STUDENT NUMBER: C18442284 NAME: AMIR AKBARI

MOBILE SOFTWARE DEVELOPMENT CMPU3026

Q1:

(a):

for handle inputs we use Event Listeners and Event Handling in android applications. It's for user interactions and to extend a View classes. Therefore we can use event handler method when using ListActivity as subclass for example the click button.

CODE SAMPLE:

```
Public void onListItemClick ( ListView l , View v, int position, long id)
{
        String selection = l.getItemAtPosition(position).toString();
        // whatever you want to do with the data returned.
        // getItemAtPosition() returns an object.. in this case, we know
        // it's just one item, so we convert to a String
}
```

(b):

A way to guaranteeing behaviour accurse a set of unrelated classes, if a class implements an interface, can invoke the i/f method safely. And some interfaces are useful for "tagging" classes. (behaviour that a class "signs" up) for example, in the listener interfaces, assigning that listener to the widget that takes the user action

So basically we implement the "listener" as we needed then we assigning that listener to the widget that takes the user action(e.g. button) and we implementing the "behavior" we wanted when the use action was taken (e.g. button clicked). Also there could be a slightly change if we want to use more than one button. So basically the onClick method will be shared across more than one button so that means we need to use the view object getID method to check which button was clicked.

CODE SAMPLE:

```
public class MyPass extends Activity{
        implements View.OnClickListener

        Button btn;
        @Override
        public void onCreate (Bundle icicle){
        //oncreate code

        btn = (Button)findViewbyId(R.id.whateverbuttoniscalled)
        btn.setOnClickListener(this


}
        public void onClick (View view )
        // do whatever you want as a result of the
        // the button Click;
}
```

Q1:
(c):
(i):
Model = classes that deal with storing and using data
If data structure(s) or rules change. Just change these classes
Can have multiple different GUIs using the same model
View = classes or code that deal with the GUI part
If the GUI layout changes, don’t (necessarily) need to change the model or control classes
Controller = (non graphical) classes that bridge the View and the Model Keep separate
Listener classes that trigger behaviour
Generally. Consists of everything that is not the layout or not directly storing/using data
Embed event handler method into XML
Simplest to implement +But
1. Presentation coupled with logic- bad –
2. Changes to method name - > need to remember to refactor the xml –
3. Multiple XML files using a single method can lead to maintenance problems if functionality diverges -

(ii):

(iii):

(d):
Android phones have different screen sizes and different screen densities and also some people would use the android tablets too. So screen size is one the most important factor which developers cant skip. For example the screen size is the actual physical screen size, measured as the screen's diagonal. In android we have different sizes (e.g. small, normal, large and …). Also the screen density is another one which is the quantity of pixels within a physical area of the screen.
This sizes would change over the time as device usage/popularity changes.
The other important is the android version a developer has to check the version before creating the app, also developers has to make sure has two layout which if one users want to use tablets, so 2 layouts one for mobile and one fore tablets.

Q2:
(a):
Use ArrayAdapter and arrayList and custom adapters
Custom adapter can use to control the output.
To populate the a custom row layout, can still use an existing off the shelf Adapter for simple cases.
Basically its about the kind of data that needs to be handled, adapters job is to manage the supply of data and convert it into the display specified in the layout, so therefore developers can use custom adapters. What that means, that means developer has to take three steps create a simple row layouts the custom the row layouts and also custom adapters.
First step to display a simple one column list of text items we need 1 XML Screen layout which Includes<ListView> which for example we can call it row.xml.
Also need to create your own Adapter (override getView and use convertView). Can also set a cursor adapter to dump results into a list.

(b):
(i):

```
ContentValues obj = new ContentValues(20)
obj.Put (string key1, float value1);
obj.Put (string key2, float value2);
obj.clear ()
```

(ii):
Insert and update.
Insert: Used to build up a set of data for the INSERT statement also Pass in the fields ”put ” in against the columns and insert. Underneath the covers, Android deals with putting it into the
INSERT statement.
also can use it for updates. Need to tell the system the new values and which row(s) to update
Underneath the covers, Android deals with putting it into the Sql update statement

(iii):

```
{
	"employees": [
	{ "firstName":"John" , "lastName":"Doe" } ]
}
```

____

```
Public void writeJSON()
{
JSONObject object = new JSONObject ();
try {
object.put("firstName":"John");
object.put("lastName":"Doe"); }
catch (JSONException e)
{
e.printStackTrace (); }
System.out.println(object);}
```

(c):

```
Public class adapter extends Arrayadapter <String>{
        Public myadapter( Context context, int textViewID, String[] object)
        {
                super(context, textViewID, object) ;
        }

        Public View getView(int position, View convertView, ViewGroup parent){

///setting the rows
        LayoutInflater inflater=getLayoutInflater();
       View row=inflater.inflate(R.layout.row, parent, false);

///setting the cites
TextView City = (TextView)row.findViewByID(R.id.cities);
City.setText(cities[position]);

///setting the Temperature
TextView Temp = (TextView)row.findViewByID(R.id.Temperature);
Temp.setText(Temperature[position]);

///setting the Weather
TextView Weather = (TextView)row.findViewByID(R.id.Weather);
Weather.setText(Weather[position]);

///setting the Image
ImageView myImage = (ImageView)row.findViewById(R.id.imageFIleNames);
```

Q3:
(a):
By using android lifecycle methods, such as onPause() and onResume().
onPause() can use in onCreate() for location tracking to restart update location, and onResume() can use in onCreate() for location tracking to stop update location.

(b):
In android consider to use of threads. The subclass call AsyncTask. Must override the doInBackGround() method, which runs on separate thread. Eg. When getWriteableDatabase() to connect to the database, it takes long time, so can pushed to background processing.
Basically AsyncTask creates the stuff In the background threads rather than using the main one.

SAMPLE CODE:
```
private class MyLongTask extends AsyncTask
{
@Override
protected Object doInBackground(final Object... objects)
{
try
{
openHelperRef.getWritableDatabase();
}
catch (etc)
} }
```

(c):

(i):
3 methods: OnLocationChanged, OnProviderDisabled, OnProviderEnabled.
This three method must be implemented to order using the LocationListener in Activity.
onLocationChnaged: onLocationChanged(Location location)
onProviderDisabled: onProviderDisabled(String provider)
OnProviderEnabled: onProviderEnabled(String provider)

Also there is one more method which is called onStatusChanged but This method was deprecated in API level 29

(ii):
minTime: Wise choice of value for minTime will help you reduce power consumption.
Elapsed time between location updates will never be less than minTime , but it can be greater, because it is influenced by the implementation of each given Provider and the update interval requested by other running applications.

minDistance:
minDistance Unlike minTime this parameter can be turned off by setting its value to 0. However, if minDistance is set to a value greater than 0, location provider will only send updates to your application if location has changed at least by given distance.
This parameter is not a great power saver like
minTime , but it should be kept in mind
although.

public void requestLocationUpdates (String provider, long 120000, float 5000, LocationListener listener)

ref:
long 120000 = 2min;
float 5000= 5km;