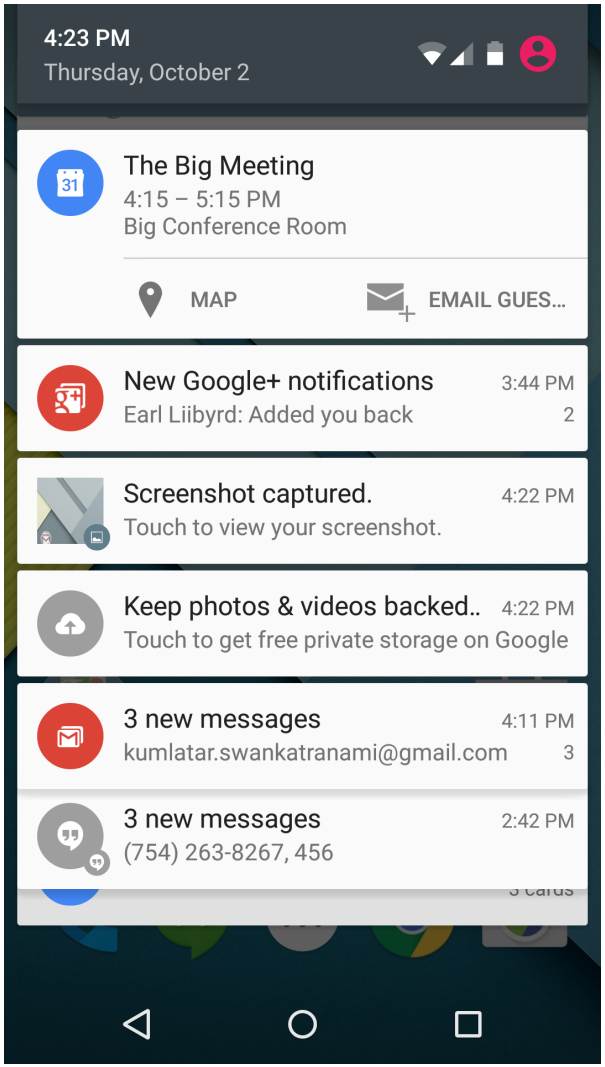
Notifications

Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.

Notification Area



Notification Drawer



Creating a Notification

We use NotificationCompat.Builder class introduced in version 4. The class Notification.Builder was added in Android 3.0

- The UI information and actions for a notification in a NotificationCompat.Builder object.
- To create the notification call NotificationCompat.Builder.build() which returns a Notification object.
- To issue the notification, pass the Notification object to the system by calling NotificationManager.notify().

Required Notification Contents

A Notification object must contain the following

- A small icon, set by setSmallIcon()
- A title, set by setContentTitle()
- Detailed text, set by setContentText()

Notification Actions

You should add atelast one action to your notification.

Notification Actions

You should add atelast one action to your notification.

An action allows users to go directly from the notification to an Activity in your application, where they can look at one or more events or do further work.

Notification Actions

You should add atelast one action to your notification.

An action allows users to go directly from the notification to an Activity in your application, where they can look at one or more events or do further work.

You can also add buttons to the notification that perform additional actions such as snoozing an alarm or responding immediately to a text message

Notification actions

Inside a Notification, the action is defined by a PendingIntent containing an Intent that starts an Activity in your application.

Notification actions

Inside a Notification, the action is defined by a PendingIntent containing an Intent that starts an Activity in your application.

If you want to start Activity when the user clicks the notification text in the notification drawer, you add the PendingIntent by calling setContentIntent().

An example

```
NotificationCompat.Builder mBuilder = new
    NotificationCompat.Builder(this, CHANNEL_ID)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle(textTitle)
        .setContentText(textContent)
        .setPriority(NotificationCompat.
            PRIORITY_DEFAULT);
```

Notification Channel

```
private void createNotificationChannel() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES
        .O) {
        CharSequence name = getString(R.string.
            channel_name);
        String description = getString(R.string.
            channel_description);
        int importance = NotificationManager.
            IMPORTANCE_DEFAULT;
        NotificationChannel channel = new
            NotificationChannel(CHANNEL_ID, name,
            importance);
        channel.setDescription(description);
        NotificationManager notificationManager =
            getSystemService(NotificationManager.class
            );
        notificationManager.createNotificationChannel
            (channel);
    }
}
```

```
// Create an explicit intent for an Activity in your
    app
Intent intent = new Intent(this, AlertDetails.class);
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
    Intent.FLAG_ACTIVITY_CLEAR_TASK);
PendingIntent pendingIntent = PendingIntent.
    getActivity(this, 0, intent, 0);

NotificationCompat.Builder mBuilder = new
    NotificationCompat.Builder(this, CHANNEL_ID)
        .setSmallIcon(R.drawable.notification_icon)
           ............
        .setContentIntent(pendingIntent)
        .setAutoCancel(true);
```

Broadcast Receivers

Broadcast Receivers

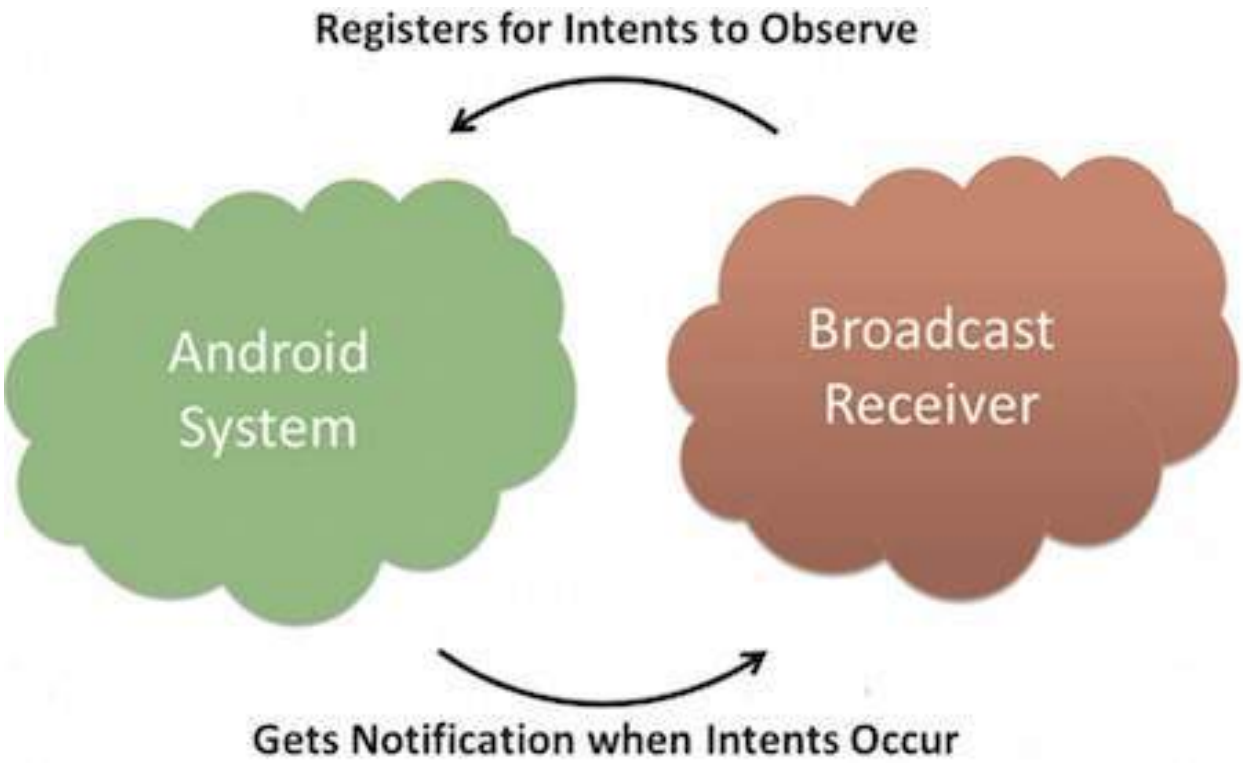
- Respond to broadcast messages from other applications or from the sytem
- Applications can also initiate broadcasts to other application
- Broadcast receiver will interept this communication and will initiate appropriate action

Creating a Broadcast Receiver

A broadcast receiver is implemented as a subclass of BroadcastReceiver class, where each message is received as a Intent object parameter.

```
public class MyReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent
        intent) {
       Toast.makeText(context, "Intent Detected.",
           Toast.LENGTH\_LONG).show();
    }
}
```

Broadcast Receiver



Registering Broadcast Receiver

```
<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <receiver android:name="MyReceiver">

        <intent-filter>
            <action android:name="android.intent.action.
                BOOT_COMPLETED">
            </action>
        </intent-filter>

    </receiver>


</application>
```

Standard Broadcast Actions I

Action	Description
ACTION_TIME_TICK	The current time has changed. Sent every minute. You cannot receive this through components declared in manifests, only by explicitly registering for it with Context#registerReceiver(BroadcastRec IntentFilter).
ACTION_TIME_CHANGED	The time was set
ACTION_TIMEZONE_CHANGED	The timezone has changed
ACTION_BOOT_COMPLETED	This is broadcast once, after the user has finished booting. You must hold the Manifest.permission.RECEIVE_BOOT_ permission in order to receive this broadcast.

Standard Broadcast Actions II

Action	Description
ACTION_PACKAGE_ADDED	A new application package has been installed on the device.
ACTION_PACKAGE_CHANGED	An existing application package has been changed (for example, a component has been enabled or disabled).
ACTION_PACKAGE_REMOVED	An existing application package has been removed from the device.
ACTION_PACKAGE_RESTARTED	The user has restarted a package, and all of its processes have been killed.
ACTION_PACKAGE_DATA_CLEARED	The user has cleared the data of a package
ACTION_PACKAGES_SUSPENDED	Packages have been suspended

Standard Broadcast Actions III

Action	Description
ACTION_PACKAGES_UNSUSPENDED	Packages have been unsuspended
ACTION_UID_REMOVED	A user ID has been removed from the system
ACTION_BATTERY_CHANGED	This is a sticky broadcast containing the charging state, level, and other information about the battery
ACTION_POWER_CONNECTED	External power has been connected to the device
ACTION_POWER_DISCONNECTED	External power has been removed from the device
ACTION_SHUTDOWN	Device is shutting down

Broadcasting Custom Intents

If the application has to generate and send custom intents then you will have to create and send intents by using sendBroadcast() method.

```
public void broadcastIntent(View view)
{
    Intent intent = new Intent();
    intent.setAction("com.mahe.example.CUSTOM_INTENT")
        ;
    sendBroadcast(intent);
}
```

Registering Custom BroadcastReceiver

```
<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <receiver android:name="MyReceiver">

        <intent-filter>
            <action android:name="com.mahe.example.
                CUSTOM_INTENT">
            </action>
        </intent-filter>

    </receiver>
</application>
```

Telephony and SMS APIs

How to make a Phone call

```
Intent intent = new Intent(Intent.ACTION_DIAL);
intent.setData(Uri.parse("tel:" + number));
startActivity(intent);
```

How to make a Phone call

```
Intent intent = new Intent(Intent.ACTION_DIAL);
intent.setData(Uri.parse("tel:" + number));
startActivity(intent);
```

No need of permission

How to place a Call directly

Add the following permission

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

How to place a Call directly

Add the following code

```
public void dialPhone(View view){
Intent intent = new Intent(Intent.ACTION_CALL);
intent.setData(Uri.parse("tel:0123456789"));
startActivity(intent);
}
```

SMS Messages

Sending SMS Messages

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

Sending SMS Messages

```
SmsManager smsManager = SmsManager.getDefault();
smsManager.sendTextMessage(phoneNumber, null, msg,
null, null);
```

Flat Files

Flat Files

- Flat files are used to persist unstructured data-primitive and complex
- Android uses Java Input-Output API to read and write flat files

Opening, Writing and Closing a file

```
private void writeToFile(String text) {
  try {
    OutputStreamWriter outputStreamWriter = new
        OutputStreamWriter(openFileOutput("
userinput.txt", Context.MODE_PRIVATE));
    outputStreamWriter.write(text);
    outputStreamWriter.close();
  }catch(IOException e) {
    e.printStackTrace();
  }
}
```

File Location

The userinput.txt file gets created inside /data/data/<package>/files folder.

Opening, Reading and Closing a file

```
private String getFileContent() {
  String fileContent = "";
  try {
    InputStream is = openFileInput("userinput.txt");
    if(inputStream != null) {
      InputStreamReader isr = new InputStreamReader(
          is);
      BufferedReader br = new BufferedReader(isr);
      String line="";
      StringBuilder sb=new StringBuilder();
      while((line = br.readLine()) != null) {
        sb.append(line);
    }
    is.close();
    fileContent = sb.toString();
  }
} catch(FileNotFoundException e) {
  e.printStackTrace();
}catch(IOException e) {
  e.printStackTrace();
}
```

File Storage in External(SDCard)

Opening, Writing and Closing a file

```
private void writeToFile(String text) {
  try {
    File sdCard=Environment.
        getExternalStorageDirectory();
    if(sdCard.exists() && sdCard.canWrite()) {
      File newFolder = new File(sdCard.
          getAbsolutePath()+"/FolderName");
      newFolder.mkdir();
      if(newFolder.exists() && newFolder.canWrite())
          {
        File textFile = new File(newFolder.
            getAbsolutePath()+"/userinput.txt");
        textFile.createNewFile();
        if(textFile.exists() && textFile.canWrite())
            {
          FileWriter fw = new FileWriter(textFile);
          fw.write(text);
          fw.flush();
          fw.close();
        }
      }
```

Reading a File from External Storage

```
private String getFileContent() {
  String fileContent = "";
  File sdCard = Environment.
      getExternalStorageDirectory();
  if(sdCard.exists() && sdCard.canRead()) {
    File appFolder = new File(sdCard.getAbsolutePath
        ()+"/FolderName");
    if(appFolder.exists() && appFolder.canRead()) {
      File textFile=new File(appFolder.
          getAbsolutePath()+"/userinput.txt");
          ....
}
```

Shared Preferences

Shared Preferences

- Shared preferences is a solution to store primitive data in key-value pairs
- Key-value pairs are used to save user preferences such as ringtone, user-preferred app settings etc.
- provide support for persisting boolean, float, int, long and String data types
- Shared preferences stores data in an XML file in the internal memory of the device

An Example

```
SharedPreferences preferences = getSharedPreferences(
    "SMSPreferences", Context.MODE_PRIVATE);
Editor ed = preferences.edit();
ed.putBoolean("SendSMS", chkEnable.isChecked());
ed.putString("Message", et.getText().toString());
ed.putString("Signature", etSign.getText().toString()
    );
ed.commit();
```

Shared Preferences file

File gets stored in /data/data/<package>/shared_prefs as an XML file

```
<?xml version="1.0" encoding="utf-8" standalone="yes"
    ?>
<map>
  <string name="Message">Will call you back later</
      string>
  <string name="Signature">Auto SMS</string>
  <boolean name="SendSMS" value="true"/>
</map>
```

Methods offered by Editor

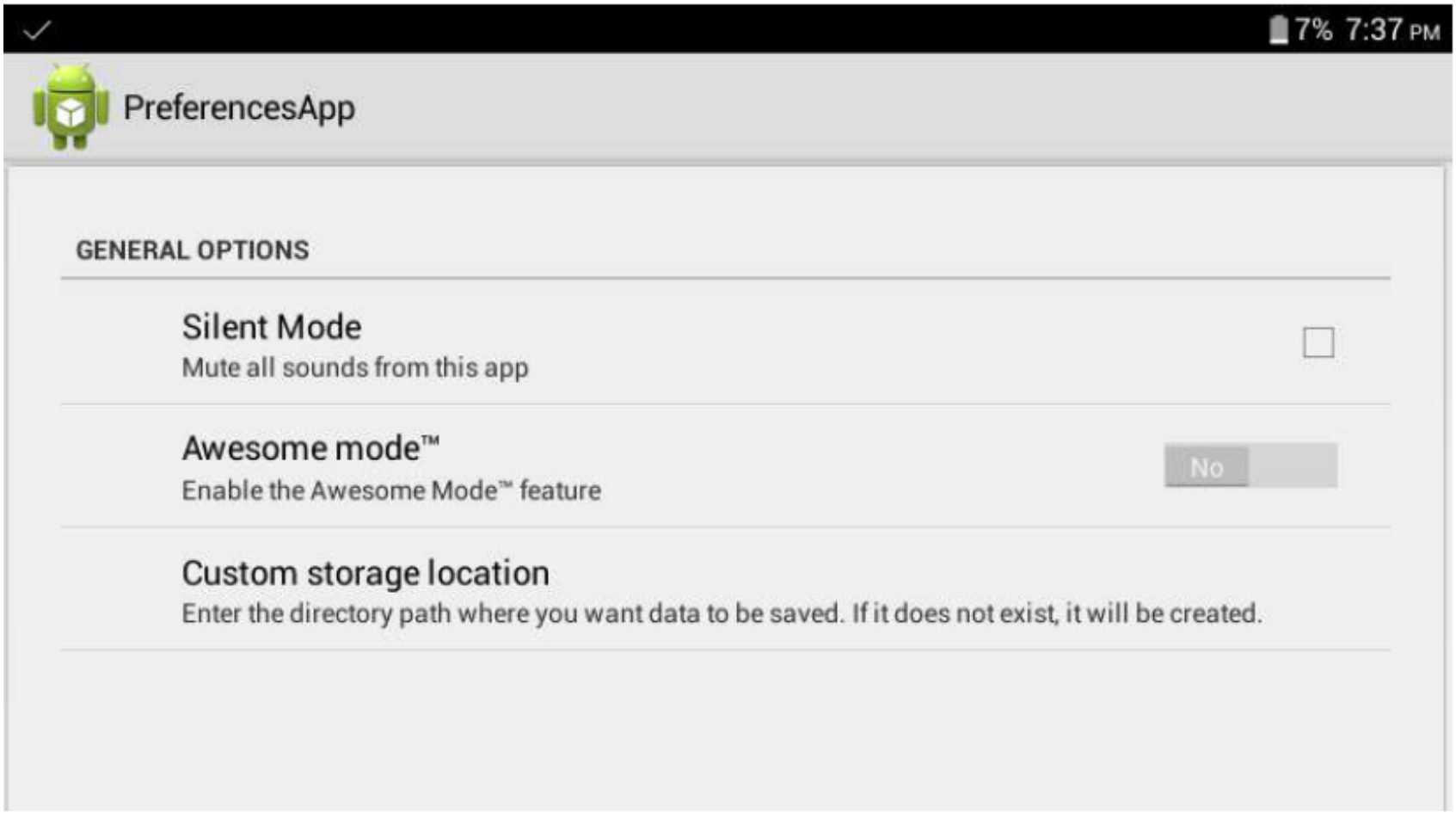
putBoolean(String key, boolean value)	Saves a boolean value against a key
putFloat(String key, float value)	Saves a float value against a key
putInt(String key, int value)	Saves an integer value against a key
putLong(String key, long value)	Saves a long value against a key
putString(String key, String value)	Saves a string value against a key
putStringSet(String key, Set values)	Saves a set of String values against a key

Reading Shared Preferences

```
SharedPreferences preferences = getSharedPreferences(
    "SMSPreferences", Context.MODE_PRIVATE);
boolean sendSms= preferences.getBoolean("SendSMS",
    false);
String message=ed.getString("Message", "");
String sign=ed.getString("Signature", "");
//Send SMS to the caller
```

Implementing a Settings Screen

Preferences Screen



res/xml/preferences.xml I

```
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/
    android">
<PreferenceCategory
android:title="General options">
<CheckBoxPreference
android:key = "silent_mode"
android:defaultValue="false"
android:title="Silent Mode
android:summary="Mute all sounds from this app" />
<SwitchPreference
android:key="awesome_mode"
android:defaultValue="false"
android:switchTextOn="Yes"
android:switchTextOff="No"
android:title="Awesome mode"
android:summary="Enable the Awesome Mode feature"/>
<EditTextPreference
android:key="custom_storage"
```

res/xml/preferences.xml II

```
android:defaultValue="/sdcard/data/"
android:title="Custom storage location"
android:summary="Enter the directory path where you
    want data to be saved. If it does
not exist, it will be created."
android:dialogTitle="Enter directory path (eg. /
    sdcard/data/ )"/>
</PreferenceCategory>
</PreferenceScreen>
```

Activity I

```
package com.example.preferences;
import android.preference.PreferenceActivity;
import android.os.Bundle;
public class PreferencesActivity extends
    PreferenceActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
addPreferencesFromResource(R.xml.preferences);
}
}
```

Relational Data

Relational Data

- So far we learnt to store unstructured, primitive data on the device using file,shared preferences
- reduced data redundancy, atomicity, consistency, isolation, durability and querying ability
- these databases have to be light weight, utilize limited processing power
- Android provides SQLite database
- SQLite is light weight(around 500KB)
- SQLite runs as part of the process

An Example

```
public class StringDBAdapter {
  private static final String DB_NAME="
      String_Database.db";
  private static final String TABLE_NAME="
      String_Table";
  private static final int DB_VERSION=1;
  private static final String KEY_ID="id";
  private static final String COLUMN_STRING="String";
  private static final String TABLE_CREATE="create
      table "+TABLE_NAME+"("+KEY_ID+" integer primary
      key autoincrement ,"+COLUMN_STRING+" text not
      null);";
  private SQLiteDatabase stringDatabase;
  private final Context context;
  private MyDBHelper helper;
```

```
public StringDBAdapter(Context context)
{
  this.context=context;
  helper=new MyDBHelper(context,DB_NAME,null,
      DB_VERSION);
}
private static class MyDBHelper extends
    SQLiteOpenHelper
{
  public MyDBHelper(Context context, String name,
      CursorFactory cursorFactory, int version){
  super(context,name,cursorFactory,version);
}
@Override
public void onCreate(SQLiteDatabase db) {
  db.execSQL(TABLE_CREATE);
}
```

```
@Override
public void onUpgrade(SQLiteDatabase db, int
    oldVersion, int newVersion){
  Log.w("Updation","Database version is being updated
      ");
  db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);
  onCreate(db);
}
public StringDBAdapter open() {
  stringDatabase=helper.getWritableDatabase();
  return this;
}
public void close() {
  stringDatabase.close();
}
```

Query

```
public Cursor getAllRecords() {
  return db.query(TABLE_NAME,null,null,null,null,null
      ,null);
}

public Cursor getRecordsWithinRange(int gid) {
  return db.query(TABLE_NAME,null,KEY_ID+"<="+gid,
      null,null,null,null);
}
```

Displaying values from Table

```
recs=db.getAllRecords();

stringlist=new ArrayList<Integer>();
while(recs.moveToNext()) {
  stringlist.add(recs.getString(0));
}
stringAdapter = new ArrayAdapter<String>(
    getApplicationContext(),android.R.layout.
    simple_list_item_1,
expAmount);
lv.setAdapter(stringAdapter);
```

Sum of a column

```
public int getTotalRecords() {
  Cursor strCursor = db.rawQuery("SELECT COUNT(id)
      FROM "+TABLE_NAME,null);
  if(strCursor != null) {
    strCursor.moveToNext();
    return strCursor.getInt(0);
  }
  else
    return 0;
```

Adding a tuple

```
public long addString(String str) {
  ContentValues cv=new ContentValues();
  cv.put(COLUMN_STRING, str);
  return db.insert(TABLE_NAME,null,cv);
}
```

Deleting a tuple

```
public boolean deleteRecord(long id) {
  return db.delete(TABLE_NAME,KEY_ID+"="+id,null)>0;
}
```

Updating a tuple

```
public int updateString(long id, String str) {
  ContentValues cv = new ContentValues();
  cv.put(COLUMN_STRING, str);
  return db.update(TABLE_NAME, cv,KEY_ID+"="+id,null)
      ;
}
```