## Week - 1: Installation of Java Wireless Toolkit (J2ME)

1) If the Java Development Kit (JDK) is not there or only having the Java Runtime Environment (JRE) installed, install the latest JDK from http://java.sun.com/javase/downloads/index.jsp. Current stable release of Java is JDK 6 Update 7 but check the web page in case there are newer non-beta releases available.

2) Next, download the **Java Wireless Toolkit** (formerly called J2ME Wireless Toolkit) from: http://java.sun.com/products/sjwtoolkit/download.html.

3) Run the installer (for example, for Windows it is: sun_java_wireless_toolkit- 2_5_2-windows.exe). The installer checks whether a compatible Java environment has been pre-installed. If not, it is necessary to uninstall old versions of Java and perform Step 1 again.

Once after successful installation of Java and the tool kit compile this program and run the following program in the toolkit.

Steps to run this program in toolkit:

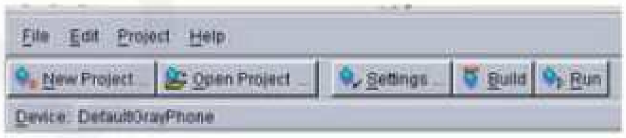
1. Start -> All Programs -> Sun Java Tool Kit -> Wireless Tool Kit
2. Click New Project – Enter Project Name -> Enter Class Name -> Click on Create Project.
3. Choose appropriate API Selection and Configurations.
4. Place Java Source file in WTK2.1 / WTK2.2\apps\projectname\src
5. Build the Project.
6. Run the Project.

```
import javax.microedition.lcdui.*; import javax.microedition.midlet.*;
public class HelloWorld extends MIDlet{ private Form form; private Display display;
public HelloWorld(){ super(); }
public void startApp(){ form = new Form("Hello World"); String msg = "Hello World!!!!!!!";
form.append(msg); display = Display.getDisplay(this); display.setCurrent(form); }
public void pauseApp(){} public void destroyApp(boolean unconditional){ notifyDestroyed();
} }
```

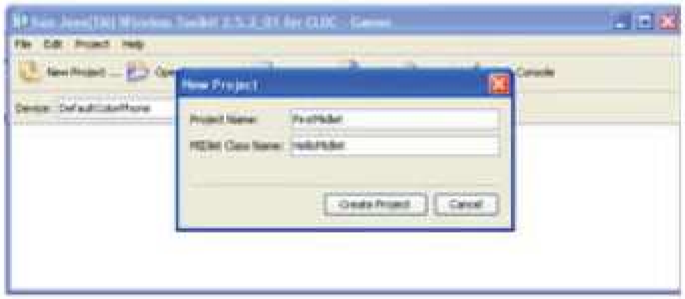
## Printing Hello World program in J2ME

**Printing Hello World program in J2ME**

**Step-1:**-Start ->AllPrograms->Sun Java Tool Kit->Wireless Tool Kit

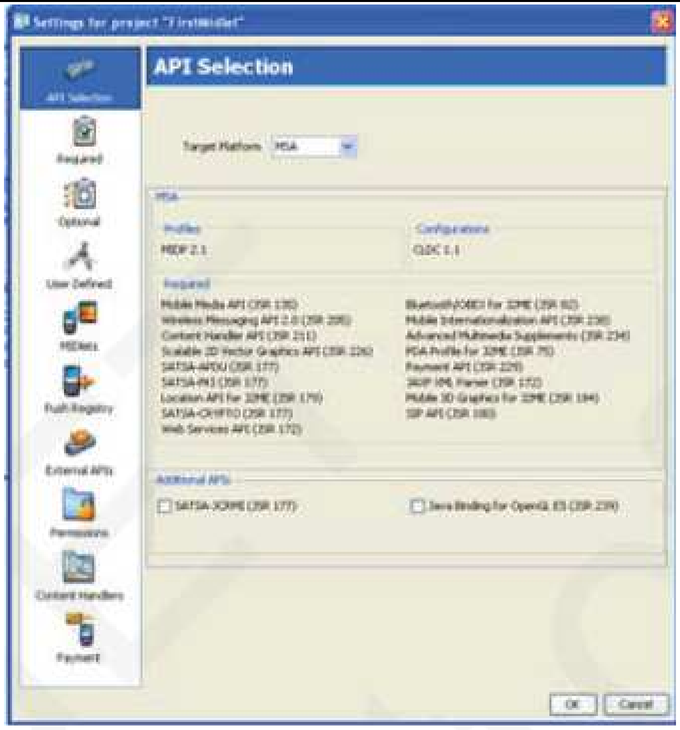


**Step-2:**-Click New Project –Enter project Name as FirstMidlet -> Enter ClassName as HelloMidlet->click on Create Project



Project Name: FirstMidlet
MIDlet Class Name: HelloMidlet

**Step-3:**- A setting window will open up. Accept the defaults by clicking ok in that window.


Settings for project "FirstMidlet"

API Selection

Target Platform: MSA

MSA

Profiles: MIDP 2.1

Configurations: CLDC 1.1

Required

Mobile Media API (JSR 135)
Wireless Messaging API 2.0 (JSR 205)
Content Handler API (JSR 211)
Scalable 2D Vector Graphics API (JSR 226)
SATSA-APDU (JSR 177)
SATSA-PKI (JSR 177)
Location API for J2ME (JSR 179)
SATSA-CRYPTO (JSR 177)
Web Services API (JSR 172)
Bluetooth/OBEX for J2ME (JSR 82)
Mobile Internationalization API (JSR 238)
Advanced Multimedia Supplements (JSR 234)
PDA Profile for J2ME (JSR 75)
Payment API (JSR 229)
JAXP XML Parser (JSR 172)
Mobile 3D Graphics for J2ME (JSR 184)
SIP API (JSR 180)

Additional APIs

☐ SATSA-JCRMI (JSR 177)
☐ Java Binding for OpenGL ES (JSR 239)

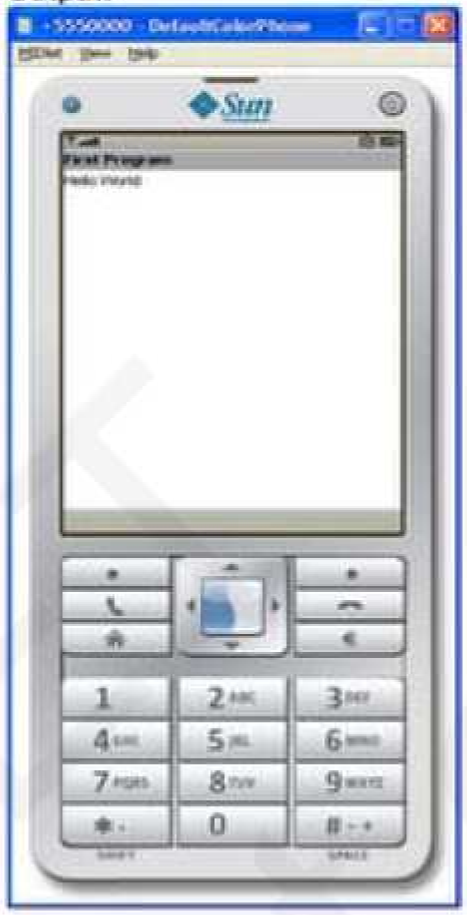
**Step-4:**-Write Following Code in Notepad and save it as HelloMidlet.java

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class HelloMidlet extends MIDlet {
public HelloMidlet() {
}
public void startApp() {
Form form = new Form( "First Program" );
form.append( "Hello World" );
Display.getDisplay(this).setCurrent( form );
}
public void pauseApp() {
}
public void destroyApp( boolean unconditional ) {
}
}
```

**Step-5:**-Place HelloMidlet.java in C:\Documents and settings\ADMIN\j2mewtk\2.5.2\apps\FirstMidlet\src\

**Step-6** :In the ktoolbar main window click on the "Build" button. When the build compiles successfully then click on the "Run" button.

Output:



+5550000 - DefaultColorPhone

First Program

Hello World

## WEEK -1
## WRITE A J2ME PROGRAM TO SHOW HOW TO CHANGE FONT SISE AND COLOR

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import java.io.*;
import java.lang.*;
import javax.microedition.io.*;
import javax.microedition.rms.*;
 public class changeFont extends MIDlet { public static final boolean COLOR = false;
 public static final boolean DEBUG = false; public static final int WHITE = 0xFFFFFF;
    public static final int BLACK = 0x000000;
    public static final int BLUE = 0x0000FF;
    public static final int LIGHT_GRAY = 0xAAAAAA;
    public static final int DARK_GRAY = 0x555555;


    private Display myDisplay = null;

           private DecodeCanvas decodeCanvas = null;

           private boolean painting = false;

    public changeFont() {
```

```
    myDisplay = Display.getDisplay(this);
    decodeCanvas = new DecodeCanvas(this);

}


public void startApp() throws MIDletStateChangeException {
    myDisplay.setCurrent(decodeCanvas);
}


public void pauseApp() {


}


protected void destroyApp(boolean unconditional)
        throws MIDletStateChangeException {
}


class DecodeCanvas extends Canvas {
    private changeFont parent = null;

    private int width = getWidth();
    private int height = getHeight();


    public DecodeCanvas(changeFont parent) {
        this.parent = parent;

         }

    public void paint(Graphics g) {


        g.setColor(WHITE);
        g.fillRect(0, 0, width, height);


        Font f1 = Font.getFont(Font.FACE_SYSTEM, Font.STYLE_PLAIN, Font.SIZE_LARGE);
        Font f2 = Font.getFont(Font.FACE_SYSTEM, Font.STYLE_PLAIN, Font.SIZE_MEDIUM);
        Font f3 = Font.getFont(Font.FACE_SYSTEM, Font.STYLE_PLAIN, Font.SIZE_SMALL);
        int yPos = 0;
        if (COLOR)
            g.setColor(BLUE);
        else
            g.setColor(LIGHT_GRAY);

                                        g.fillRect(0, yPos, width, f1.getHeight());

        if (COLOR)
            g.setColor(WHITE);
        else
```

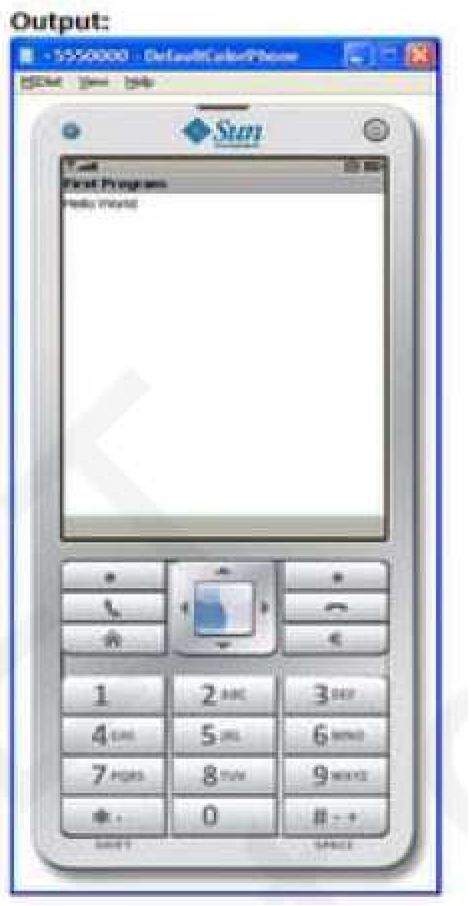
```
            g.setColor(BLACK);
                              g.setFont(f1);
          g.drawString("BIG FONT", 0, yPos, Graphics.LEFT | Graphics.TOP);
                              yPos = yPos + f1.getHeight() + 10;
          g.setFont(f2);
                    //        g.drawLine(0, f1.getHeight() + yPos - 1, width, f1.getHeight() + yPos - 1);
          g.drawString("MEDIUM FONT", 0, yPos, Graphics.LEFT | Graphics.TOP);
          g.setColor(BLACK);
          //g.drawLine(0, f2.getHeight() + yPos - 1, width, f2.getHeight() + yPos - 1);

                              yPos = yPos + f1.getHeight() + 10;
          g.setFont(f3);
          g.drawString("SMALL FONT", 0, yPos, Graphics.LEFT | Graphics.TOP);
                              yPos = yPos + f1.getHeight() + 10;
         g.drawLine(0, f3.getHeight() + yPos - 1, width, f3.getHeight() + yPos - 1);

          painting = false;
       }

           }
 }
```

WRITE A J2ME APPLICATION TO DISPLAY HELLO WORLD.

Source Code:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class HelloMidlet extends MIDlet {
public HelloMidlet() {
}public void startApp() {
Form form = new Form( "First Program" );
form.append( "Hello World" );
Display.getDisplay(this).setCurrent( form );
}public void pauseApp() {
}public void destroyApp( boolean unconditional ) {
}
}
```

**Screen shot:**



```
First Program
Hello World
```

## Week - 2 Working with J2ME Features:

Working with J2ME Features: Say, creating a *Hello World* program Experiment with the most basic features and mobile application interaction concepts (lists, text boxes, buttons, radio boxes, soft buttons, graphics, etc)

**2.1 Create a program which creates to following kind of menu.**

* cut

* copy

* past

* delete

* select all

* unselect all

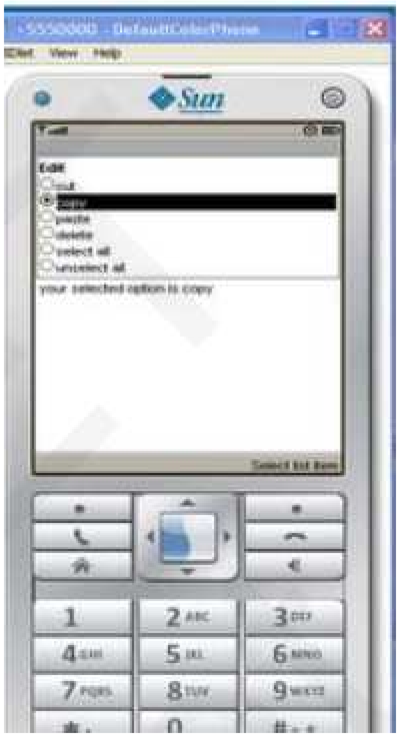
### Create a program which creates a select menu

**Aim:** Develop a MIDlet Application for select list item

**Source code:**

```
 import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class MenuCreation extends MIDlet   implements CommandListener {
   public ChoiceGroup ch;
   public Form form;
   public Display display;
   public Command command;
    public StringItem st;
   public MenuCreation()
   {
      display=Display.getDisplay(this);
      ch=new ChoiceGroup("Edit",Choice.EXCLUSIVE);
      ch.append("cut",null);
      ch.append("copy",null);
      ch.append("paste",null);
      ch.append("delete",null);
      ch.append("select all",null);
      ch.append("unselect all",null);
      ch.setSelectedIndex(1, true);
      command=new Command("Select list item",Command.OK,1);
      form=new Form("");
      form.append(ch);
      form.addCommand(command);
      form.setCommandListener(this);
      st=new StringItem("","");
   }
public void startApp() {
   display.setCurrent(form);
}

public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
}

public void commandAction(Command command,Displayable displayable)
{
         if(command==command)

              {
                 st.setText("");
                 st.setText("your selected option is "+ch.getString(ch.getSelectedIndex()));
                 form.append(st);
              }
      }
   }
```

**OUTPUT:**



```
Edit
○ cut
◉ copy
○ paste
○ delete
○ select all
○ unselect all
your selected option is copy
                    Select list item
```

**WEEK -3**

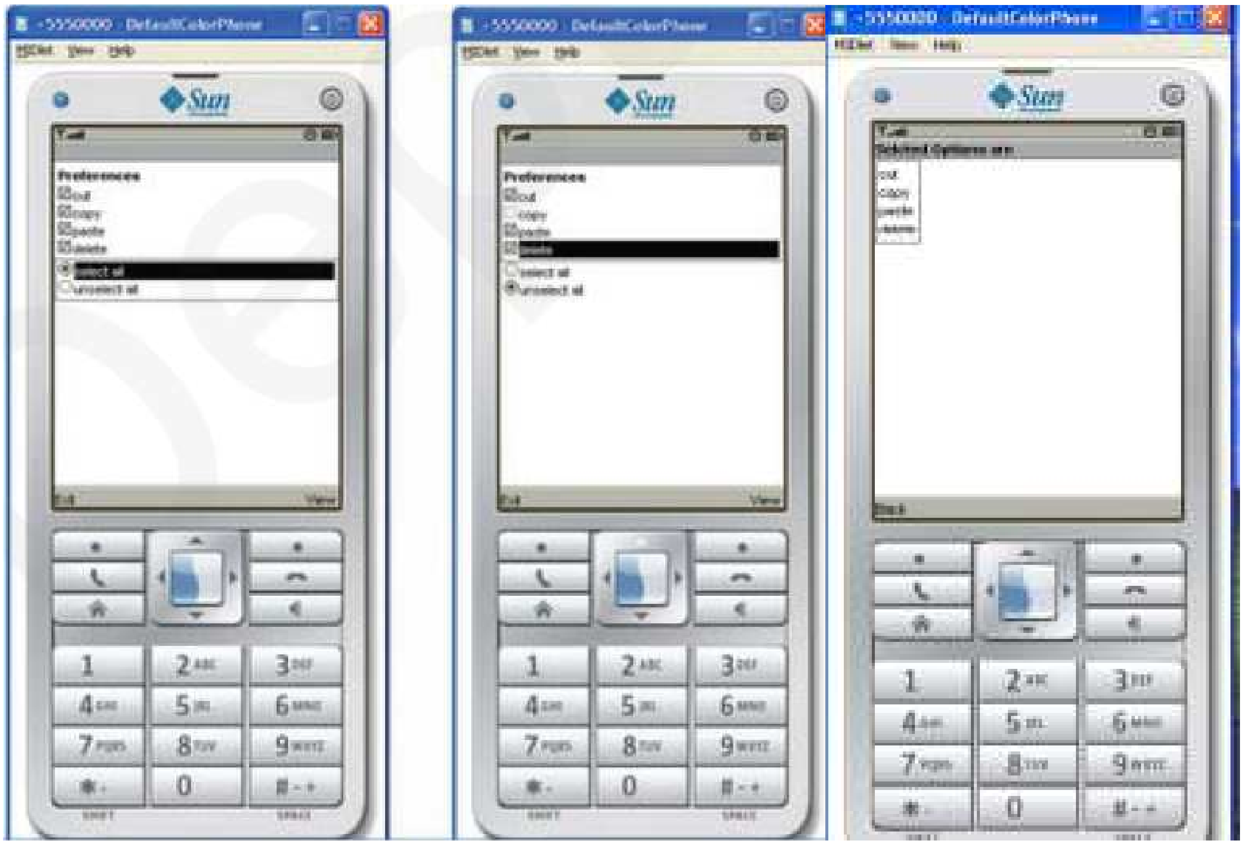
WRITE A J2ME APPLICATION TO IMPLEMENT EVENT HANDLING.

Source code:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class MenuEvents extends MIDlet implements CommandListener,ItemStateListener {
        public ChoiceGroup ch;
        public ChoiceGroup ch1;
        public Form form;
        public Form form1;
        public Display display;
        public Command View;
        public Command Exit;
        public Command Back;
        public StringItem options;
        public Item item;
        public MenuEvents()
        {
                display=Display.getDisplay(this);
                form=new Form("");
                form1=new Form("Selcted Options are");
                ch=new ChoiceGroup("Preferences",Choice.MULTIPLE);
                ch.append("cut",null);
                ch.append("copy",null);
                ch.append("paste",null);
                ch.append("delete",null);
                ch.setSelectedIndex(1, true);
                form.append(ch);
                ch1=new ChoiceGroup("",Choice.EXCLUSIVE);
                ch1.append("select all",null);
                ch1.append("unselect all",null);
                ch1.setSelectedIndex(1, true);
                form.append(ch1);
                View=new Command("View",Command.OK,1);
                Exit =new Command("Exit",Command.EXIT,1);
                Back=new Command("Back",Command.BACK,1);
                form.addCommand(View);
                form.addCommand(Exit);
                form1.addCommand(Back);
                form.setCommandListener(this);
                form1.setCommandListener(this);
                form.setItemStateListener(this);
                }
                        public void startApp() {
                        display.setCurrent(form);
                        } public void pauseApp() {
                        }public void destroyApp(boolean unconditional) {
                        }
                public void commandAction(Command command,Displayable displayable)
                {
                if(displayable==form)
                {
                if(command==View)
                {
                boolean opt[]=new boolean[ch.size()];
                options=new StringItem("","");
                String values="";
                ch.getSelectedFlags(opt);
                options.setText("");
                for(int i=0;i<opt.length;i++)
                {
                if(opt[i])
                {
                values+=ch.getString(i)+"\n";
                }
                }
                options.setText(values);
                form1.append(options);
                display.setCurrent(form1);
                }
                else if(command==Exit)
                {
                destroyApp(true);
                notifyDestroyed();
                }
                }
                else if(displayable==form1)
                {
                if(command==Back)
                {
                display.setCurrent(form);
                options.setText("");
                }}}
                public void itemStateChanged(Item item)
                {
                if(item==ch1)
                {
                int i=0;
                int size=ch.size();
                while(i<size)
                {
                if(ch1.getSelectedIndex()==0)
```

```
                    ch.setSelectedIndex(i, true);
                    else
                    ch.setSelectedIndex(i, false);
                    i++;} } }}
```

**OUTPUT:**



**WEEK-4**

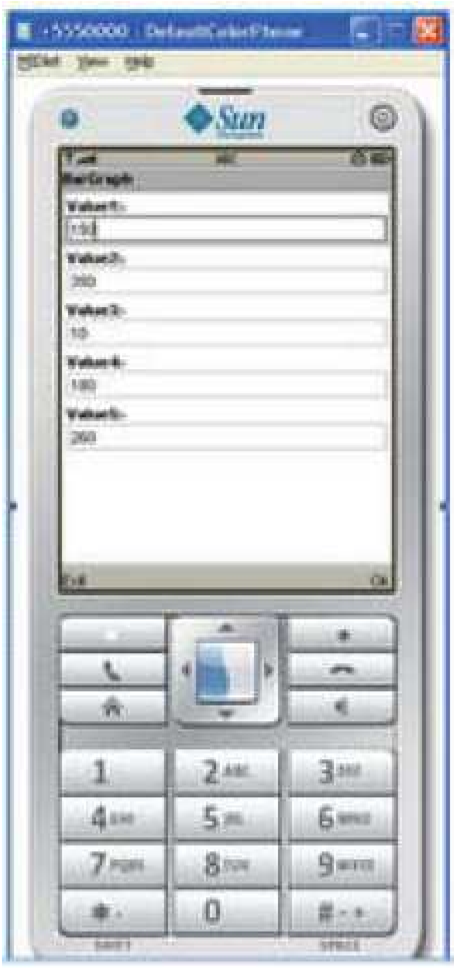
WRITE A J2ME APPLICATION TO DISPLAY BAR GRAPH.

Source code:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class BarGraphMIDlet extends MIDlet implements CommandListener{
public Form form;
public Command exitCommand;
public Command OkCommand;
public Command backCommand;
public Displayable d;
public Display display;
public TextField textfield1;
public TextField textfield2;
public TextField textfield3;
public TextField textfield4;
public TextField textfield5;
public BarGraphMIDlet()
{
display=Display.getDisplay(this);
form=new Form("BarGraph");
textfield1=new TextField("Value1:-","",30,TextField.ANY);
textfield2=new TextField("Value2:-","",30,TextField.ANY);
textfield3=new TextField("Value3:-","",30,TextField.ANY);
textfield4=new TextField("Value4:-","",30,TextField.ANY);
textfield5=new TextField("Value5:-","",30,TextField.ANY);
form.append(textfield1);
form.append(textfield2);
form.append(textfield3);
form.append(textfield4);
form.append(textfield5);
OkCommand=new Command("Ok",Command.OK,1);
exitCommand=new Command("Exit",Command.EXIT,1);
backCommand=new Command("Back",Command.BACK,1);
form.addCommand(OkCommand);
form.addCommand(exitCommand);
form.setCommandListener(this);
}
public void startApp() {
display.setCurrent(form);
}
public void pauseApp() {
}
public void destroyApp(boolean unconditional) {
}
```

```
public void commandAction(Command command,Displayable displayable)
{
if(displayable==form)
{
if(command==OkCommand)
{
int[] data=new int[5];
data[0]=Integer.parseInt(textfield1.getString());
data[1]=Integer.parseInt(textfield2.getString());
data[2]=Integer.parseInt(textfield3.getString());
data[3]=Integer.parseInt(textfield4.getString());

data[4]=Integer.parseInt(textfield5.getString());
d=new BarCanvas(data);
d.addCommand(backCommand);
d.setCommandListener(this);
display.setCurrent(d);
}
else if(command==exitCommand)
notifyDestroyed();
}
else if(displayable==d)
{
if(command==backCommand)
display.setCurrent(form);
}}}
class BarCanvas extends Canvas{
int[] data;
public int x;
public int y;
public int y1;
public int h;
public BarCanvas(int[] data)
{
this.data=data;
x=10;
}
public void paint(Graphics g)
{
g.setColor(255, 255, 255);
g.fillRect(0, 0, this.getWidth(), this.getHeight());
g.setColor(255, 125, 100);
int i=0;
y1=data[0];
h=200;
while(i<data.length)
{
y=data[i];
h=200+y1-y;
g.fillRect(x, y,25 , h);
x+=30;
i++;}} }
```

Screen Shot:



BarGraph

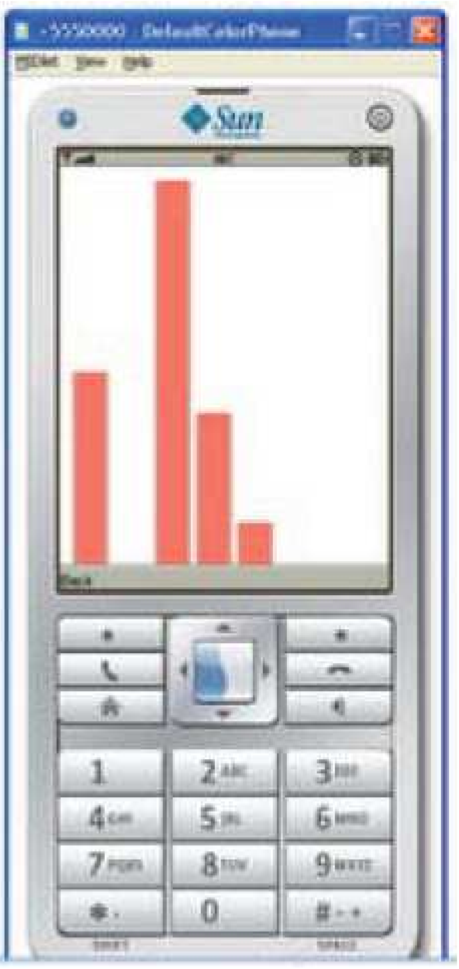
Value1:
150

Value2:
350

Value3:
10

Value4:
180

Value5:
260

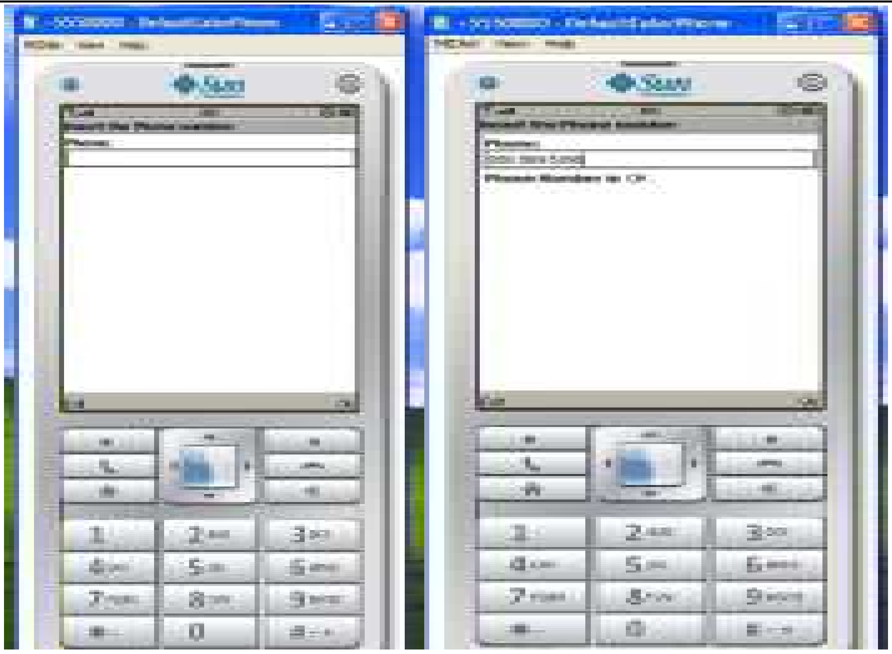


**WEEK-5**

WRITE A J2ME APPLICATION TO IMPLEMENTTHE INPUT CHECKING.
Source code:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
/**
* @author ADMIN
*/
public class InputChecking extends MIDlet implements CommandListener {
public Form form1;
public TextField textfield1;
public Command exitCommand;
public Command okCommand;
public StringItem st;
public Display display;
public InputChecking()
{
display=Display.getDisplay(this);
form1=new Form("Insert the Phone number");
exitCommand=new Command("Exit",Command.EXIT,1);
okCommand=new Command("Ok",Command.OK,1);
st=new StringItem("Phone Number is ","");
textfield1=new TextField("Phone;","",30,TextField.ANY);
form1.append(textfield1);
form1.addCommand(okCommand);
form1.addCommand(exitCommand);
form1.setCommandListener(this);
}
public void startApp() {
display.setCurrent(form1);
}
public void pauseApp() {
}
public void destroyApp(boolean unconditional) {
}
public void commandAction(Command cmd,Displayable displayable)
{
if(cmd==exitCommand)
notifyDestroyed();
else if(cmd==okCommand)
{ String s=textfield1.getString();
s=s.replace(' ', '.');
int len=s.length();
int i=0;
int c=0;
String s1="";
while(i<len)
{
if(s.charAt(i)=='.')
{
if(c==0)
{
if(s1.equals("040") || s1.equals("041") || s1.equals("050") || s1.equals("0400") ||
s1.equals("044"))
{
c++;
s1="";
}
}
if(c==1)
{

if(s1.length()-1==3)
{ c++;
s1=""; } }}
s1=s1+s.charAt(i);
i++;
}
if(s1.length()-1==3 || s1.length()-1==4 || s1.length()-1==5)
c++;
if(c==3)
st.setText("OK");
else
{ st.setText("wrong\n Phone Number Format is xxx xxx xxxx\nArea code must be
040|050|041|0400|044");
}
form1.append(st);
 }
}
}
```

**OUTPUT:**



**WEEK-6**

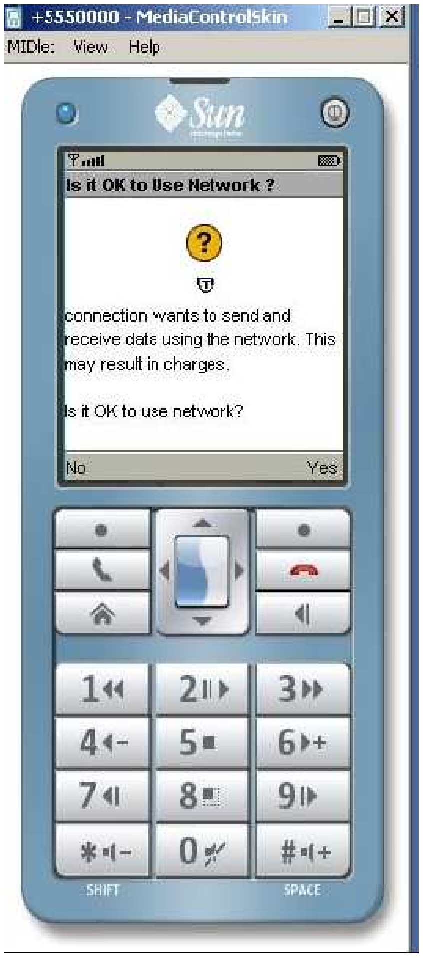
Write a Program to Make a SOCKET connection from a J2ME Program.

This J2ME sample program shows how to how to make a SOCKET Connection from a J2ME Phone.Many a times there is a need to connect theto a backend HTTP server from the J2ME application. This free J2ME sample program for example shows how to make a SOCKET connection from the phone to port 80 of

**SOURCE CODE:**

```
import javax.microedition.midlet.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import java.io.*;|
public class socket extends MIDlet {
          // StreamConnection allows bidirectional communication
  private StreamConnection streamConnection = null;
          // use OutputStream to send requests
  private OutputStream outputStream = null;
  private DataOutputStream dataOutputStream = null;
           // use InputStream to receive responses from Web server
  private InputStream inputStream = null;
  private DataInputStream dataInputStream = null;
           // specify the connect string
  private String connectString = "socket://www.java-samples.com:80";
           // use a StrignBuffer to store the retrieved page contents
  private StringBuffer results;
           // define GUI components
  private Display myDisplay = null;
  private Form resultScreen;
  private StringItem resultField;
   public socket() {
         // initializing GUI display
    results = new StringBuffer();
    myDisplay = Display.getDisplay(this);
    resultScreen = new Form("Page Content:");
  }

  public void startApp() {
    try {
              // establish a socket connection with remote server
      streamConnection =
      (StreamConnection) Connector.open(connectString);
              // create DataOuputStream on top of the socket connection
      outputStream = streamConnection.openOutputStream();
      dataOutputStream = new DataOutputStream(outputStream);
              // send the HTTP request
      dataOutputStream.writeChars("GET /index.htm HTTP/1.0 \n");
      dataOutputStream.flush();
              // create DataInputStream on top of the socket connection
      inputStream = streamConnection.openInputStream();
      dataInputStream = new DataInputStream(inputStream);
              // retrieve the contents of the requested page from Web server
      int inputChar;
      while ( (inputChar = dataInputStream.read()) != -1) {
        results.append((char) inputChar);
      }
              // display the page contents on the phone screen
      resultField = new StringItem(null, results.toString());
      resultScreen.append(resultField);
      myDisplay.setCurrent(resultScreen);
        } catch (IOException e) {
      System.err.println("Exception caught:" + e);
    } finally {
          // free up I/O streams and close the socket connection
      try {
        if (dataInputStream != null)
          dataInputStream.close();
      } catch (Exception ignored) {}
      try {
        if (dataOutputStream != null)
          dataOutputStream.close();
      } catch (Exception ignored) {}
      try {
        if (outputStream != null)
          outputStream.close();
      } catch (Exception ignored) {}
      try {
        if (inputStream != null)
          inputStream.close();
      } catch (Exception ignored) {}
      try {
        if (streamConnection != null)
          streamConnection.close();
      } catch (Exception ignored) {}
    }
  }
    public void pauseApp() {
  }
   public void destroyApp(boolean unconditional) {
  }
}
```

**OUTPUT:**



Is it OK to Use Network ?

connection wants to send and receive data using the network. This may result in charges.

Is it OK to use network?

**WEEK-7**

Login to HTTP Server from a J2ME Program

This J2ME sample program shows how to display a simple LOGIN SCREEN on the J2ME phone and how to authenticate to a HTTP server.

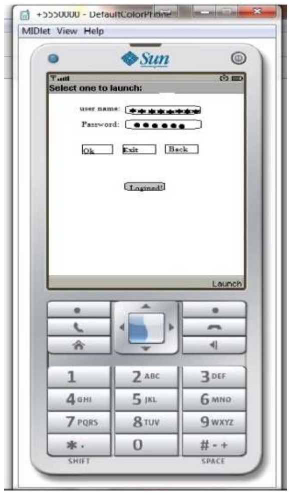
Many J2ME applications for security reasons require the authentication of the user. This free J2ME sample program, shows how a J2ME application can do authentication to the backend server.

**SOURCE CODE:**

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import java.io.*;
import java.lang.*;
import javax.microedition.io.*;
import javax.microedition.rms.*;
import  com.symbol.j2me.midlets.ascanner.BarCodeReader;
public class Login extends MIDlet implements CommandListener {
        TextField UserName=null;
        TextField Location=null;
        Form authForm,mainscreen;
        TextBox t = null;
        StringBuffer b = new StringBuffer();
   private Display myDisplay = null;
   private Command okCommand = new Command("OK", Command.OK, 1);
   private Command exitCommand = new Command("Exit", Command.EXIT, 2);
   private Command backCommand = new Command("Back", Command.BACK, 2);
   private Alert alert = null;
   public Login() {
      myDisplay = Display.getDisplay(this);
        UserName=new TextField("Your Name","",10,TextField.ANY);
        Location=new TextField("Location","",10,TextField.ANY);
        authForm=new Form("Identification");
        mainscreen=new Form("Logging IN");
        mainscreen.append("Logging in....");
        mainscreen.addCommand(backCommand);
        authForm.append(UserName);
        authForm.append(Location);
        authForm.addCommand(okCommand);
        authForm.addCommand(exitCommand);
        authForm.setCommandListener(this);
        myDisplay.setCurrent(authForm);
   }
   public void startApp() throws MIDletStateChangeException {
   }
   public void pauseApp() {
   }
    protected void destroyApp(boolean unconditional)
           throws MIDletStateChangeException {
   }
   public void commandAction(Command c, Displayable d) {
            if ((c == okCommand) && (d == authForm)) {
        if (UserName.getString().equals("") || Location.getString().equals("")){
  alert = new Alert("Error", " enter Username and Location", null, AlertType.ERROR);
                alert.setTimeout(Alert.FOREVER);
                myDisplay.setCurrent(alert);
        }
        else{
        //myDisplay.setCurrent(mainscreen);
        login(UserName.getString(),Location.getString());
        }
        }
      if ((c == backCommand) && (d == mainscreen)) {
                myDisplay.setCurrent(authForm);
        }
        if ((c == exitCommand) && (d == authForm)) {
                notifyDestroyed();
        }
        }
            public void login(String UserName,String PassWord) {
        HttpConnection connection=null;
        DataInputStream in=null;
        String url="http://www.java-samples.com/use-your-own/urlhere.jsp";

        OutputStream out=null;
        try
        {
                connection=(HttpConnection)Connector.open(url);
                connection.setRequestMethod(HttpConnection.POST);
connection.setRequestProperty("IF-Modified-Since", "2 Oct 2002 15:10:15 GMT");
```

```
            connection.setRequestProperty("User-Agent","Profile/MIDP-1.0
Configuration/CLDC-1.0");
            connection.setRequestProperty("Content-Language", "en-CA");
            connection.setRequestProperty("Content-Length",""+
(UserName.length()+PassWord.length()));
            connection.setRequestProperty("UserName",UserName);
            connection.setRequestProperty("PassWord",PassWord);
      out = connection.openDataOutputStream();
      out.flush();
      in = connection.openDataInputStream();
      int ch;
      while((ch = in.read()) != -1) {
            b.append((char) ch);
            //System.out.println((char)ch);
      }
      //t = new TextBox("Reply",b.toString(),1024,0);
      mainscreen.append(b.toString());
      if(in!=null)
            in.close();
      if(out!=null)
            out.close();
      if(connection!=null)
            connection.close();
      }
      catch(IOException x){}
      myDisplay.setCurrent(mainscreen);

  }
}
```

**OUTPUT:**



**WEEK-8**

Program to be carried out with respect to the given set of Application domain

(i) Students Mark Enquiry
(ii) Town/City Movie Enquiry
(iii) Railway/Road/Air(For PNR)Enquiry/Status
(iv) Sports Update
(v) Town/City Weather Update
(vi) Public Exam Results Enquiry.

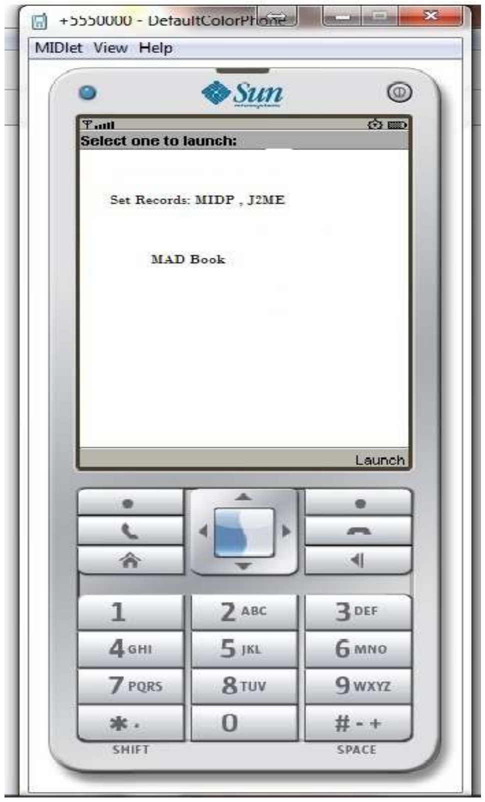
Design a Database according to Domain and render information according to the request.

**SOURCE CODE:**

```
import javax.microedition.rms.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
public class ReadWrite extends MIDlet
{
  private RecordStore rs = null;
  static final String REC_STORE = "db_1";
  public ReadWrite()
  {
    openRecStore();   // Create the record store
    writeRecord("J2ME and MIDP");
    writeRecord("Wireless Technology");
    readRecords();

    closeRecStore();  // Close record store
    deleteRecStore(); // Remove the record store
  }
  public void destroyApp( boolean unconditional )
  {
  }
```

```
public void startApp()
{
 destroyApp(false);
 notifyDestroyed();
}
public void pauseApp()
{
}
public void openRecStore()
{
 try
 {
   rs = RecordStore.openRecordStore(REC_STORE, true );
 }
 catch (Exception e)
 {
  db(e.toString());
 }
}
 public void closeRecStore()
{
 try
 {
  rs.closeRecordStore();
 }
 catch (Exception e)
 {
  db(e.toString());
 }
}
public void deleteRecStore()
{
 if (RecordStore.listRecordStores() != null)
 {
  try
  {
   RecordStore.deleteRecordStore(REC_STORE);
  }
  catch (Exception e)
  {
   db(e.toString());
  }
 }
}
public void writeRecord(String str)
{
    byte[] rec = str.getBytes();

    try
    {
      rs.addRecord(rec, 0, rec.length);
    }
    catch (Exception e)
    {
      db(e.toString());
    }
  }
  public void readRecords()
  {
    try
    {
      // Intentionally make this too small to test code below
      byte[] recData = new byte[5];
      int len;

      for (int i = 1; i <= rs.getNumRecords(); i++)
      {
        if (rs.getRecordSize(i) > recData.length)
          recData = new byte[rs.getRecordSize(i)];

        len = rs.getRecord(i, recData, 0);
        System.out.println("Record #" + i + ": " + new String(recData, 0, len));
        System.out.println("------------------------------");
      }
    }
    catch (Exception e)
    {
      db(e.toString());
    }
  }  private void db(String str)
  {
    System.err.println("Msg: " + str);
  }
}
```

**OUTPUT:**



**WEEK-9**

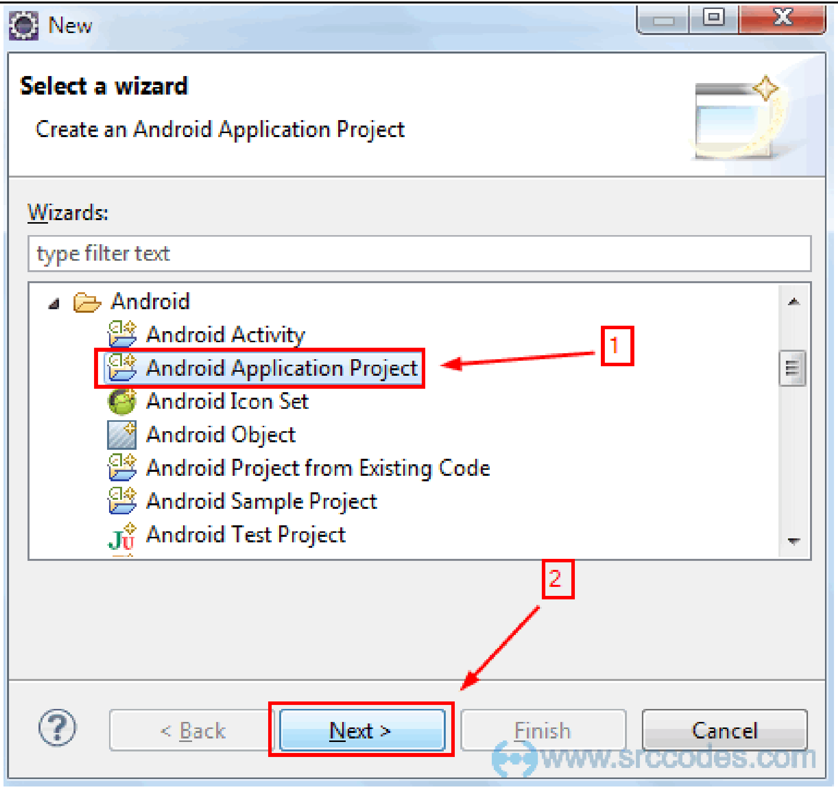
Android Application Program that displays Hello world Using Eclipse

**SOURCE CODE:**

Eclipse IDE with Android Developer Tools (ADT) plugin to build the application and Android Emulator - Android Virtual Device (AVD) to run the application which will draw 'Hello World!' text on the screen .Tools & Technologies used:JDK 1.6, Eclipse 3.7,Android SDK.

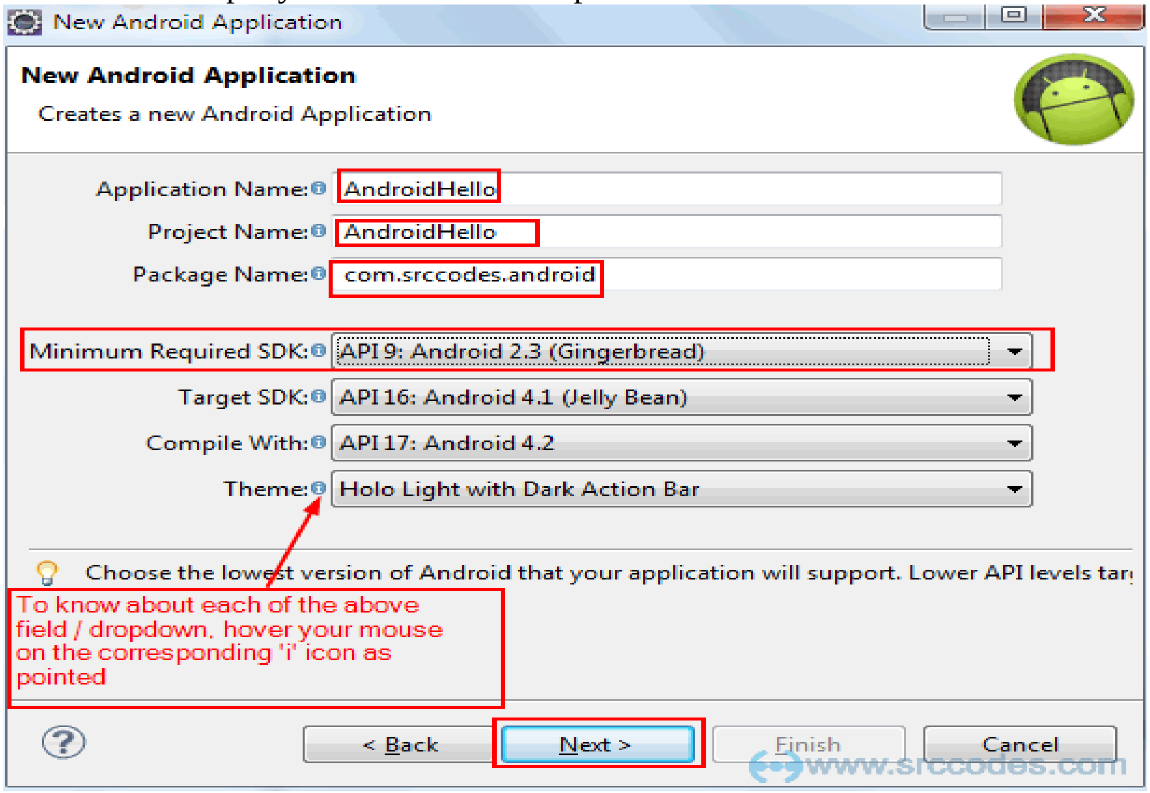
**Step 1.** Create Android Project

Select from the menu File --> New --> Other --> Android --> Android Application Project (say 'Android Hello') and click Next button.



**Step 2.** Configure Project Settings

Enter Application, Project and Package Name. Select 'Minimum Required SDK' (lowest version of Android that this app supports), 'Target SDK' (highest version of Android with which this application has been tested), 'Compile With' (platform version against which this application will be compiled with) and 'Theme' (Android UI style) from the corresponding theme. To make it simple you can leave the dropdown value as it is. Click Next button


New Android Application

Creates a new Android Application

Application Name: AndroidHello

Project Name: AndroidHello

Package Name: com.srccodes.android

Minimum Required SDK: API 9: Android 2.3 (Gingerbread)

Target SDK: API 16: Android 4.1 (Jelly Bean)

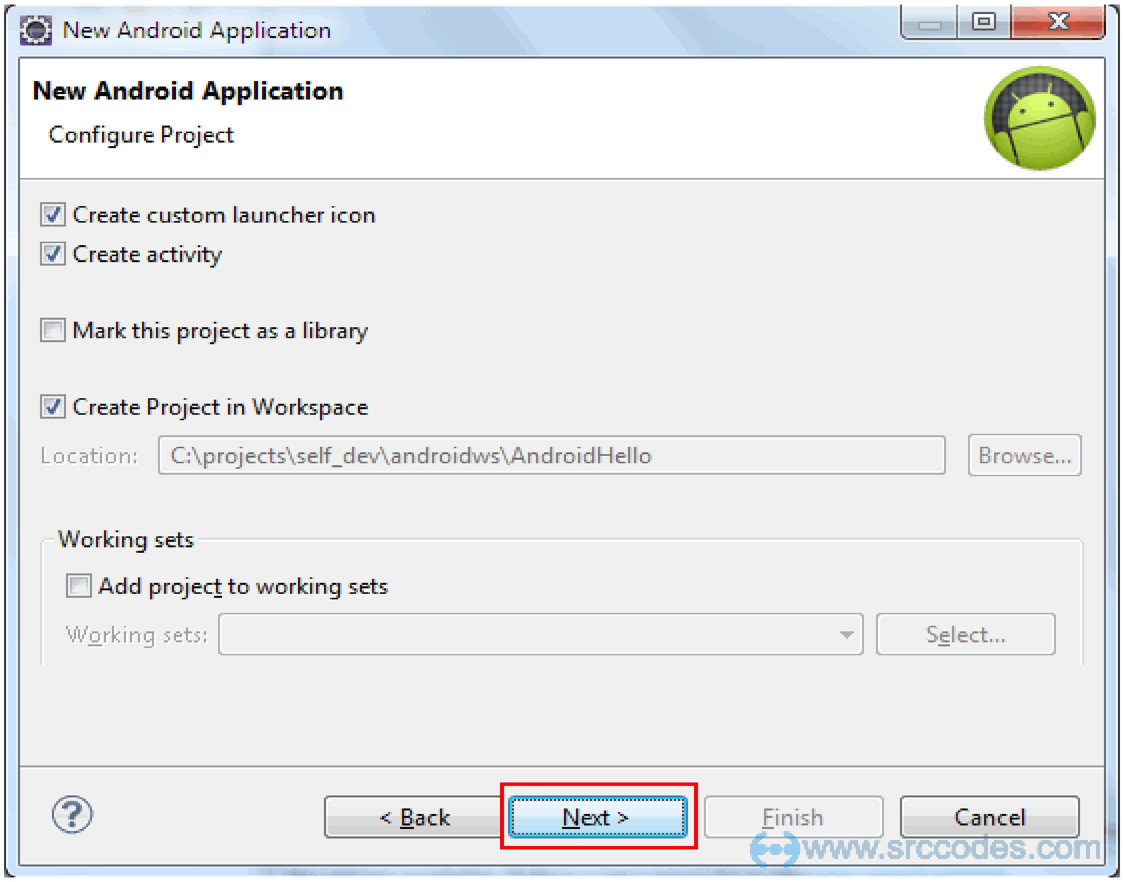
Compile With: API 17: Android 4.2

Theme: Holo Light with Dark Action Bar

Choose the lowest version of Android that your application will support. Lower API levels tar

To know about each of the above field / dropdown, hover your mouse on the corresponding 'i' icon as pointed

Click Next button



New Android Application

**New Android Application**

Configure Project

- [✓] Create custom launcher icon
- [✓] Create activity
- [ ] Mark this project as a library
- [✓] Create Project in Workspace

Location: C:\projects\self_dev\androidws\AndroidHello

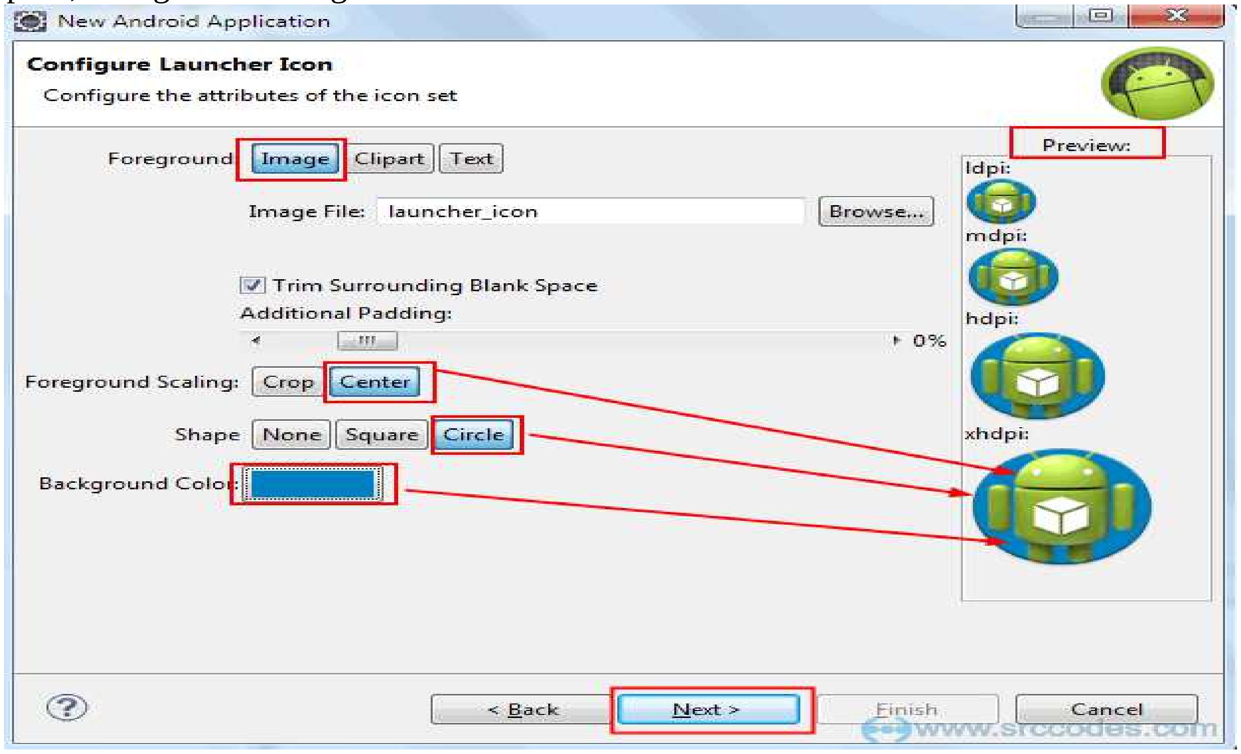
Working sets

- [ ] Add project to working sets

Working sets:

**Step 3.** Configure App Launcher Icon

Choose your App icon and configure as per your requirement. For demonstration purpose, change few settings as shown below



New Android Application

**Configure Launcher Icon**

Configure the attributes of the icon set

Foreground: [✓ Image] Clipart Text

Image File: launcher_icon

[✓] Trim Surrounding Blank Space

Additional Padding: 0%

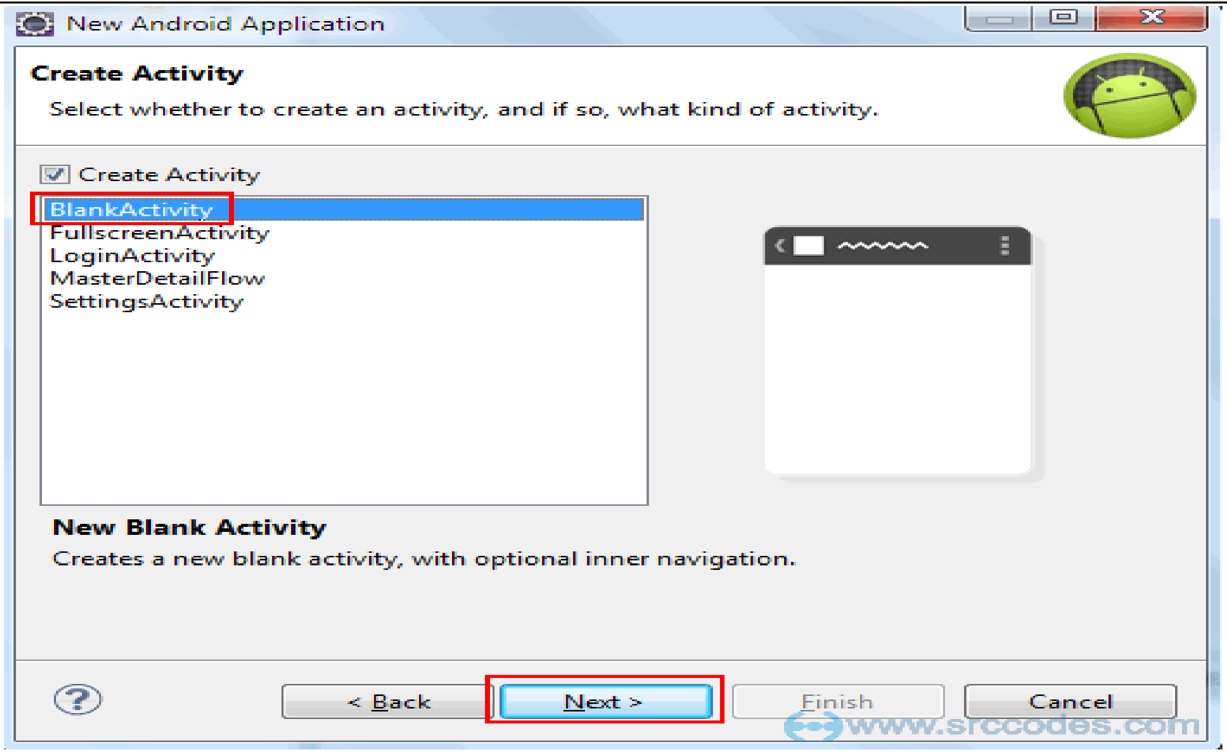
Foreground Scaling: Crop [✓ Center]

Shape: None Square [✓ Circle]

Background Color: 🟦

**Step 4.** Create Activity

Choose an activity template (say 'BlankActivity') and click Next button


New Android Application

**Create Activity**

Select whether to create an activity, and if so, what kind of activity.

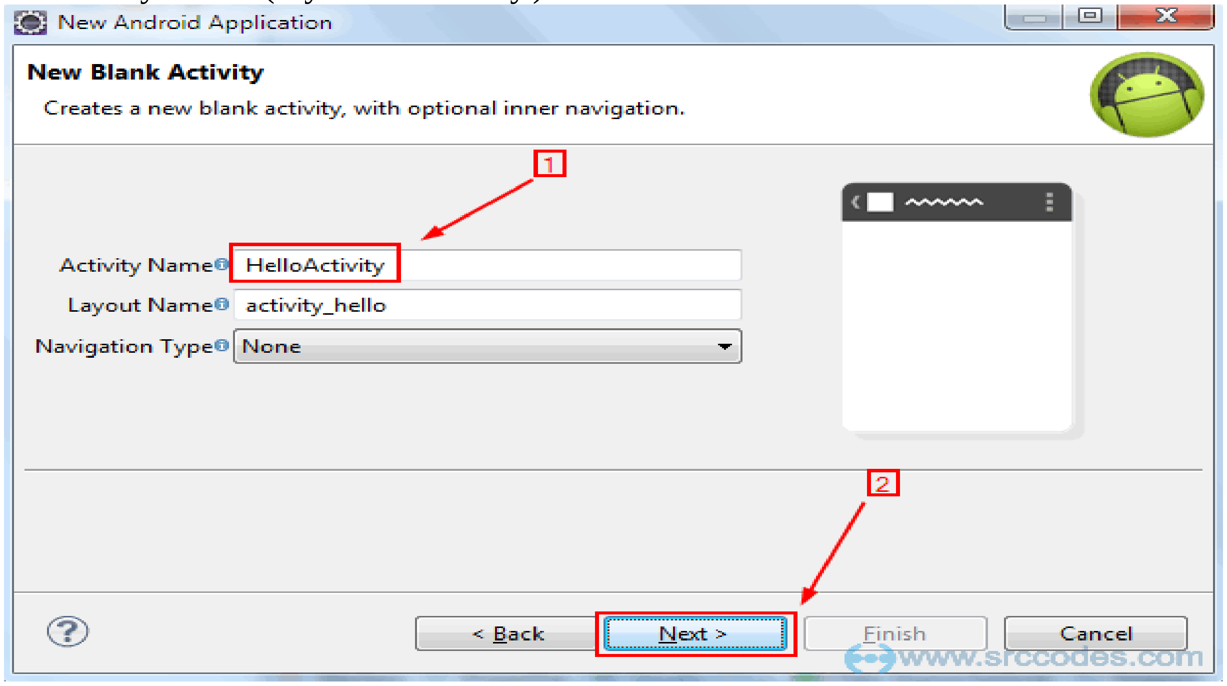
☑ Create Activity

- BlankActivity
- FullscreenActivity
- LoginActivity
- MasterDetailFlow
- SettingsActivity

**New Blank Activity**

Creates a new blank activity, with optional inner navigation.

Enter 'Activity Name' (say 'HelloActivity') and click Finish button

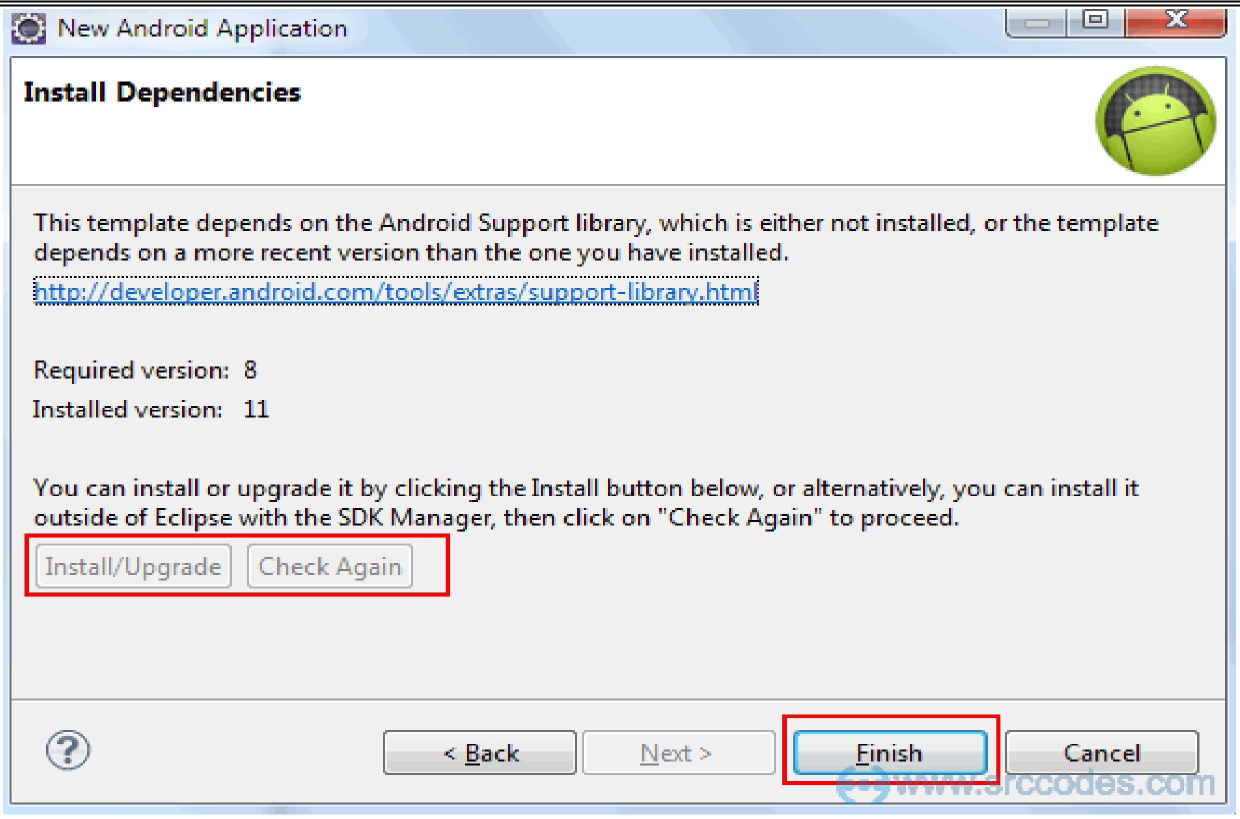

New Android Application

**New Blank Activity**

Creates a new blank activity, with optional inner navigation.

| | |
|---|---|
| Activity Name | HelloActivity |
| Layout Name | activity_hello |
| Navigation Type | None |

If Finish button is not enabled and Next is enabled that means required dependencies (Supporting library) are not installed. In this case click Next button and hit Install/Upgrade' button to install or upgrade required dependencies.


New Android Application

**Install Dependencies**

This template depends on the Android Support library, which is either not installed, or the template depends on a more recent version than the one you have installed.
http://developer.android.com/tools/extras/support-library.html

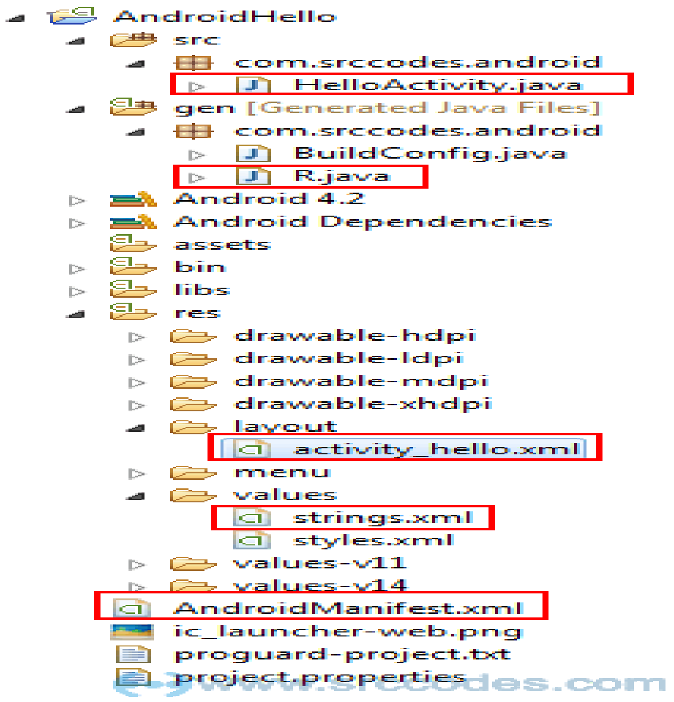
Required version: 8
Installed version: 11

You can install or upgrade it by clicking the Install button below, or alternatively, you can install it outside of Eclipse with the SDK Manager, then click on "Check Again" to proceed.

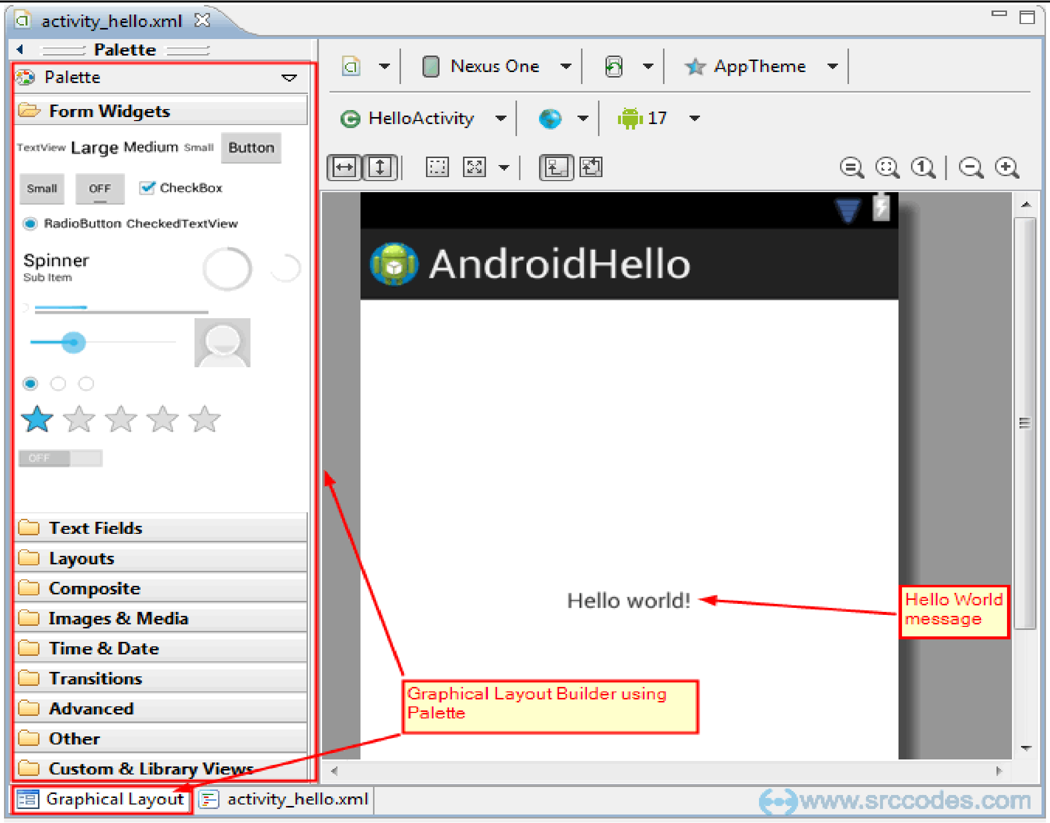
Finally click Finish button.

**Step 5.** Overall Project Structure

Android project will be created with some default files as shown below


```
AndroidHello
  src
    com.srccodes.android
      HelloActivity.java
  gen [Generated Java Files]
    com.srccodes.android
      BuildConfig.java
      R.java
  Android 4.2
  Android Dependencies
  assets
  bin
  libs
  res
    drawable-hdpi
    drawable-ldpi
    drawable-mdpi
    drawable-xhdpi
    layout
      activity_hello.xml
    menu
    values
      strings.xml
      styles.xml
    values-v11
    values-v14
  AndroidManifest.xml
  ic_launcher-web.png
  proguard-project.txt
  project.properties
```

'android_hello.xml' (layout) will be opened using 'Android Common XML Editor'. Here we can build UI by simply dragging and dropping UI components from the Palette.



**Step 6.** Code

'hello_world' resource string contains the message 'Hello world!' which will be shown on launching of the application.

File : strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <string name="app_name">AndroidHello</string>
 <string name="hello_world">Hello world!</string>
  <string name="menu_settings">Settings</string>
   </resources>
```

'activity_hello.xml' is the layout built using 'Android Common XML Editor'. Instead of using a hard-coded string value ('Hello world!') in '<TextView>' element, the "@string/hello_world" value refers to a string resource defined in strings.xml.

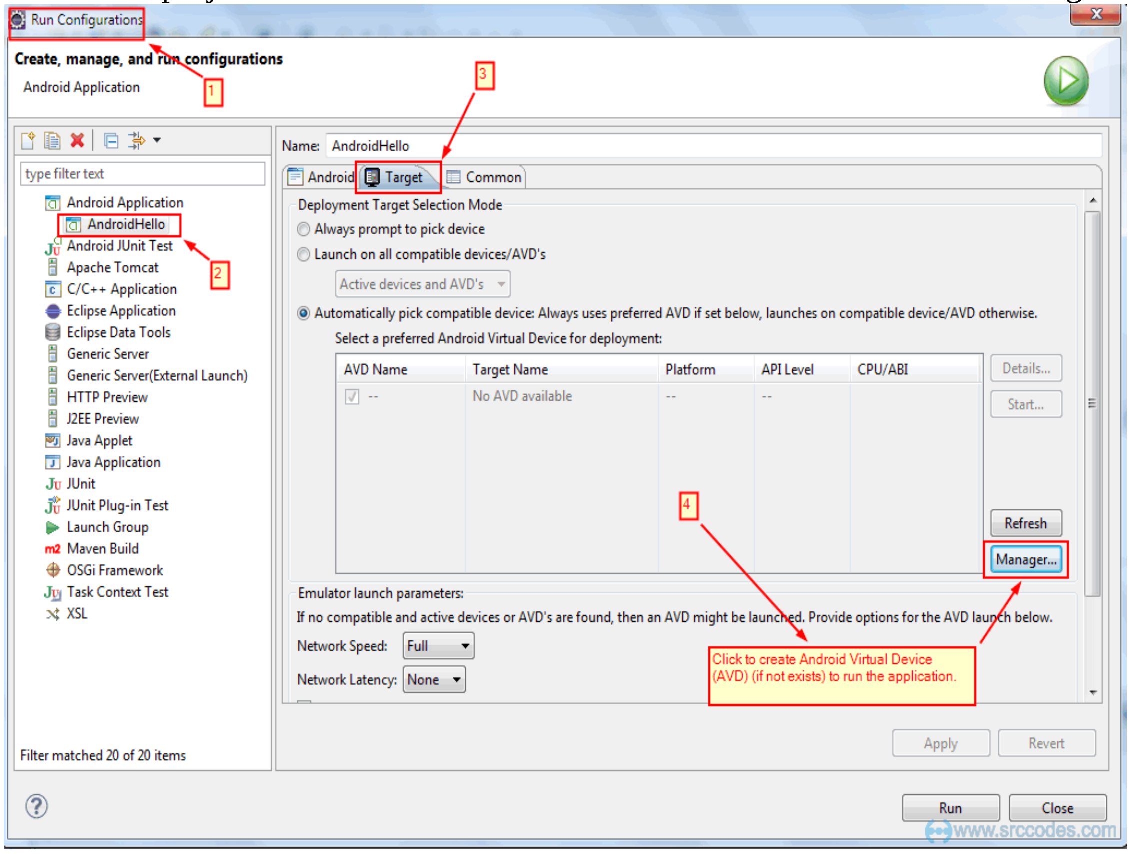
File : activity_hello.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HelloActivity" >
     <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/hello_world" />
 </RelativeLayout>
```

For this application we do not require to change anything in the generated activity code.
File : HelloActivity.java

```
package com.srccodes.android;
 import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
 public class HelloActivity extends Activity {
     @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hello);
    }
     @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.activity_hello, menu);
        return true;
    }
}
```

**Step 7.** Run Configuration
Right click on the project and from the context menu select 'Run As' --> 'Run Configurations..'



Name: AndroidHello

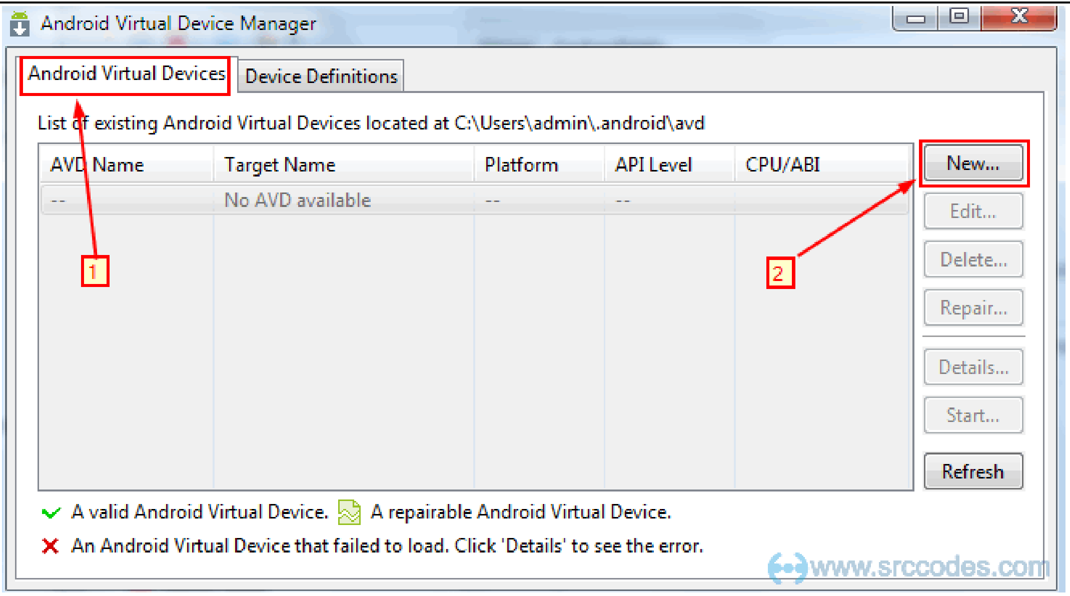
Deployment Target Selection Mode
Always prompt to pick device
Launch on all compatible devices/AVD's
[✓ Automatically pick compatible device: Always uses preferred AVD if set below, launches on compatible device/AVD otherwise.]
Select a preferred Android Virtual Device for deployment:

| AVD Name | Target Name | Platform | API Level | CPU/ABI |
|---|---|---|---|---|
| -- | No AVD available | -- | -- | |

Emulator launch parameters:
If no compatible and active devices or AVD's are found, then an AVD might be launched. Provide options for the AVD launch below.
Network Speed: Full
Network Latency: None

Click to create Android Virtual Device (AVD) (if not exists) to run the application.

If there is no Android Virtual Device (AVD) already created, then click Manager button to create one


Android Virtual Device Manager

Android Virtual Devices | Device Definitions

List of existing Android Virtual Devices located at C:\Users\admin\.android\avd

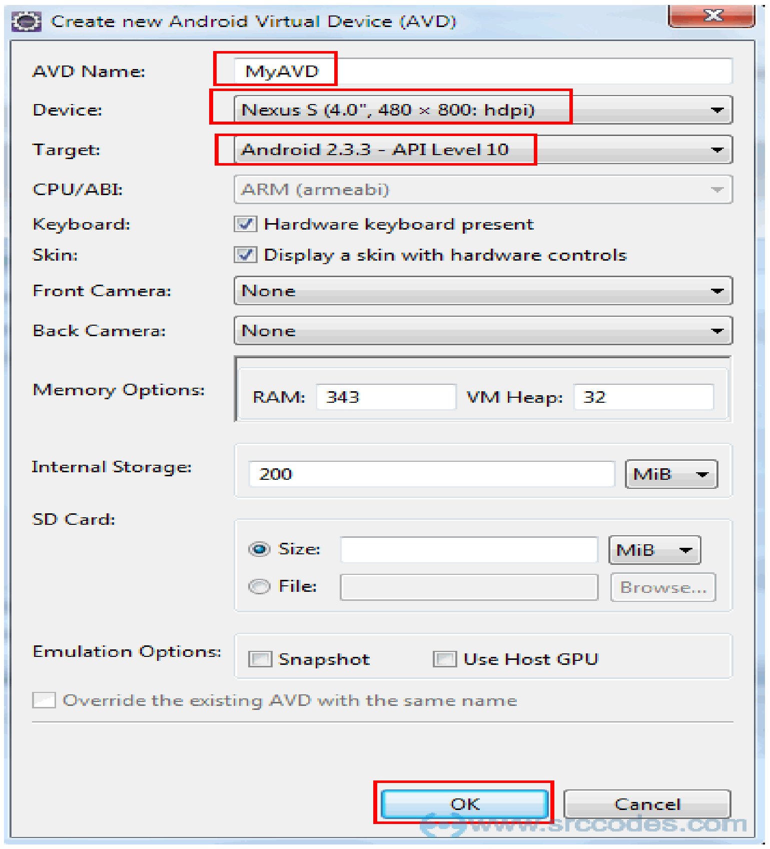
| AVD Name | Target Name | Platform | API Level | CPU/ABI |
|---|---|---|---|---|
| -- | No AVD available | -- | -- | |

✓ A valid Android Virtual Device. A repairable Android Virtual Device.

✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

www.srccodes.com

Configure AVD as shown below and click OK button.


Create new Android Virtual Device (AVD)

| | |
|---|---|
| AVD Name: | MyAVD |
| Device: | Nexus S (4.0", 480 × 800: hdpi) |
| Target: | Android 2.3.3 - API Level 10 |
| CPU/ABI: | ARM (armeabi) |
| Keyboard: | ☑ Hardware keyboard present |
| Skin: | ☑ Display a skin with hardware controls |
| Front Camera: | None |
| Back Camera: | None |
| Memory Options: | RAM: 343 VM Heap: 32 |
| Internal Storage: | 200 MiB |
| SD Card: | ● Size: MiB ○ File: Browse... |
| Emulation Options: | ☐ Snapshot ☐ Use Host GPU |

☐ Override the existing AVD with the same name

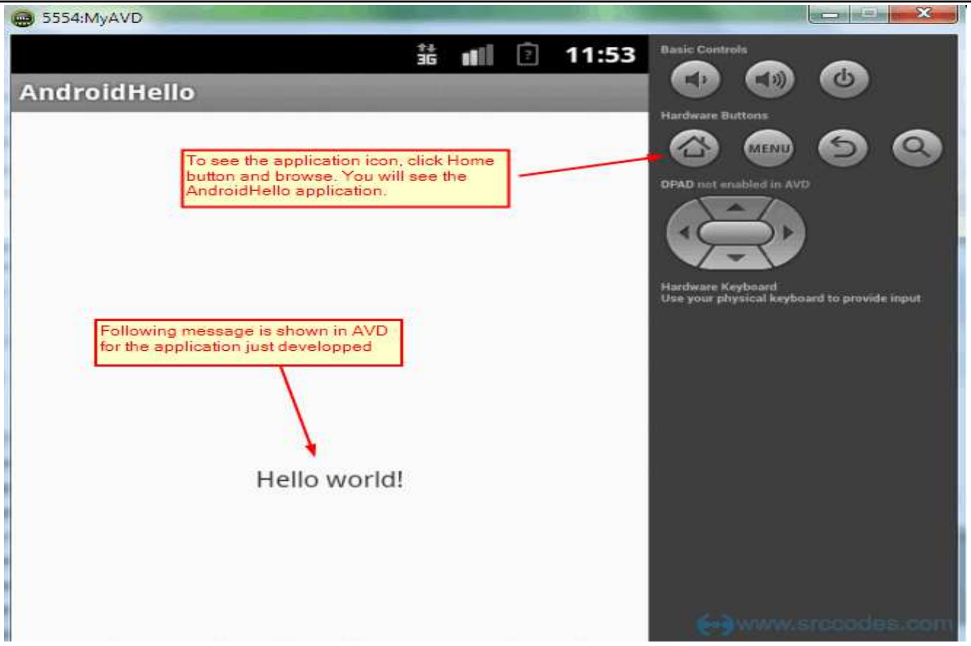
www.srccodes.com

**Step 8.** Run Application

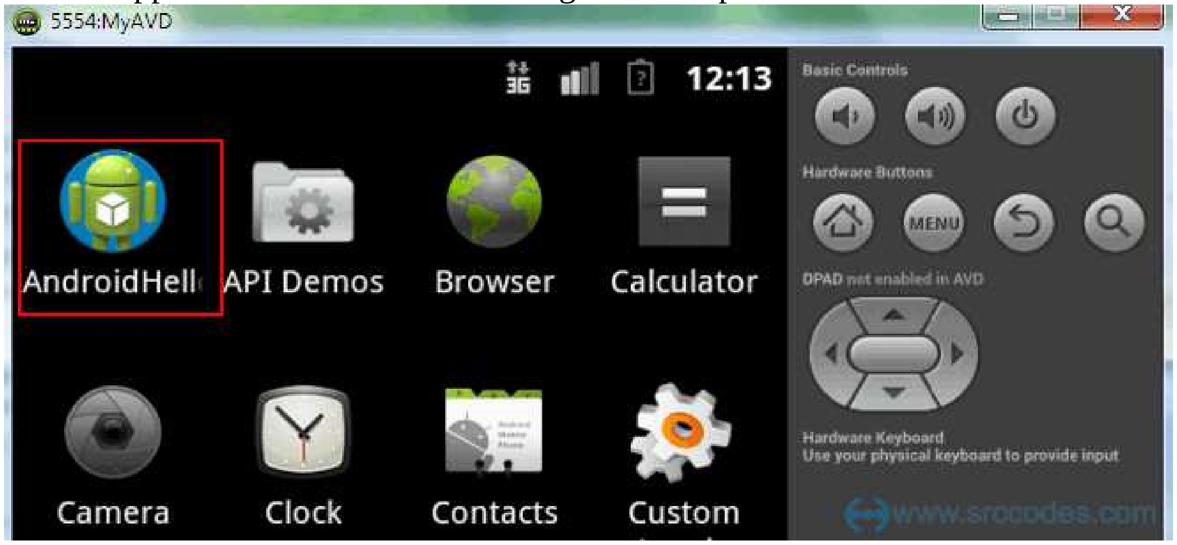
Right click on the project and from the context menu select 'Run As' --> 'Android Application'.

**Step 9.** Output

Eclipse ADT will start the AVD and launch your application with 'Hello world!' message on the screen.


5554:MyAVD
11:53
AndroidHello
To see the application icon, click Home button and browse. You will see the AndroidHello application.
Following message is shown in AVD for the application just developped
Hello world!
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input
www.srccodes.com

Click home icon in the emulator and click the launcher icon to find your application. There you'll see the app icon which have been configured at step #3.



**WEEK-10**

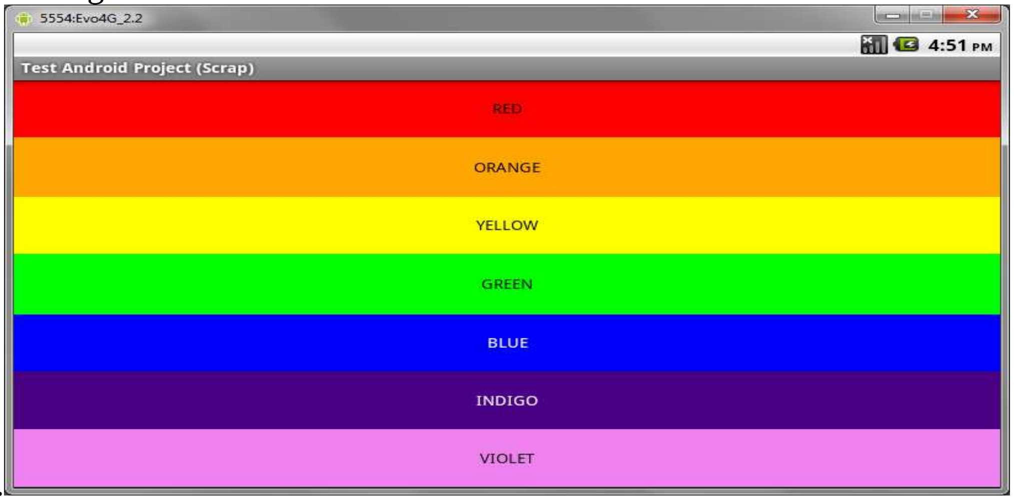
Android Application Program that demonstrates the following

i. Linear Layout
ii. Relative Layout
iii. Table Layout
iv. Grid View Layout

**SOURCE CODE:**

***Linear Layout:***

This linear layout's orientation is set to vertical, causing each of the TextView controls to display in a single column.



.

➢ **Defining an XML Layout Resource with a Linear Layout**

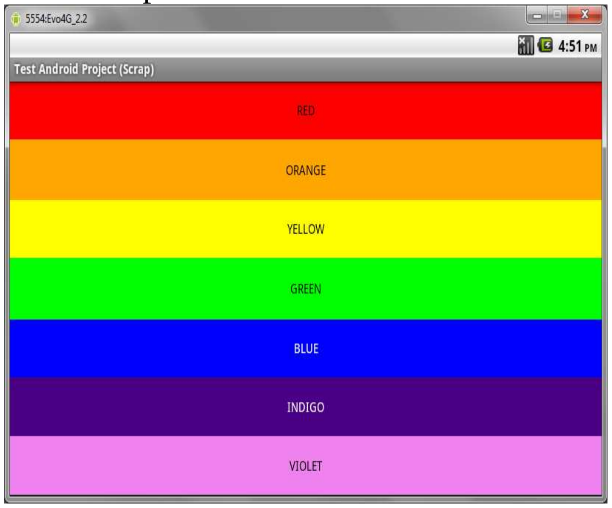
This layout resource file, aptly named /res/layout/rainbow.xml, is defined in XML as follows:

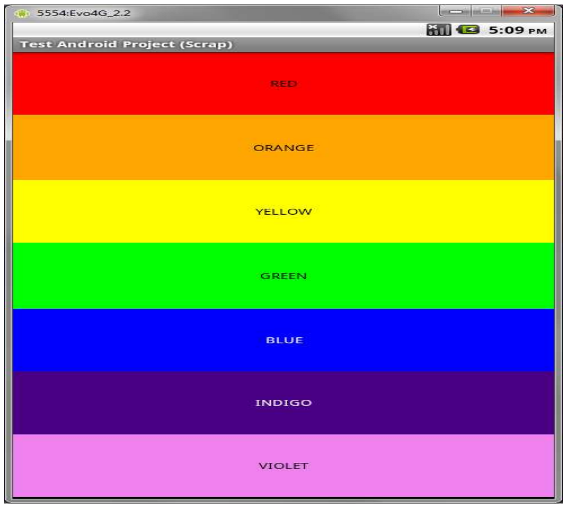
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent" android:layout_height="fill_parent"
```

```
            android:orientation="vertical">
            <TextView android:text="RED" android:id="@+id/TextView01"
                android:layout_height="wrap_content" android:background="#f00"
               android:layout_width="fill_parent" android:layout_weight=".14"
             android:gravity="center" android:textColor="#000"></TextView>
          <TextView android:text="ORANGE" android:id="@+id/TextView02"
                android:layout_height="wrap_content" android:layout_width="fill_parent"
                android:layout_weight=".15" android:background="#ffa500"
                android:gravity="center" android:textColor="#000"></TextView>
            <TextView android:text="YELLOW" android:id="@+id/TextView03"
                android:layout_height="wrap_content" android:layout_width="fill_parent"
                android:layout_weight=".14" android:background="#ffff00"
                android:gravity="center" android:textColor="#000"></TextView>
            <TextView android:text="GREEN" android:id="@+id/TextView04"
                android:layout_height="wrap_content" android:layout_width="fill_parent"
  android:layout_weight=".15" android:background="#0f0" android:gravity="center"
                android:textColor="#000"></TextView>
            <TextView android:text="BLUE" android:id="@+id/TextView05"
                android:layout_height="wrap_content" android:layout_width="fill_parent"
         android:layout_weight=".14" android:background="#00f" android:gravity="center"
         android:textColor="#fff"></TextView>
           <TextView android:text="INDIGO" android:id="@+id/TextView06"
                android:layout_height="wrap_content" android:layout_width="fill_parent"
                android:layout_weight=".14" android:background="#4b0082"
                android:gravity="center" android:textColor="#fff"></TextView>
            <TextView android:text="VIOLET" android:id="@+id/TextView07"
                android:layout_height="wrap_content" android:layout_width="fill_parent"
                android:layout_weight=".14" android:background="#ee82ee"
                android:gravity="center" android:textColor="#000"></TextView>
        </LinearLayout>
```

Each of the child controls of the linear layout have a number of interesting attributes, including one called layout_weight.

The following two figures show what this layout might look like on a device in both portrait and landscape modes:





Only a single line of code within the on Create() method is necessary to load and display a layout resource on the screen. If the layout resource was stored in the /res / layout / rainbow. xml file,that line of code would be: set Content View (R.layout.rainbow);

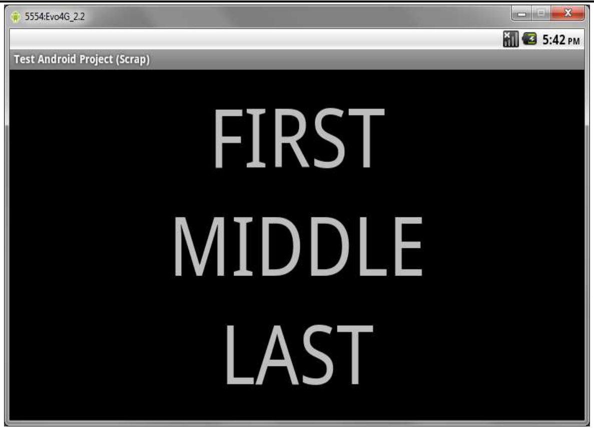
- **Defining a Linear Layout Programmatically**

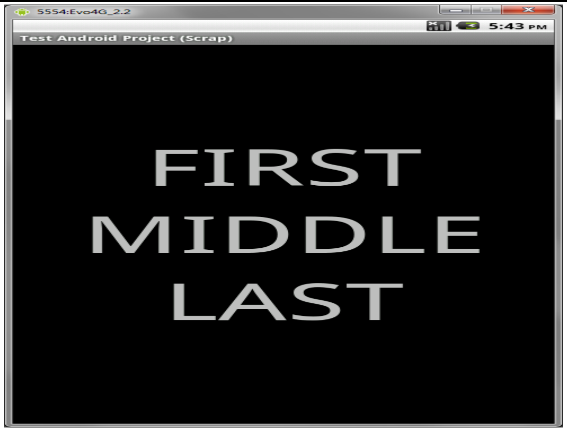
Programmatically create and configure linear layouts. This is done using the LinearLayout class (android.widget.LinearLayout).
child-specific parameters in the LinearLayout.LayoutParams class.

Layout parameters (android.view.ViewGroup.LayoutParams), such as layout_height and layout_width, as well as margin parameters (ViewGroup.MarginLayoutParams), still apply to LinearLayout objects. programmatically have an Activity instantiate a Linear Layout and place three TextView objects within it in its onCreate() method:

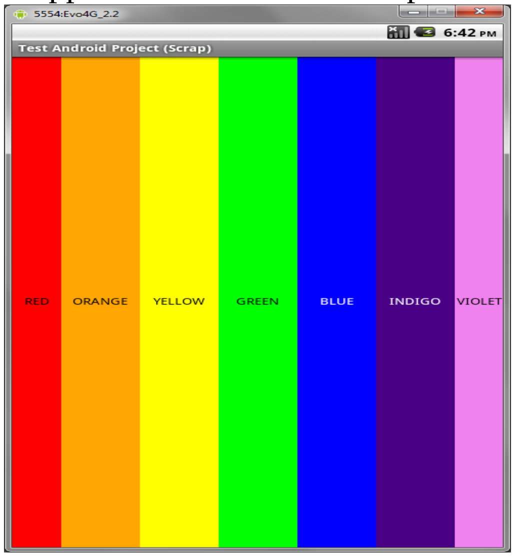
```
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   // setContentView(R.layout.rainbow);
   TextView tv1 = new TextView(this);
   tv1.setText("FIRST");
   tv1.setTextSize(100);
   tv1.setGravity(Gravity.CENTER);
   TextView tv2 = new TextView(this);
   tv2.setTextSize(100);
   tv2.setGravity(Gravity.CENTER);
   tv2.setText("MIDDLE");
   TextView tv3 = new TextView(this);
   tv3.setTextSize(100);
   tv3.setGravity(Gravity.CENTER);
   tv3.setText("LAST");
   LinearLayout ll = new LinearLayout(this);
   ll.setOrientation(LinearLayout.VERTICAL);
   ll.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT, LayoutParams
.FILL_PARENT));
   ll.setGravity(Gravity.CENTER);
   ll.addView(tv1);
   ll.addView(tv2);
   ll.addView(tv3);
   setContentView(ll);
}
```

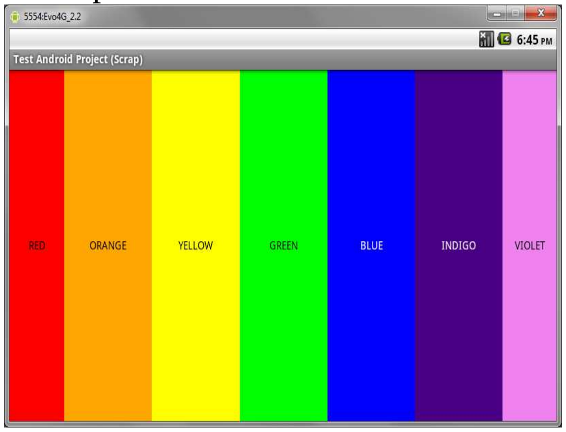
Device in both portrait and landscape modes:





Two figures show what this same layout (only changed to a horizontal orientation) as it might appear on a device in both portrait and landscape modes:





What we expect is that the red and violet areas (weight 0.15) will be slightly larger than the other colors (weight 0.14), but this is not how it displays. If you look closely at the RED TextView, it should take up more space than its neighbor ORANGE TextView. However, because "Red" is a short word, and "Orange" is not, some jostling is done automatically in order to try to keep all words from wrapping. The result is more pleasing, but it can be a bit annoying to work around if this is not the effect desired.

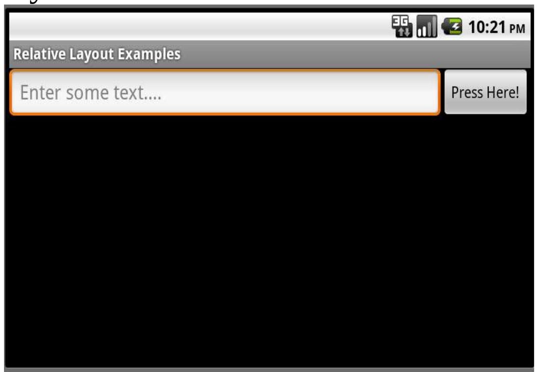
**Conclusion**

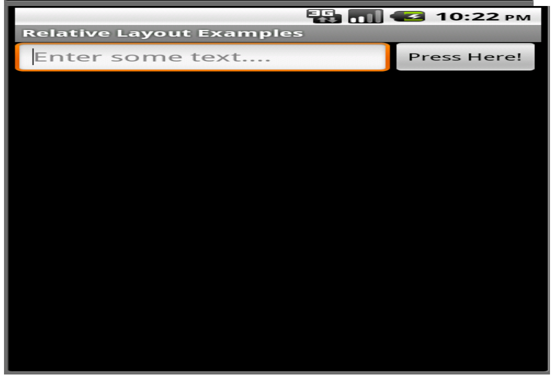
Android application user interfaces are defined using layouts, and linear layouts are one of the fundamental layout types used. The linear layout allows child controls to be organized in a single row (horizontally) or single column (vertically). The child control placement can be further adjusted using gravity and weight attributes.

***A Relative Layout***

The relative layout works much as its name implies: it organizes controls relative to one another, or to the parent control itself.

Relative layout child control placement is defined using rules. The following figures show just such a relative layout, displayed in portrait or landscape mode. The relative layout has two child controls: an EditText control and a Button control.





- **Defining an XML Layout Resource with a Relative Layout**

XML layout resources must be stored in the /res/layout project directory hierarchy. This layout resource file, aptly named /res/layout/relative.xml, is defined in XML as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_height="fill_parent"
   android:layout_width="fill_parent">
   <EditText
      android:id="@+id/EditText01"
      android:hint="Enter some text..."
      android:layout_alignParentLeft="true"
      android:layout_width="fill_parent"
      android:layout_toLeftOf="@+id/Button01"
      android:layout_height="wrap_content"></EditText>
   <Button
      android:id="@+id/Button01"
      android:text="Press Here!"
      android:layout_width="wrap_content"
      android:layout_alignParentRight="true"
      android:layout_height="wrap_content"></Button>
</RelativeLayout>
```

only a single line of code within the onCreate() method is necessary to load and display a layout resource on the screen. If the layout resource was stored in the /res/layout/relative.xml file, that line of code would be:

```
setContentView(R.layout.relative);
```

This relative layout has its width and height set to fill the screen and three rules configured on its child controls:

**EditText01:** Align to the left-hand side of the layout
**EditText01**: Display to the left of Button01
**Button01:** Align to the right-hand side of the layout

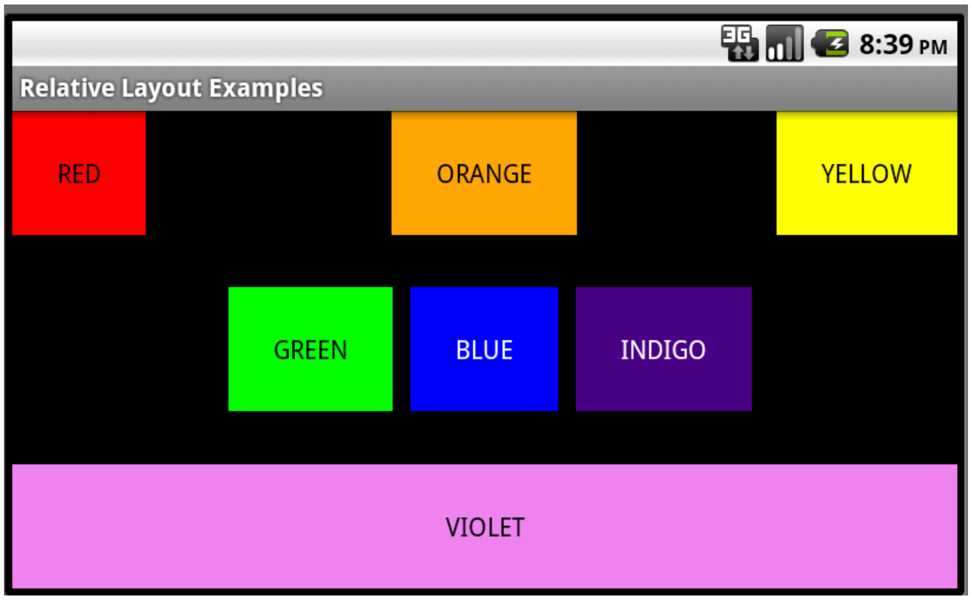
- **Defining a Relative Layout Programmatically**

The following code illustrates how to programmatically have an Activity instantiate a RelativeLayout and place a TextView and a Button control within it in its onCreate() method.

```
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   // setContentView(R.layout.relative);
      EditText ed = new EditText(this);
  RelativeLayout.LayoutParams params = new RelativeLayout.LayoutParams(LayoutParams.
FILL_PARENT,  LayoutParams.WRAP_CONTENT);
      params.addRule(RelativeLayout.ALIGN_PARENT_LEFT);
      // use same id as defined when adding the button
```

```
        params.addRule(RelativeLayout.LEFT_OF, 1001);
        ed.setLayoutParams(params);
        ed.setHint("Enter some text....");
        Button but1 = new Button(this);
RelativeLayout.LayoutParams params2 = new RelativeLayout.LayoutParams(LayoutParams.
WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
        params2.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
        but1.setLayoutParams(params2);
        but1.setText("Press Here!");
        // give the button an id that we know
        but1.setId(1001);
        RelativeLayout layout1 = new RelativeLayout(this);
        layout1.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT, LayoutPa
rams.FILL_PARENT));
        layout1.addView(ed);
        layout1.addView(but1);
        setContentView(layout1);  }
```

Design a screen that looks like this:



In order to design this screen using a relative layout, continue with the following steps.

**Step 1:** Define a Relative Layout in Your XML Resource File

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_height="fill_parent"
   android:layout_width="fill_parent">
</RelativeLayout>
```

**Step 2:** Determine Child Controls

Next, we determine what child controls that needs seven TextView controls (one for each color).

**Step 3:** Define Relative Layout Rules
Next, we define the rules for each child control, in order to get them to draw in the appropriate places:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_height="fill_parent"
   android:layout_width="fill_parent">
   <TextView
      android:text="RED"
      android:id="@+id/TextView01"
      android:layout_height="wrap_content"
      android:background="#f00"
      android:gravity="center"
      android:textColor="#000"
      android:layout_width="wrap_content"
      android:padding="25dp"></TextView>
   <TextView
      android:text="ORANGE"
      android:layout_height="wrap_content"
      android:background="#ffa500"
      android:gravity="center"
      android:textColor="#000"
      android:id="@+id/TextView02"
      android:layout_width="wrap_content"
      android:layout_centerHorizontal="true"
      android:padding="25dp"></TextView>
   <TextView
      android:text="YELLOW"
      android:layout_height="wrap_content"
      android:background="#ffff00"
      android:gravity="center"
      android:textColor="#000"
      android:id="@+id/TextView03"
      android:layout_width="wrap_content"
      android:layout_alignParentRight="true"
      android:padding="25dp"></TextView>
   <TextView
      android:text="GREEN"
      android:layout_height="wrap_content"
      android:background="#0f0"
      android:gravity="center"
      android:textColor="#000"
```

```
        android:id="@+id/TextView04"
        android:layout_width="wrap_content"
        android:layout_toLeftOf="@+id/TextView05"
        android:padding="25dp"
        android:layout_centerVertical="true"></TextView>
    <TextView
        android:text="BLUE"
        android:layout_height="wrap_content"
        android:background="#00f"
        android:gravity="center"
        android:textColor="#fff"
        android:id="@+id/TextView05"
        android:layout_width="wrap_content"
        android:layout_centerInParent="true"
        android:layout_margin="10dp"
        android:padding="25dp"></TextView>
    <TextView
        android:text="INDIGO"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textColor="#fff"
        android:id="@+id/TextView06"
        android:layout_width="wrap_content"
        android:layout_toRightOf="@+id/TextView05"
        android:background="#4b0082"
        android:padding="25dp"
        android:layout_centerVertical="true"></TextView>
    <TextView
        android:text="VIOLET"
        android:layout_height="wrap_content"
        android:background="#ee82ee"
        android:gravity="center"
        android:textColor="#000"
        android:id="@+id/TextView07"
        android:layout_alignParentBottom="true"
        android:layout_width="fill_parent"
        android:padding="25dp"></TextView>
</RelativeLayout>
```

Use relative layouts instead of nesting linear layouts to improve application performance and responsiveness.

**Conclusion**

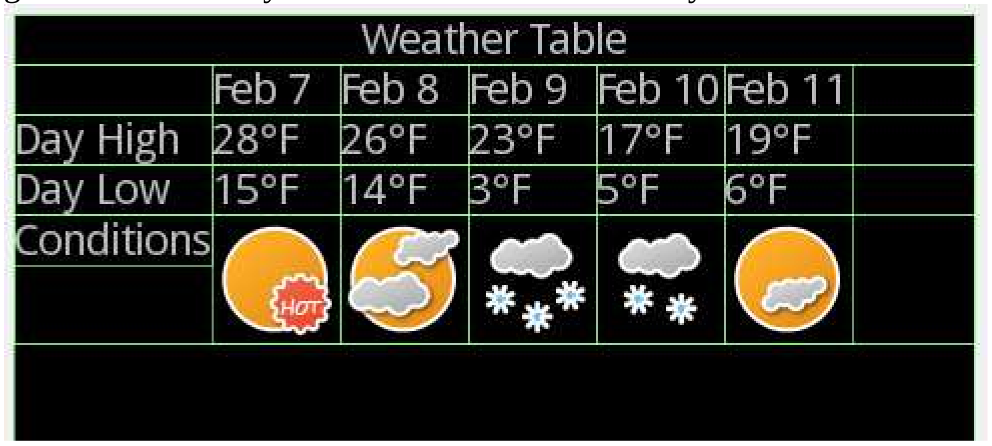
Android application user interfaces are defined using layouts, and relative layouts are one of the layout types used to make application screens that are both flexible and powerful. The relative layout allows child controls to be organized in relative to one another and relative to the parent (edges and centered vertically and horizontally).

***Table Layout***

A table layout is exactly a grid of made up of rows and columns, where a cell can display a view control. From a user interface design perspective, a TableLayout is comprised of TableRow controls—one for each row in your table. The contents of a TableRow are simply the view controls that will go in each "cell" of the table grid.
The appearance of a TableLayout is governed by several additional rules.

- **Designing a Simple Table Layout**

In the first TableRow, we can display a title for the screen.
In the second TableRow, we can display the dates in a familiar calendar-like format.
In the third TableRow, we can display a Daily High temperature information.
In the fourth TableRow, we can display a Daily Low temperature information.
In the fifth TableRow, we can display graphics to identify the weather conditions, such as rain, snow, sun, or cloudy with a chance of meatballs.
This first figure shows an early look at the table inside the layout editor:



Weather Table

| | Feb 7 | Feb 8 | Feb 9 | Feb 10 | Feb 11 |
|---|---|---|---|---|---|
| Day High | 28°F | 26°F | 23°F | 17°F | 19°F |
| Day Low | 15°F | 14°F | 3°F | 5°F | 6°F |
| Conditions | | | | | |

- **Defining an XML Layout Resource with a Table Layout**

The most convenient and maintainable way to design application user interfaces is by creating XML layout resources. This method greatly simplifies the UI design process, moving much of the static creation and layout of user interface controls and definition of control attributes, to the XML, instead of littering the code.

XML layout resources must be stored in the /res/layout project directory hierarchy. Let's take a look at the table layout introduced in the previous section. This layout resource file, aptly named /res/layout/table.xml, is defined in XML as follows:

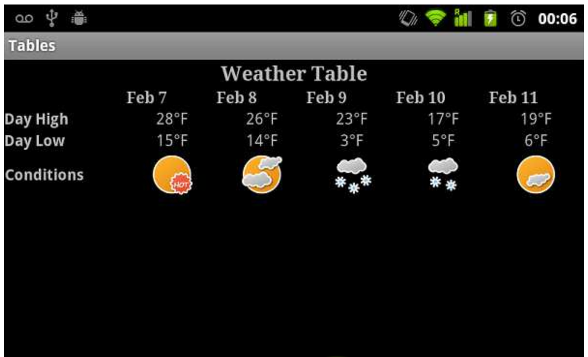
```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/tableLayout1"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:shrinkColumns="*"
   android:stretchColumns="*">
```

```
<TableRow
   android:id="@+id/tableRow4"
   android:layout_height="wrap_content"
   android:layout_width="match_parent"
   android:gravity="center_horizontal">
   <TextView
      android:id="@+id/textView9"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textStyle="bold"
      android:typeface="serif"
      android:textSize="18dp"
      android:text="Weather Table"
      android:gravity="center"
      android:layout_span="6"></TextView>
</TableRow>
<TableRow
   android:id="@+id/tableRow1"
   android:layout_height="wrap_content"
   android:layout_width="match_parent">
   <TextView
      android:id="@+id/TextView04"
      android:text=""></TextView>
   <TextView
      android:id="@+id/TextView04"
      android:text="Feb 7"
      android:textStyle="bold"
      android:typeface="serif"></TextView>
   <TextView
      android:id="@+id/TextView03"
      android:text="Feb 8"
      android:textStyle="bold"
      android:typeface="serif"></TextView>
   <TextView
      android:id="@+id/TextView02"
      android:text="Feb 9"
      android:textStyle="bold"
      android:typeface="serif"></TextView>
   <TextView
      android:id="@+id/TextView01"
      android:text="Feb 10"
      android:textStyle="bold"
      android:typeface="serif"></TextView>
```

```
    <TextView
        android:text="Feb 11"
        android:id="@+id/textView1"
        android:textStyle="bold"
        android:typeface="serif"></TextView>
</TableRow>
<TableRow
    android:layout_height="wrap_content"
    android:id="@+id/tableRow2"
    android:layout_width="match_parent">
    <TextView
        android:text="Day High"
        android:id="@+id/textView2"
        android:textStyle="bold"></TextView>
    <TextView
        android:id="@+id/textView3"
        android:text="28°F"
        android:gravity="center_horizontal"></TextView>
    <TextView
        android:text="26°F"
        android:id="@+id/textView4"
        android:gravity="center_horizontal"></TextView>
    <TextView
        android:text="23°F"
        android:id="@+id/textView5"
        android:gravity="center_horizontal"></TextView>
    <TextView
        android:text="17°F"
        android:id="@+id/textView6"
        android:gravity="center_horizontal"></TextView>
    <TextView
        android:text="19°F"
        android:id="@+id/textView7"
        android:gravity="center_horizontal"></TextView>
</TableRow>
<TableRow
    android:layout_height="wrap_content"
    android:id="@+id/tableRow2"
    android:layout_width="match_parent">
    <TextView
        android:text="Day Low"
        android:id="@+id/textView2"
        android:textStyle="bold"></TextView>
    <TextView
        android:text="15°F"
        android:id="@+id/textView3"
        android:gravity="center_horizontal"></TextView>
    <TextView
        android:text="14°F"
        android:id="@+id/textView4"
        android:gravity="center_horizontal"></TextView>
    <TextView
        android:text="3°F"
        android:id="@+id/textView5"
        android:gravity="center_horizontal"></TextView>
    <TextView
        android:text="5°F"
        android:id="@+id/textView6"
        android:gravity="center_horizontal"></TextView>
    <TextView
        android:text="6°F"
        android:id="@+id/textView7"
        android:gravity="center_horizontal"></TextView>
</TableRow>
<TableRow
    android:id="@+id/tableRow3"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:gravity="center">
    <TextView
        android:id="@+id/textView8"
        android:text="Conditions"
        android:textStyle="bold"></TextView>
    <ImageView
        android:id="@+id/imageView1"
        android:src="@drawable/hot"></ImageView>
    <ImageView
        android:id="@+id/imageView2"
        android:src="@drawable/pt_cloud"></ImageView>
    <ImageView
        android:id="@+id/imageView3"
        android:src="@drawable/snow"></ImageView>
    <ImageView
        android:id="@+id/imageView4"
        android:src="@drawable/lt_snow"></ImageView>
    <ImageView
```

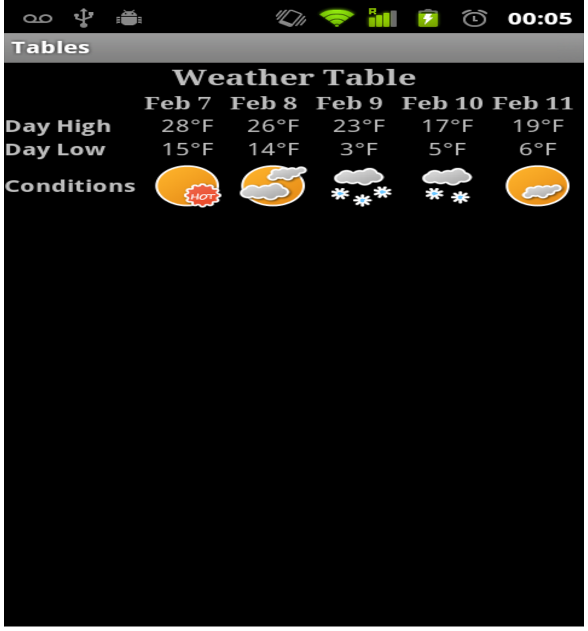
```
            android:id="@+id/imageView5"
            android:src="@drawable/pt_sun"></ImageView>
    </TableRow>
</TableLayout>
```

Only a single line of code within the onCreate() method is necessary to load and display a layout resource on the screen. If the layout resource was stored in the /res/layout/table.xml file, that line of code would be:  setContentView(R.layout.table);
This table layout has all its columns set to both shrink and stretch by using a “*” in the value. If just certain columns should shrink or stretch, the values would be a comma seperated list (using 0-based indexes for columns).
The table now looks like the following to screenshots when in portrait and landscape mode.



Weather Table

| | Feb 7 | Feb 8 | Feb 9 | Feb 10 | Feb 11 |
|---|---|---|---|---|---|
| Day High | 28°F | 26°F | 23°F | 17°F | 19°F |
| Day Low | 15°F | 14°F | 3°F | 5°F | 6°F |
| Conditions | | | | | |



Weather Table

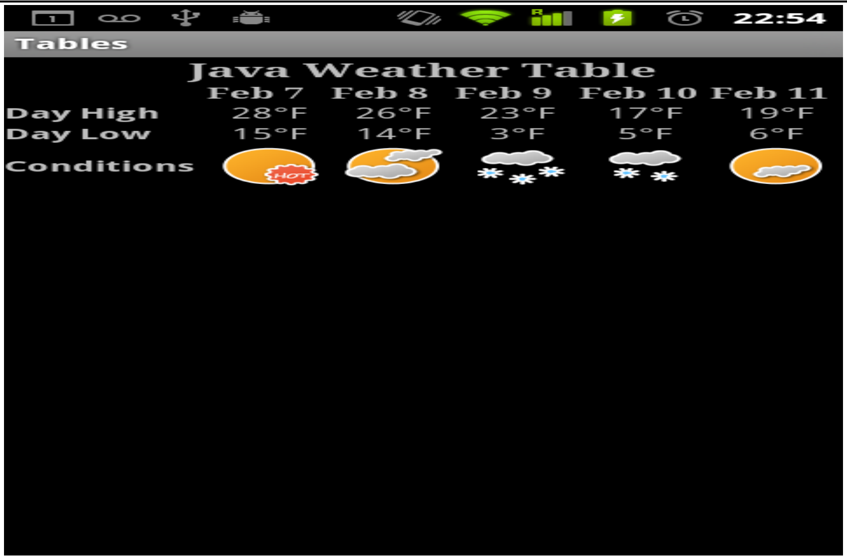
| | Feb 7 | Feb 8 | Feb 9 | Feb 10 | Feb 11 |
|---|---|---|---|---|---|
| Day High | 28°F | 26°F | 23°F | 17°F | 19°F |
| Day Low | 15°F | 14°F | 3°F | 5°F | 6°F |
| Conditions | | | | | |

➢ **Defining a Table Layout Programmatically**

Programmatically have an Activity instantiate a TableLayout layout parameters and reproduce the example shown earlier in XML:

```
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   TableLayout table = new TableLayout(this);
   table.setStretchAllColumns(true);
   table.setShrinkAllColumns(true);
   TableRow rowTitle = new TableRow(this);
   rowTitle.setGravity(Gravity.CENTER_HORIZONTAL);
   TableRow rowDayLabels = new TableRow(this);
   TableRow rowHighs = new TableRow(this);
   TableRow rowLows = new TableRow(this);
   TableRow rowConditions = new TableRow(this);
   rowConditions.setGravity(Gravity.CENTER);
   TextView empty = new TextView(this);
   // title column/row
   TextView title = new TextView(this);
   title.setText("Java Weather Table");
   title.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 18);
   title.setGravity(Gravity.CENTER);
   title.setTypeface(Typeface.SERIF, Typeface.BOLD);
   TableRow.LayoutParams params = new TableRow.LayoutParams();
   params.span = 6;
   rowTitle.addView(title, params);
   // labels column
   TextView highsLabel = new TextView(this);
   highsLabel.setText("Day High");
   highsLabel.setTypeface(Typeface.DEFAULT_BOLD);
   TextView lowsLabel = new TextView(this);
   lowsLabel.setText("Day Low");
   lowsLabel.setTypeface(Typeface.DEFAULT_BOLD);
   TextView conditionsLabel = new TextView(this);
   conditionsLabel.setText("Conditions");
   conditionsLabel.setTypeface(Typeface.DEFAULT_BOLD);
   rowDayLabels.addView(empty);
   rowHighs.addView(highsLabel);
   rowLows.addView(lowsLabel);
   rowConditions.addView(conditionsLabel);
   // day 1 column
   TextView day1Label = new TextView(this);
   day1Label.setText("Feb 7");
   day1Label.setTypeface(Typeface.SERIF, Typeface.BOLD);
   TextView day1High = new TextView(this);
   day1High.setText("28°F");
   day1High.setGravity(Gravity.CENTER_HORIZONTAL);
   TextView day1Low = new TextView(this);
   day1Low.setText("15°F");
   day1Low.setGravity(Gravity.CENTER_HORIZONTAL);
   ImageView day1Conditions = new ImageView(this);
   day1Conditions.setImageResource(R.drawable.hot);
   rowDayLabels.addView(day1Label);
   rowHighs.addView(day1High);
   rowLows.addView(day1Low);
   rowConditions.addView(day1Conditions);
   // day2 column
   TextView day2Label = new TextView(this);
   day2Label.setText("Feb 8");
   day2Label.setTypeface(Typeface.SERIF, Typeface.BOLD);
   TextView day2High = new TextView(this);
   day2High.setText("26°F");
   day2High.setGravity(Gravity.CENTER_HORIZONTAL);
   TextView day2Low = new TextView(this);
   day2Low.setText("14°F");
   day2Low.setGravity(Gravity.CENTER_HORIZONTAL);
   ImageView day2Conditions = new ImageView(this);
   day2Conditions.setImageResource(R.drawable.pt_cloud);
   rowDayLabels.addView(day2Label);
   rowHighs.addView(day2High);
   rowLows.addView(day2Low);
   rowConditions.addView(day2Conditions);
   // day3 column
   TextView day3Label = new TextView(this);
   day3Label.setText("Feb 9");
   day3Label.setTypeface(Typeface.SERIF, Typeface.BOLD);
   TextView day3High = new TextView(this);
   day3High.setText("23°F");
   day3High.setGravity(Gravity.CENTER_HORIZONTAL);
   TextView day3Low = new TextView(this);
   day3Low.setText("3°F");
   day3Low.setGravity(Gravity.CENTER_HORIZONTAL);
   ImageView day3Conditions = new ImageView(this);
   day3Conditions.setImageResource(R.drawable.snow);
   rowDayLabels.addView(day3Label);
   rowHighs.addView(day3High);
   rowLows.addView(day3Low);
   rowConditions.addView(day3Conditions);
   // day4 column
TextView day4Label = new TextView(this);
   day4Label.setText("Feb 10");
```

```
    day4Label.setTypeface(Typeface.SERIF, Typeface.BOLD);
    TextView day4High = new TextView(this);
    day4High.setText("17°F");
    day4High.setGravity(Gravity.CENTER_HORIZONTAL);
    TextView day4Low = new TextView(this);
    day4Low.setText("5°F");
    day4Low.setGravity(Gravity.CENTER_HORIZONTAL);
    ImageView day4Conditions = new ImageView(this);
    day4Conditions.setImageResource(R.drawable.lt_snow);
    rowDayLabels.addView(day4Label);
    rowHighs.addView(day4High);
    rowLows.addView(day4Low);
    rowConditions.addView(day4Conditions);
    // day5 column
    TextView day5Label = new TextView(this);
    day5Label.setText("Feb 11");
    day5Label.setTypeface(Typeface.SERIF, Typeface.BOLD);
    TextView day5High = new TextView(this);
    day5High.setText("19°F");
    day5High.setGravity(Gravity.CENTER_HORIZONTAL);
    TextView day5Low = new TextView(this);
    day5Low.setText("6°F");
    day5Low.setGravity(Gravity.CENTER_HORIZONTAL);
    ImageView day5Conditions = new ImageView(this);
    day5Conditions.setImageResource(R.drawable.pt_sun);
    rowDayLabels.addView(day5Label);
    rowHighs.addView(day5High);
    rowLows.addView(day5Low);
    rowConditions.addView(day5Conditions);
    table.addView(rowTitle);
    table.addView(rowDayLabels);
    table.addView(rowHighs);
    table.addView(rowLows);
    table.addView(rowConditions);
    setContentView(table);
}
```



22:54

Tables

Java Weather Table

| | Feb 7 | Feb 8 | Feb 9 | Feb 10 | Feb 11 |
|---|---|---|---|---|---|
| Day High | 28°F | 26°F | 23°F | 17°F | 19°F |
| Day Low | 15°F | 14°F | 3°F | 5°F | 6°F |
| Conditions | | | | | |

**Conclusion**

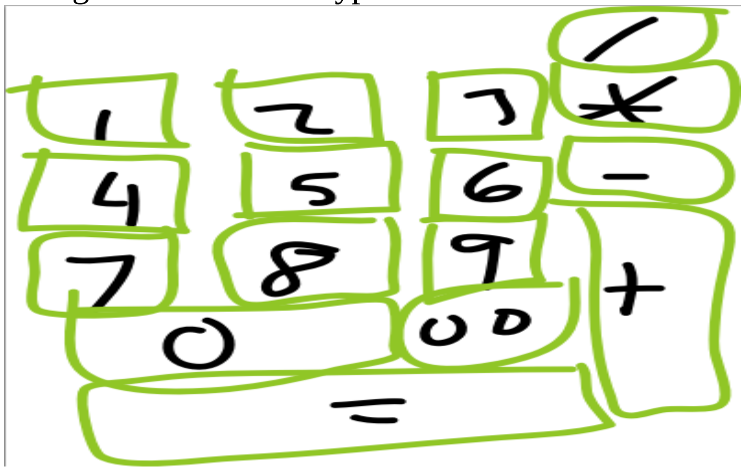
Android application user interfaces are defined using layouts, and table layouts are incredibly handy for displaying view data or controls in rows and columns. Using table layouts where they are appropriate can make many screen designs simpler and faster..

***GRID LAYOUT:***

The new GridLayout class even exists in Android 4.0 (aka Ice Cream Sandwich). It sounds a lot like TableLayout. In fact, it's a very useful new layout control. A simple numeric keypad using GridLayout to demonstrate a small taste of its power and elegance.

**Step 1:** Planning for the Keypad

The following shows a rough sketch of the keypad we will build.



Some things of note for the layout:

5 rows, 4 columns

Both column span and row span are used

Not all cells are populated

But in Android 4.0, there's a more efficient control that suits our purposes: GridLayout.

**Step 2:** Identifying a Grid Strategy

GridLayout controls, like LinearLayout controls, can have horizontal and vertical orientations. That is, setting a vertical orientation means the next cell will be down a row from the current one and possibly moving right to the next column. Horizontal orientation means the next cell is

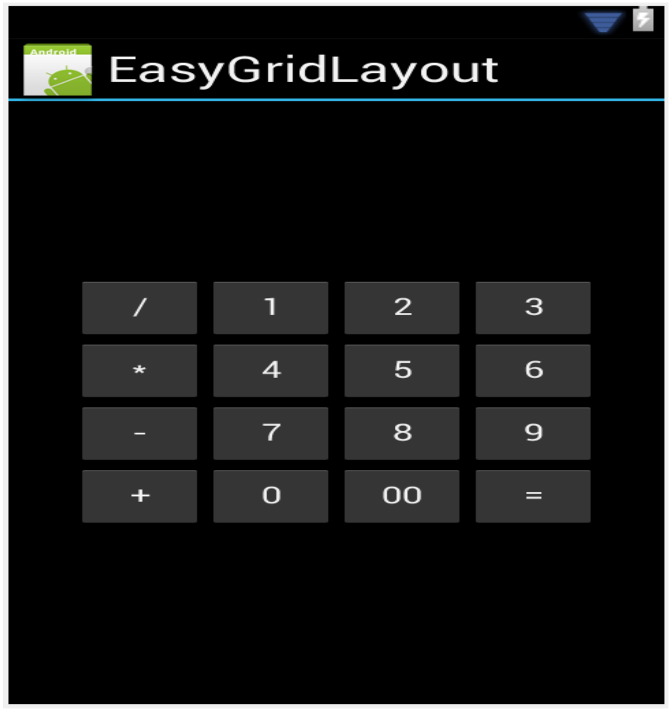
to the right, and also possibly wrapping around to the next row, starting on the left.
Finally, we want the View control in each cell (in this case, these are Button controls) to be centered and we want the whole layout to size itself to the content.
The following XML defines the GridLayout container we'll need:

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:columnCount="4"
    android:orientation="horizontal" >                    </GridLayout>
```

**Step 3:** Defining the Simple Cells

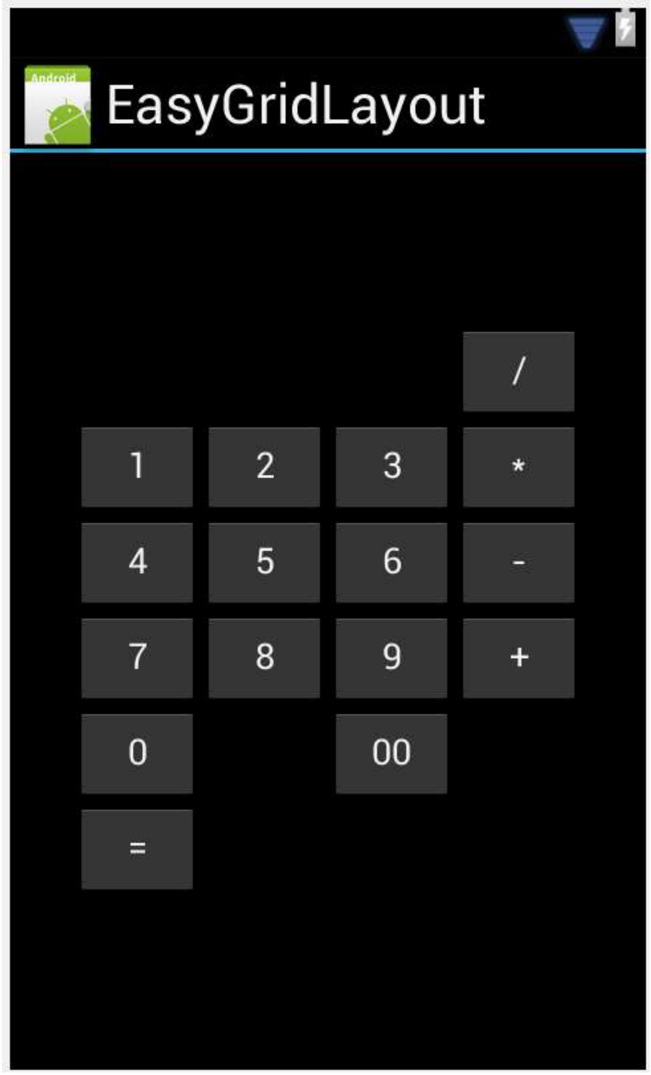
Each cell will contain a single Button control with a text label. Therefore, each of the simple cells is merely defined as follows:

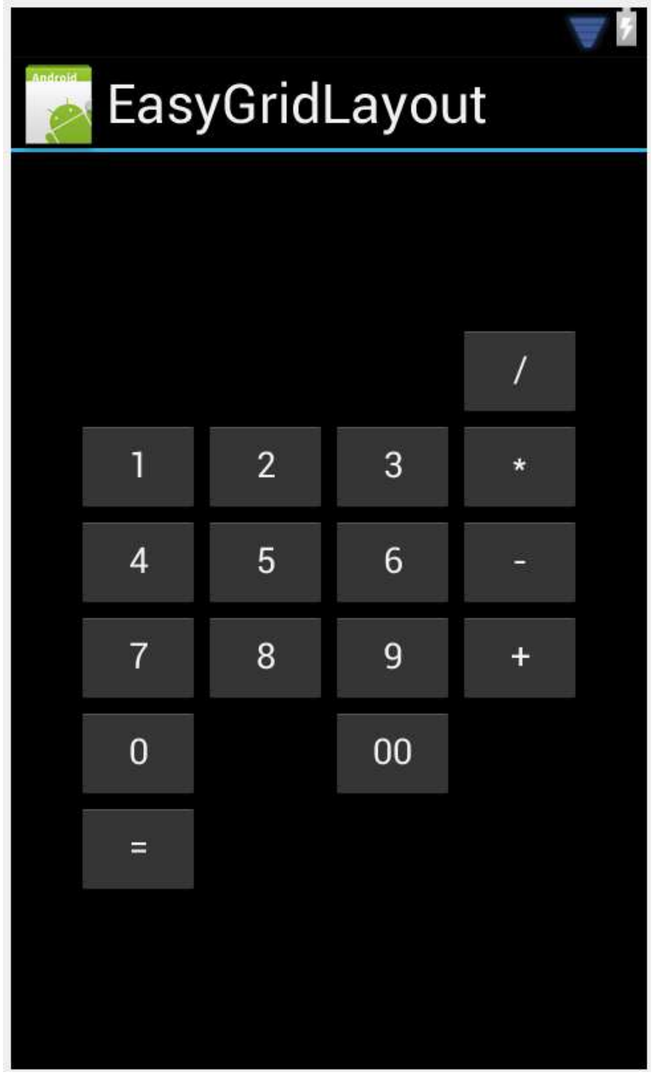
```
<Button android:text="1" />
<Button android:text="2" />  <!-- and so on... -->
```



**Step 4:** Defining the Rest of the Cells
The current layout isn't exactly what we want. The /, +, 0, and = Button controls are all special when it comes to laying them out properly.
The / (division sign or forward slash) Button control retains its current size, but it should start in the 4th column.
The + (plus sign) Button control first appears in the horizontal orientation direction directly after the 9 button, but it should span three rows.
The 0 (zero) Button control should span two columns.
The = (equal sign) button should span three columns.
Applying these subtle changes to the GridLayout results in the following XML definition:

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_gravity="center"
   android:columnCount="4"
   android:orientation="horizontal" >
   <Button
      android:layout_column="3"
      android:text="/" />
   <Button android:text="1" />
<!-- Other numbers -->
   <Button android:text="9" />
   <Button
      android:layout_rowSpan="3"
      android:text="+" />
   <Button
      android:layout_columnSpan="2"
      android:text="0" />
   <Button android:text="00" />
   <Button
      android:layout_columnSpan="3"
      android:text="=" />
</GridLayout>
```

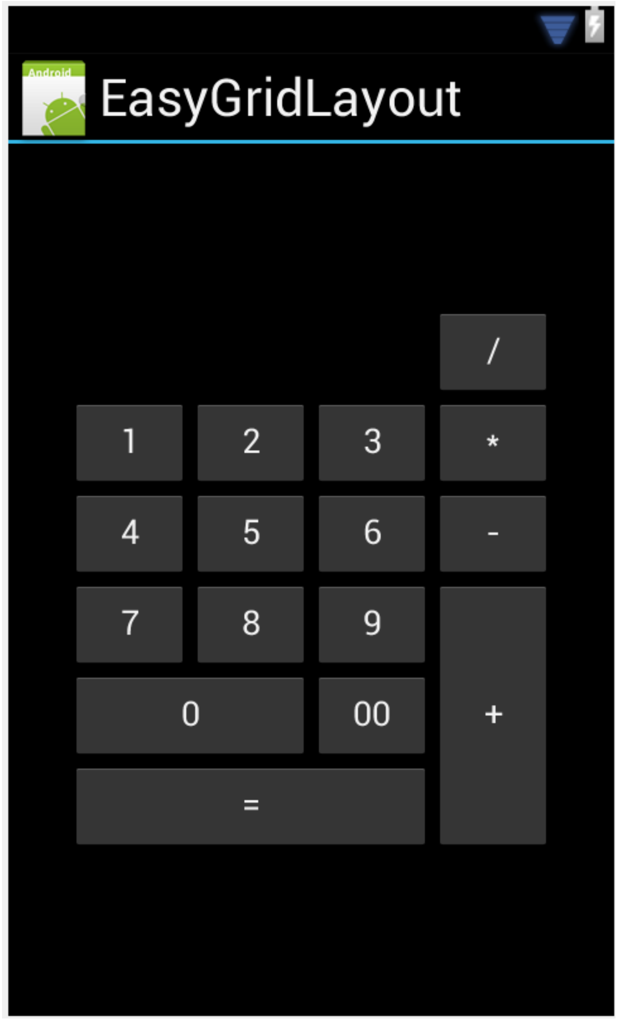

EasyGridLayout
/
1
2
3
*
4
5
6
-
7
8
9
+
0
00
=



**Step 5:** Filling in the Holes
The width and height values of the Button controls are not yet correct. You might immediately think that the solution is to adjust the layout_width and layout_height. But remember, the values for automatic scaling, just as wrap_content and match_parent, both behave the same and are already applied.

```
        <?xml version="1.0" encoding="utf-8"?>
       3<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_gravity="center"
           android:columnCount="4"
           android:orientation="horizontal" >
           <Button
              android:layout_column="3"
              android:text="/" />
           <Button android:text="1" />
           <Button android:text="2" />
           <Button android:text="3" />
```

```
    <Button android:text="*" />
    <Button android:text="4" />
    <Button android:text="5" />
    <Button android:text="6" />
    <Button android:text="-" />
    <Button android:text="7" />
    <Button android:text="8" />
    <Button android:text="9" />
    <Button
        android:layout_gravity="fill"
        android:layout_rowSpan="3"
        android:text="+" />
    <Button
        android:layout_columnSpan="2"
        android:layout_gravity="fill"
        android:text="0" />
    <Button android:text="00" />
    <Button
        android:layout_columnSpan="3"
        android:layout_gravity="fill"
        android:text="=" />
</GridLayout>
```



**Conclusion**

While GridLayout isn't just for use with items that line up in a regular sized table-like layout, it may be easier to use than TableLayout for such designs.Any layout that can be defined in terms of grid lines — not just cells — can likely be done with less effort and better performance in a Grid Layout than other container types. The new GridLayout control for Android 4.0 is very powerful .

## WEEK-11

Android Application Program that converts the temperature in Celsius to Fahrenheit.

**SOURCE CODE:**

1.) Create a new project by File-> New ->Android Project name it convertTemperatureExample.

2.) Write following into main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/myshape"
    android:orientation="vertical" >
     <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"     >
    </EditText>
     <RadioGroup
        android:id="@+id/radioGroup1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
         <RadioButton
            android:id="@+id/radio0"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:checked="true"
            android:text="@string/celsius" >
        </RadioButton>
         <RadioButton
            android:id="@+id/radio1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/fahrenheit" >
        </RadioButton>
    </RadioGroup>
     <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/calc"
        android:onClick="myClickHandler">
    </Button>
</LinearLayout>
```

3.) Create and write following into res/drawable/myshape.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <stroke
        android:width="2dp"
        android:color="#FFFFFFFF" />
    <gradient
```

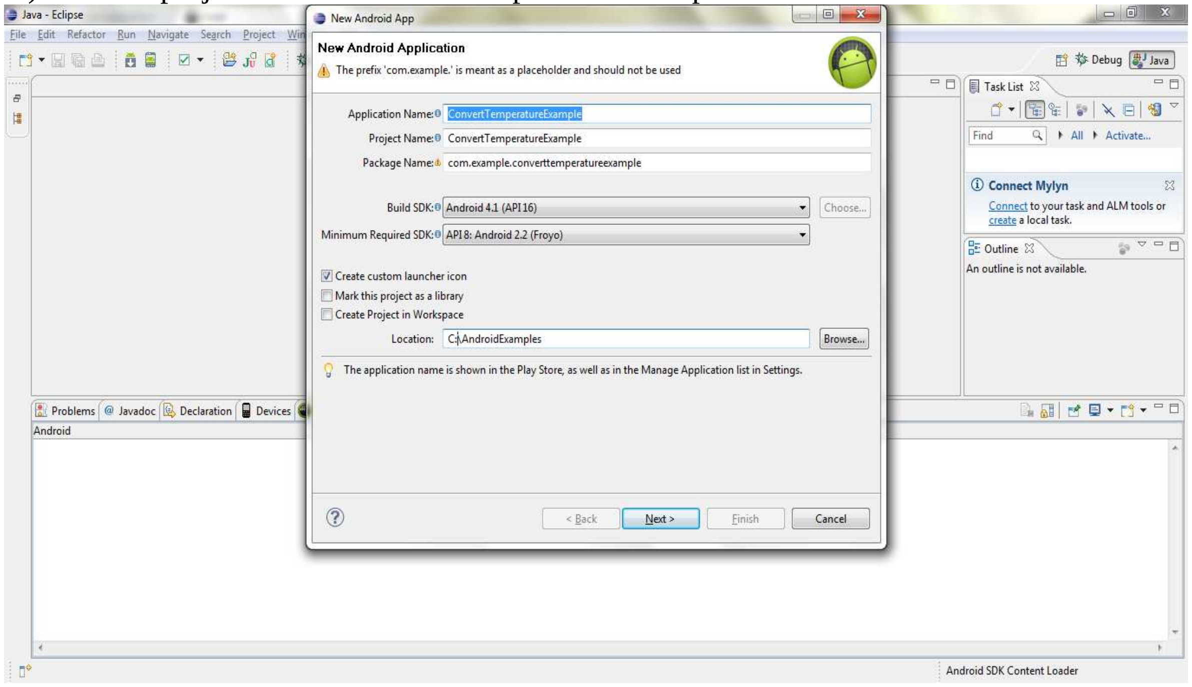
```
          android:endColor="#DDBBBBBB"
          android:startColor="#DD777777"
          android:angle="90" />
      <corners
          android:bottomRightRadius="7dp"
          android:bottomLeftRadius="7dp"
          android:topLeftRadius="7dp"
          android:topRightRadius="7dp" />
  </shape>
```

4.) Create and write following into res/values/strings.xml:

```
<resources>
     <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string
name="title_activity_convert_temperture_example">ConvertTempertureExample</string>
    <string name="app_name">Temparature Converter</string>
    <color name="myColor">#3399CC</color>
    <string name="myClickHandler">myClickHandler</string>
    <string name="celsius">to Celsius</string>
    <string name="fahrenheit">to Fahrenheit</string>
    <string name="calc">Calculate</string>
 </resources>
```

5. Run for output

1.) Create a project named ConvertTemperatureExample and set the information as stated in the image.



New Android App

New Android Application

The prefix 'com.example.' is meant as a placeholder and should not be used

Application Name: ConvertTemperatureExample

Project Name: ConvertTemperatureExample

Package Name: com.example.converttemperatureexample

Build SDK: Android 4.1 (API 16)

Minimum Required SDK: API 8: Android 2.2 (Froyo)

☑ Create custom launcher icon

☐ Mark this project as a library

☐ Create Project in Workspace

Location: C:\AndroidExamples

The application name is shown in the Play Store, as well as in the Manage Application list in Settings.

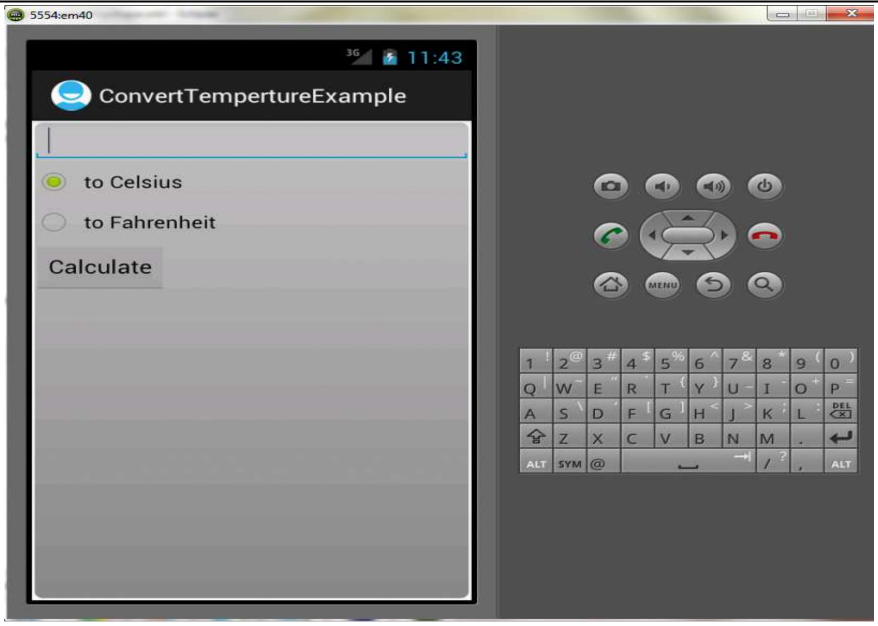
Java - Eclipse
File Edit Refactor Run Navigate Search Project
Debug Java
Task List
Find All Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems Javadoc Declaration Devices
Android
Choose...
Browse...
< Back
Next >
Finish
Cancel
Android SDK Content Loader

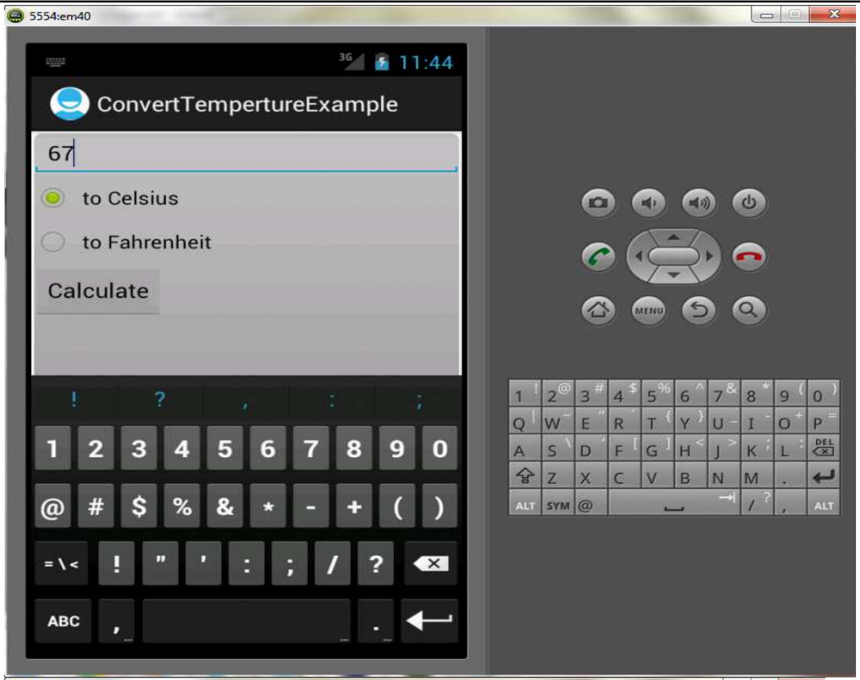
Build Target: Android 4.0
Application Name: ConvertTemperatureExample
Package Name: com. example. ConvertTemperatureExample
Activity Name: ConvertTemperatureExample
Min SDK Version: 8

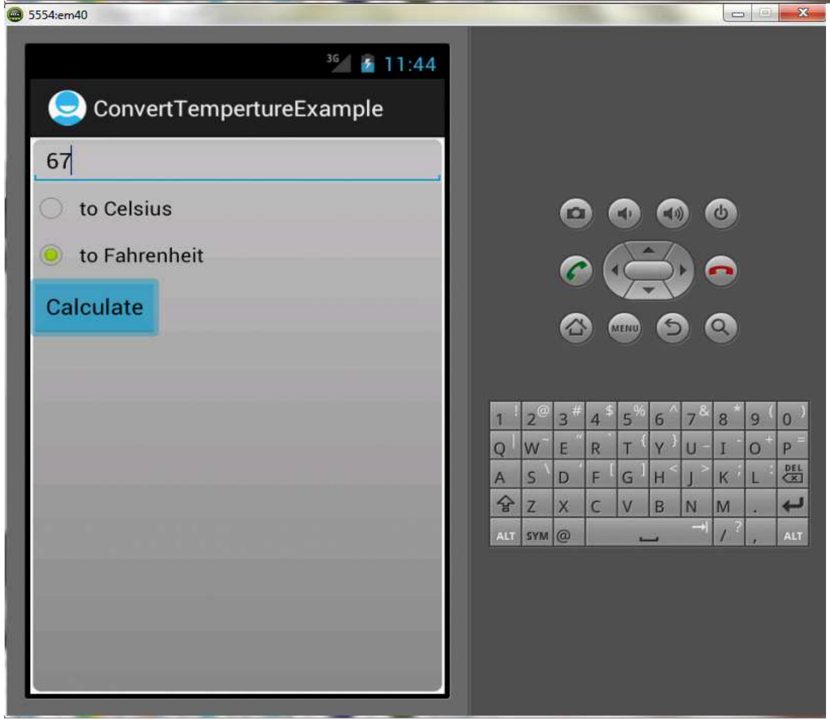
2.) Open ConvertTemperatureExample.java file and write following code there:

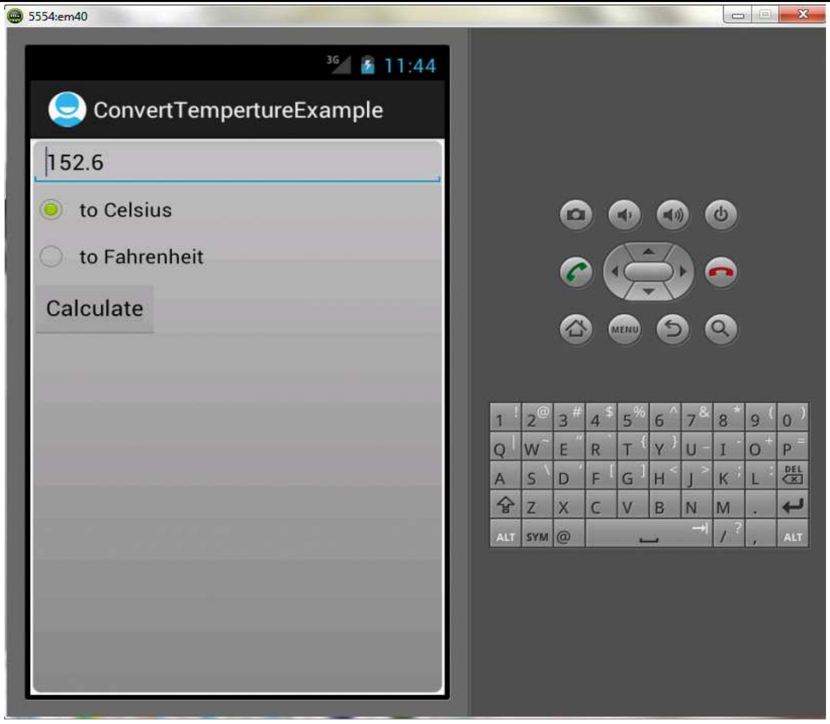
```
package com.example.converttemperatureexample;
```

```
 import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.Toast;
 public class ConvertTempertureExample extends Activity {
    private EditText text;
     @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        text = (EditText) findViewById(R.id.editText1);    }
     // This method is called at button click because we assigned the name to the
    // "On Click property" of the button
    public void myClickHandler(View view) {
        switch (view.getId()) {
        case R.id.button1:
            RadioButton celsiusButton = (RadioButton) findViewById(R.id.radio0);
            RadioButton fahrenheitButton = (RadioButton) findViewById(R.id.radio1);
            if (text.getText().length() == 0) {
                Toast.makeText(this, "Please enter a valid number",
                        Toast.LENGTH_LONG).show();
                return;
            }
             float inputValue = Float.parseFloat(text.getText().toString());
            if (celsiusButton.isChecked()) {
                text.setText(String
                        .valueOf(convertFahrenheitToCelsius(inputValue)));
                celsiusButton.setChecked(false);
                fahrenheitButton.setChecked(true);
            } else {
                text.setText(String
                        .valueOf(convertCelsiusToFahrenheit(inputValue)));
                fahrenheitButton.setChecked(false);
                celsiusButton.setChecked(true);
            }
            break;
        }
    }
     // Converts to celsius
    private float convertFahrenheitToCelsius(float fahrenheit) {
        return ((fahrenheit - 32) * 5 / 9);
    }
     // Converts to fahrenheit
    private float convertCelsiusToFahrenheit(float celsius) {
        return ((celsius * 9) / 5) + 32;
    }
}
```


5554:em40
11:43
ConvertTempertureExample
to Celsius
to Fahrenheit
Calculate


5554:em40
11:44
ConvertTempertureExample
67
to Celsius
to Fahrenheit
Calculate


5554:em40
11:44
ConvertTempertureExample
67
to Celsius
to Fahrenheit
Calculate

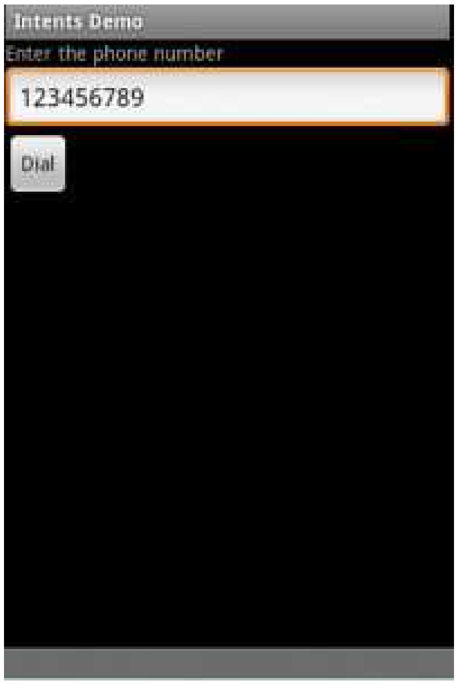

5554:em40
11:44
ConvertTempertureExample
152.6
to Celsius
to Fahrenheit
Calculate

WEEK 12

Android Application Program that demonstrates intent in mobile application development.

SOURCE CODE:

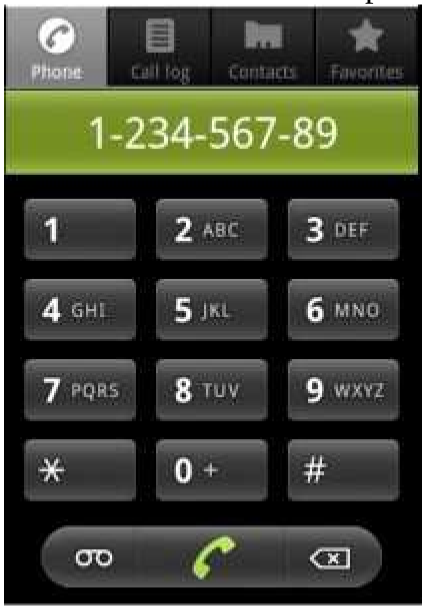
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"    >
<TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="Enter the phone number"     />
<EditText
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:id="@+id/txtNumber"
android:inputType="phone" />
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Dial"
android:id="@+id/btnDial" />
</LinearLayout>
```



When you press the button the phone dialer launches and then you can call the number. This is done using the following code:

```
btnDial.setOnClickListener(new OnClickListener() {
  public void onClick(View v) {
   // TODO Auto-generated method stub
Intent dialIntent=new
Intent(Intent.ACTION_DIAL,Uri.parse("tel:"+(txtNumber.getText()).toString()));
startActivity(dialIntent);
  }
 });
```

Notice that the dialer has been launched but the user has to press the call button to make a call.



If you want the phone to dial the number automatically you could have used this intent: Intent.ACTION_CALL

If you use this it requires adding the following permission to the manifest file:

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

Another example to launch an intent to open the browser and navigate to a certain URL:

```
Intent in=new Intent(Intent.ACTION_VIEW, Uri.parse("http://mobileorchard.com"));
            startActivity(in);
```

**Intent Filters:**

Create a new android project, create an activity and name it **Dialer.**

In the AndroidManifest.xml file of this application add the following to the **Dialer** activity:

```
<intent-filter android:priority="100" >
          <action android:name="android.intent.action.DIAL"  />
             <category android:name="android.intent.category.DEFAULT" />
             <data android:scheme="tel"/>
          </intent-filter>
```

**to become:**

```
<activity android:name=".Dialer"
               android:label="@string/app_name">
          <intent-filter>
             <action android:name="android.intent.action.MAIN" />
             <category android:name="android.intent.category.LAUNCHER" />
```

```
        </intent-filter>
        <intent-filter android:priority="100" >
        <action android:name="android.intent.action.DIAL"  />
           <category android:name="android.intent.category.DEFAULT" />
           <data android:scheme="tel"/>
        </intent-filter>

    </activity>
```

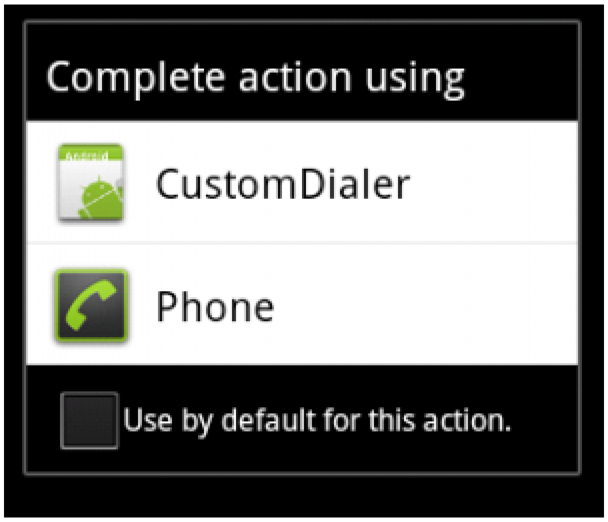
This intent filter has the following properties:

1. Action: the type of implicit intents that this activity responds to. in our case it is the dial action. higher numbers represent higher priority.
2. Priority: about the priority of that activity over other activities that respond to the same type of intents.
3. Category: Implicit intents have built-in categories. in order that an implicit intent be captured by our activity, the implicit intent category must match our activity category.
4. Data: adds data specification scheme to the intent filter.

So if any other application has the following module to launch the dialer:

```
Intent in=new Intent(Intent.ACTION_DIAL, Uri.parse("tel:000"));
          startActivity(in);
```

The user will see the following dialog offering him/her the choice between the default dialer and our custom dialer.

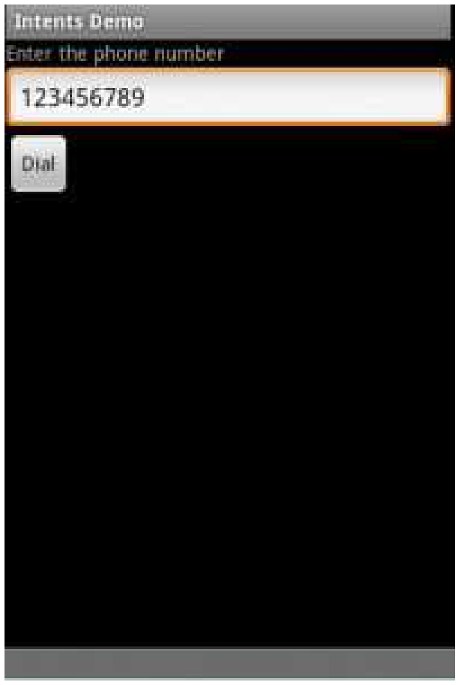


WEEK 12

Android Application Program that demonstrates intent in mobile application development.

SOURCE CODE:

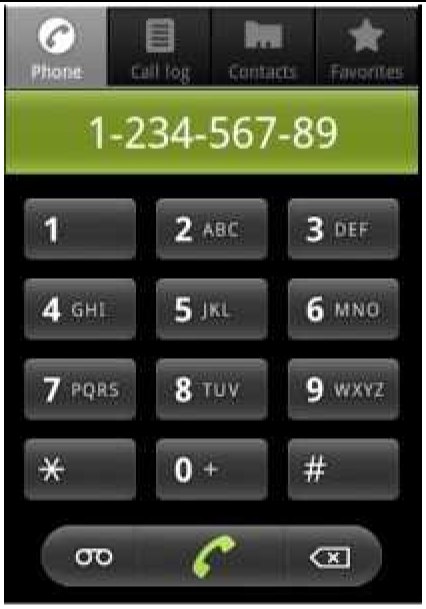
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"     >
<TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="Enter the phone number"     />
<EditText
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:id="@+id/txtNumber"
android:inputType="phone" />
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Dial"
android:id="@+id/btnDial" />
</LinearLayout>
```



When you press the button the phone dialer launches and then you can call the number.
This is done using the following code:

```
btnDial.setOnClickListener(new OnClickListener() {
  public void onClick(View v) {
   // TODO Auto-generated method stub
Intent dialIntent=new
Intent(Intent.ACTION_DIAL,Uri.parse("tel:"+(txtNumber.getText()).toString()));
startActivity(dialIntent);
   }
  });
```

Notice that the dialer has been launched but the user has to press the call button to make a call.



If you want the phone to dial the number automatically you could have used this intent: Intent.ACTION_CALL

If you use this it requires adding the following permission to the manifest file:

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

Another example to launch an intent to open the browser and navigate to a certain URL:

```
Intent in=new Intent(Intent.ACTION_VIEW, Uri.parse("http://mobileorchard.com"));
            startActivity(in);
```

**Intent Filters:**

Create a new android project, create an activity and name it **Dialer.**

In the AndroidManifest.xml file of this application add the following to the **Dialer** activity:

```
<intent-filter android:priority="100" >
          <action android:name="android.intent.action.DIAL"  />
             <category android:name="android.intent.category.DEFAULT" />
             <data android:scheme="tel"/>
          </intent-filter>
```

**to become:**

```
<activity android:name=".Dialer"
                android:label="@string/app_name">
          <intent-filter>
             <action android:name="android.intent.action.MAIN" />
             <category android:name="android.intent.category.LAUNCHER" />
          </intent-filter>
          <intent-filter android:priority="100" >
          <action android:name="android.intent.action.DIAL"  />
             <category android:name="android.intent.category.DEFAULT" />
             <data android:scheme="tel"/>
          </intent-filter>
       </activity>
```

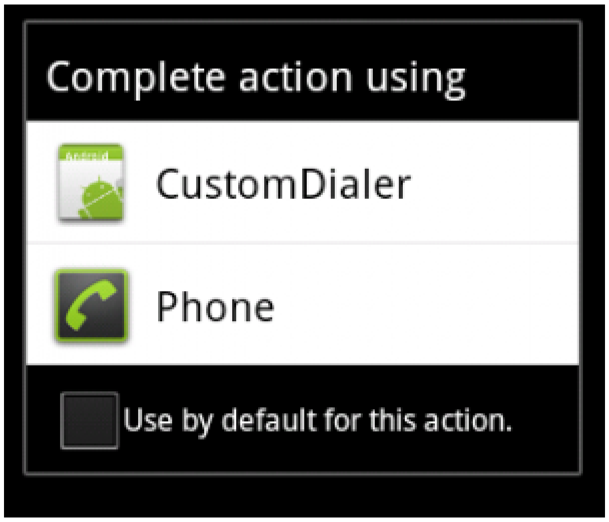
This intent filter has the following properties:

1. Action: the type of implicit intents that this activity responds to. in our case it is the dial action. higher numbers represent higher priority.
2. Priority: about the priority of that activity over other activities that respond to the same type of intents.
3. Category: Implicit intents have built-in categories. in order that an implicit intent be captured by our activity, the implicit intent category must match our activity category.
4. Data: adds data specification scheme to the intent filter.

So if any other application has the following module to launch the dialer:

```
Intent in=new Intent(Intent.ACTION_DIAL, Uri.parse("tel:000"));
        startActivity(in);
```

The user will see the following dialog offering him/her the choice between the default dialer and our custom dialer.


Complete action using
CustomDialer
Phone
Use by default for this action.