ST. ANNE'S

COLLEGE OF ENGINEERING AND TECHNOLOGY

ANGUCHETTYPALAYAM, PANRUTI – 607 110

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

CS6611- MOBILE APPLICATION DEVELOPMENT LABORATORY

Regulation 2013

Year / Semester: III / VI

PREPARED BY

Mr. S. MANAVALAN, M.Tech.,
Associate Professor / CSE

SYLLABUS

CS6611- MOBILE APPLICATION DEVELOPMENT LABORATORY

LIST OF EXPERIMENTS:

1. Develop an application that uses GUI components, Font and Colours.
2. Develop an application that uses Layout Managers and event listeners.
3. Develop a native calculator application.
4. Write an application that draws basic graphical primitives on the screen.
5. Develop an application that makes use of database.
6. Develop an application that makes use of RSS Feed.
7. Implement an application that implements Multi threading
8. Develop a native application that uses GPS location information.
9. Implement an application that writes data to the SD card.
10. Implement an application that creates an alert upon receiving a message.
11. Write a mobile application that creates alarm clock.

TOTAL: 45 PERIODS

TABLE OF CONTENTS

S.NO.	DATE	EXPERIMENT TITLE	MARKS/ 10	SIGN.
1.		Develop an application that uses GUI components, Font and Colours		
2.		Develop an application that uses Layout Managers and event listeners.		
3.		Develop a native calculator application.		
4.		Write an application that draws basic graphical primitives on the screen.		
5.		Develop an application that makes use of database.		
6.		Develop an application that makes use of RSS Feed.		
7.		Implement an application that implements Multi threading .		
8.		Develop a native application that uses GPS location information.		
9.		Implement an application that writes data to the SD card.		
10.		Implement an application that creates an alert upon receiving a message.		
11.		Write a mobile application that creates alarm clock.		

Ex.No.1 Develop an application that uses GUI components, Font and Colours.

Date:

AIM:

To develop an application that uses GUI components, Font and Colours.

PROCEDURE:

Create a new Android Application

1. In Eclipse go to **File->New->Project**
2. Select an **Android Project** from the Android Folder and press **Next.**
3. Fill in the details of your Android application.
    a. **Project Name**: The project name and folder that Eclipse will store the project files
    b. **Build Target:** The version of the Android SDK that will be used when you build your program. Select a platform that is equal to or lower than the target chosen for the AVD.
    c. **Application Name:** This is the name of the application.
    d. **Package Name**: The namespace that all of the source code will reside under.
    e. **Create Activity:** The name for that class stub that is generated by the plugin.
4. The values that are used in this example are:
    a. **Project Name**: GUI components
    b. **Build Target:** 2.3.3
    c. **Application Name:** GUI components
    d. **Package Name**: com. GUI components.example
    e. **Create Activity:** GUI components
5. Click on **Finish.**

Coding the Application

1. Open **AndroidMainfest.xml** which is located in **res->values-> AndroidMainfest.xml.** This file will hold all of the text that our layout will use.
2. Click on the **AndroidMainfest**.xml at the bottom to bring up the raw xml file.

Editing the the java code

1. Open SampleApp.java from the left hand side.
2. Save the files.

Running the Application

1. Click on the green circle with the white arrow.
2. Choose the AVD that we created in a previous step.
3. The android AVD will load and the program will run.

PROGRAMS

File Name : MainActivity.java

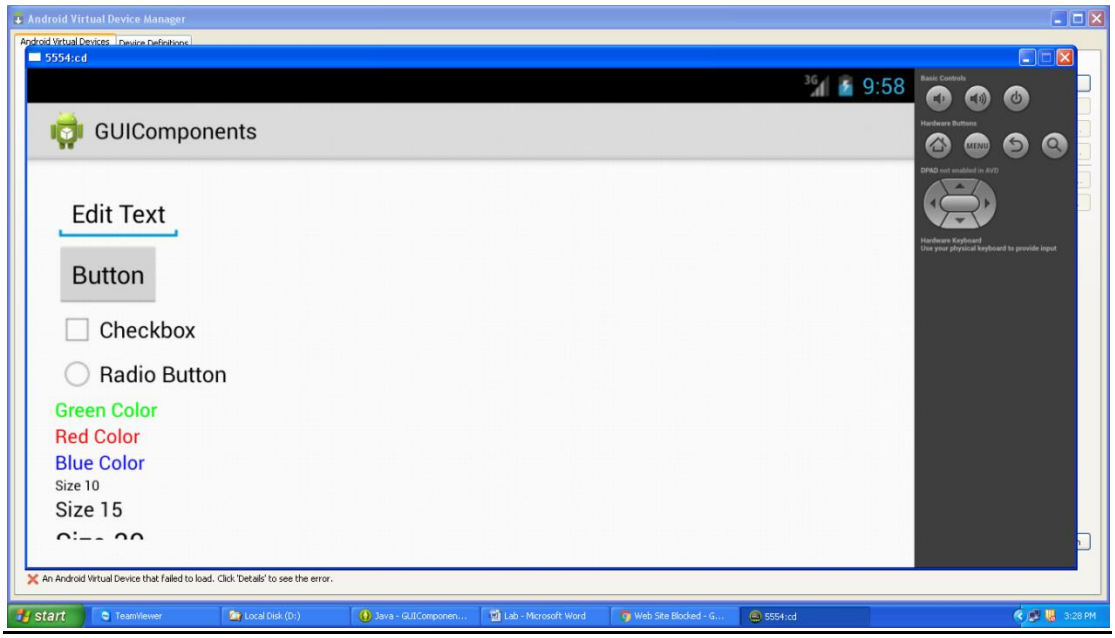
```
package com.lab.guicomponents;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
public class MainActivity extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
```

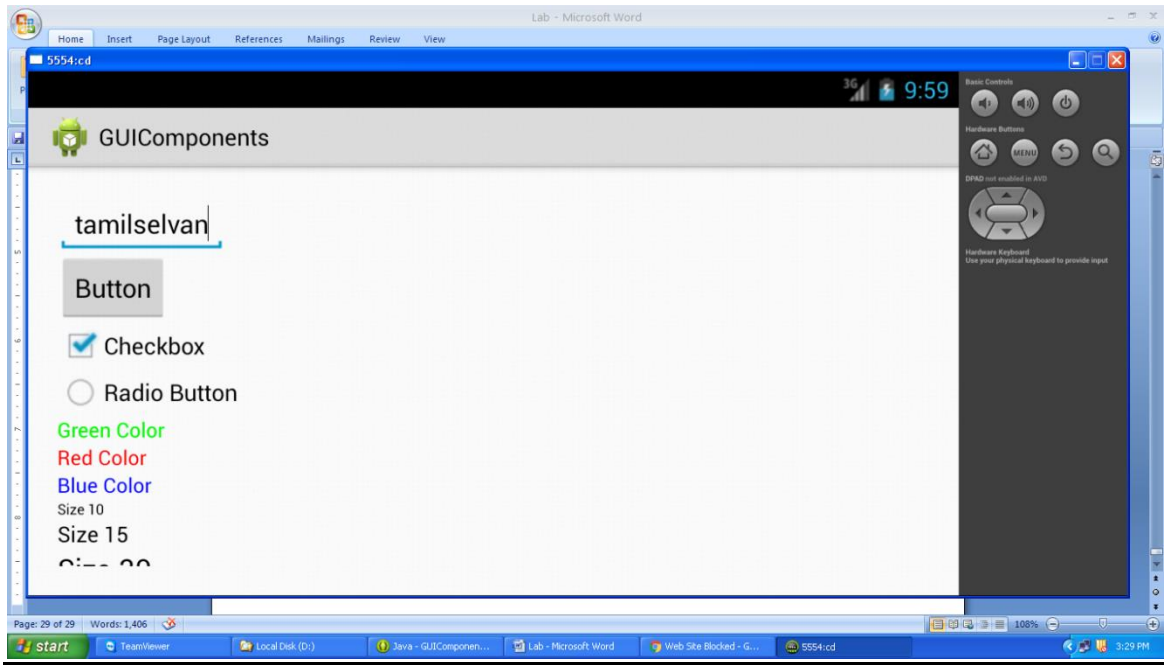
```
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
    }
```

File Name: AndroidMainfest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lab.guicomponents"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.lab.guicomponents.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Output:





RESULT:

Thus the application GUI components, Font and Colours is executed successfully by using Eclipse.

Ex.No.2 Develop an application that uses Layout Managers and event listeners.

Date :

AIM:

To develop an application that uses Layout Managers and event listeners.

PROCEDURE:

Create a new Android Application

1. In Eclipse go to **File->New->Project**
2. Select an **Android Project** from the Android Folder and press **Next.**
3. Fill in the details of your Android application.
    a. **Project Name**: The project name and folder that Eclipse will store the project files
    b. **Build Target:** The version of the Android SDK that will be used when you build your program. Select a platform that is equal to or lower than the target chosen for the AVD.
    c. **Application Name:** This is the name of the application.
    d. **Package Name**: The namespace that all of the source code will reside under.
    e. **Create Activity:** The name for that class stub that is generated by the plugin.
4. The values that are used in this example are:
    a. **Project Name**: LayMEvL
    b. **Build Target:** 2.3.3
    c. **Application Name:** LayMEvL
    d. **Package Name**: com.LayMEvL.example
    e. **Create Activity:** LayMEvL
5. Click on **Finish.**

Coding the Application

1. Open **AndroidMainfest.xml** which is located in **res->values->AndroidMainfest.xml.** This file will hold all of the text that our layout will use.
2. Click on the **AndroidMainfest**.xml at the bottom to bring up the raw xml file.

Editing the the java code

1. Open SampleApp.java from the left hand side.
2. Save the files.

Running the Application

1. Click on the green circle with the white arrow.
2. Choose the AVD that we created in a previous step.
3. The android AVD will load and the program will run.

PROGRAMS

File Name: main Activity.java

```
package com.lab.layout_and_events;

import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.view.Menu;
import android.view.View;
```

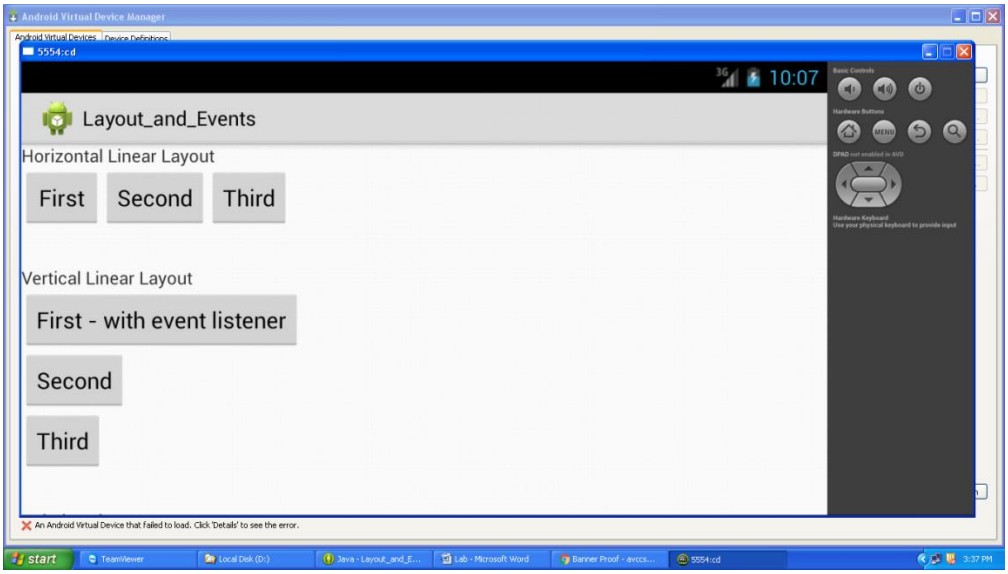
```
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {

                        private Button btnFirst;
                        private Context context;

                        @Override
                        protected void onCreate(Bundle savedInstanceState) {
                            super.onCreate(savedInstanceState);
                            setContentView(R.layout.activity_main);

                            btnFirst = (Button) findViewById(R.id.btnFirst);

                            btnFirst.setOnClickListener(new OnClickListener() {

                                @Override
                                public void onClick(View v) {
                                    // TODO Auto-generated method stub
                                    Toast.makeText(context, "First button
clicked.",Toast.LENGTH_SHORT).show();
                                }
                            });

                        }

                        @Override
                        public boolean onCreateOptionsMenu(Menu menu) {
                            // Inflate the menu; this adds items to the action bar if it is present.
                            getMenuInflater().inflate(R.menu.main, menu);
                            return true;
                        }
}
```

AndroidMainfest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.lab.layout_and_events"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="18" />

   <application
```

```
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.lab.layout_and_events.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

OUTPUT:



RESULT:

Thus the application Layout Managers and event listeners is executed successfully by using Eclipse.

Ex.No.3 Develop a native calculator application.

Date:

AIM:

To develop a native calculator application.

PROCEDURE:

Create a new Android Application

1. In Eclipse go to **File->New->Project**
2. Select an **Android Project** from the Android Folder and press **Next.**
3. Fill in the details of your Android application.
    a. **Project Name**: The project name and folder that Eclipse will store the project files
    b. **Build Target:** The version of the Android SDK that will be used when you build your program. Select a platform that is equal to or lower than the target chosen for the AVD.
    c. **Application Name:** This is the name of the application.
    d. **Package Name**: The namespace that all of the source code will reside under.
    e. **Create Activity:** The name for that class stub that is generated by the plugin.
4. The values that are used in this example are:
    a. **Project Name**: Calc
    b. **Build Target:** 2.3.3
    c. **Application Name:** Calc
    d. **Package Name**: com.Calc.example
    e. **Create Activity:** Calc
5. Click on **Finish.**

Coding the Application

1. Open **AndroidMainfest.xml** which is located in **res->values-> AndroidMainfest.xml.** This file will hold all of the text that our layout will use.
2. Click on the **AndroidMainfest**.xml at the bottom to bring up the raw xml file.

Editing the the java code

1. Open SampleApp.java from the left hand side.
2. Save the files.

Running the Application

1. Click on the green circle with the white arrow.
2. Choose the AVD that we created in a previous step.
3. The android AVD will load and the program will run.

PROGRAMS

File Name: MainActivity.java

```
package com.lab.calculator;


import android.R.string;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
```

```
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends Activity implements OnClickListener {

        private TextView txtDisplay;
        private Button btn1, btn2, btn3, btn4, btn5, btn6, btn7, btn8, btn9, btn0;
        private Button btnPlus, btnMinus, btnDivide, btnMultiply, btnEquals,
                btnDelete;
        private float fNum1 = 0;
        private float fNum2 = 0;
        private float fResult = 0;
        private String strOperator = "";

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);

            txtDisplay = (TextView) findViewById(R.id.txtDisplay);
            btn1 = (Button) findViewById(R.id.btn1);
            btn2 = (Button) findViewById(R.id.btn2);
            btn3 = (Button) findViewById(R.id.btn3);
            btn4 = (Button) findViewById(R.id.btn4);
            btn5 = (Button) findViewById(R.id.btn5);
            btn6 = (Button) findViewById(R.id.btn6);
            btn7 = (Button) findViewById(R.id.btn7);
            btn8 = (Button) findViewById(R.id.btn8);
            btn9 = (Button) findViewById(R.id.btn9);
            btn0 = (Button) findViewById(R.id.btn0);
            btnPlus = (Button) findViewById(R.id.btnPlus);
            btnMinus = (Button) findViewById(R.id.btnMinus);
            btnDivide = (Button) findViewById(R.id.btnDivide);
            btnMultiply = (Button) findViewById(R.id.btnMultiply);
            btnEquals = (Button) findViewById(R.id.btnEquals);
            btnDelete = (Button) findViewById(R.id.btnDelete);

            btn1.setOnClickListener(this);
            btn2.setOnClickListener(this);
            btn3.setOnClickListener(this);
            btn4.setOnClickListener(this);
            btn5.setOnClickListener(this);
            btn6.setOnClickListener(this);
            btn7.setOnClickListener(this);
            btn8.setOnClickListener(this);
            btn9.setOnClickListener(this);
```

```
        btn0.setOnClickListener(this);
        btnPlus.setOnClickListener(this);
        btnMinus.setOnClickListener(this);
        btnDivide.setOnClickListener(this);
        btnMultiply.setOnClickListener(this);
        btnEquals.setOnClickListener(this);
        btnDelete.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {
        String strDisplay = "";

        // TODO Auto-generated method stub
        switch (v.getId()) {
        case R.id.btn1:
        case R.id.btn2:
        case R.id.btn3:
        case R.id.btn4:
        case R.id.btn5:
        case R.id.btn6:
        case R.id.btn7:
        case R.id.btn8:
        case R.id.btn9:
        case R.id.btn0:
                strDisplay = txtDisplay.getText().toString();

                txtDisplay.setText(strDisplay + ((Button) v).getText());
                break;
        case R.id.btnDelete:
                strDisplay = txtDisplay.getText().toString();
                if (!strDisplay.equals("")) {
                        strDisplay = (String) strDisplay.subSequence(0,
                                        strDisplay.length() - 1);
                        txtDisplay.setText(strDisplay);

                        if (strDisplay.equals("")) {
                                fNum1 = 0;
                                fNum2 = 0;
                        }
                }
                break;

        case R.id.btnPlus:
                strOperator = "plus";
```

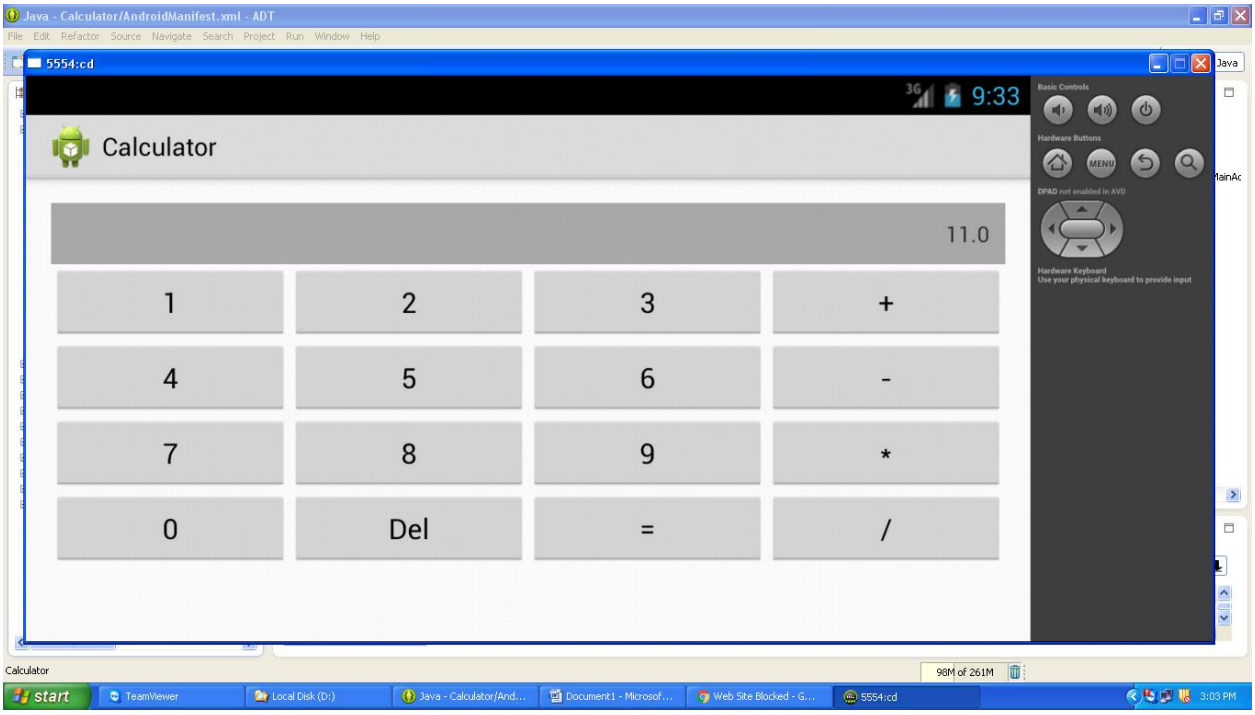
```
                strDisplay = txtDisplay.getText().toString();

                fNum1 = strDisplay.equals("") ? 0 : Integer.parseInt(strDisplay);
                txtDisplay.setText("");
                break;
        case R.id.btnMinus:
                strOperator = "minus";
                strDisplay = txtDisplay.getText().toString();

                fNum1 = strDisplay.equals("") ? 0 : Integer.parseInt(strDisplay);
                txtDisplay.setText("");
                break;
        case R.id.btnMultiply:
                strOperator = "multiply";
                strDisplay = txtDisplay.getText().toString();

                fNum1 = strDisplay.equals("") ? 0 : Integer.parseInt(strDisplay);
                txtDisplay.setText("");
                break;
        case R.id.btnDivide:
                strOperator = "divide";

                strDisplay = txtDisplay.getText().toString();

                fNum1 = strDisplay.equals("") ? 0 : Integer.parseInt(strDisplay);
                txtDisplay.setText("");
                break;

        case R.id.btnEquals:
                strDisplay = txtDisplay.getText().toString();

                fNum2 = strDisplay.equals("") ? 0 : Integer.parseInt(strDisplay);

                if(strOperator.equals("plus"))
                        fResult = fNum1 + fNum2;
                else if(strOperator.equals("minus"))
                        fResult = fNum1 - fNum2;
                else if(strOperator.equals("multiply"))
                        fResult = fNum1 * fNum2;
                else if(strOperator.equals("divide"))
                        fResult = fNum1 / fNum2;


                txtDisplay.setText(Float.toString(fResult));

                break;
```

```
                    }
                }
}
```

File Name: AndroidMaindfest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lab.calculator"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.lab.calculator.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

OUTPUT:



RESULT:

Thus the native calculator application is executed successfully by using Eclipse.

Ex.No.4 Write an application that draws basic graphical primitives on the screen.

Date:

AIM:

To develop an application that draws basic graphical primitives on the screen.

PROCEDURE:

Create a new Android Application

1. In Eclipse go to **File->New->Project**
2. Select an **Android Project** from the Android Folder and press **Next.**
3. Fill in the details of your Android application.
    a. **Project Name**: The project name and folder that Eclipse will store the project files
    b. **Build Target:** The version of the Android SDK that will be used when you build your program. Select a platform that is equal to or lower than the target chosen for the AVD.
    c. **Application Name:** This is the name of the application.
    d. **Package Name**: The namespace that all of the source code will reside under.
    e. **Create Activity:** The name for that class stub that is generated by the plugin.
4. The values that are used in this example are:
    a. **Project Name**: GraphPrim
    b. **Build Target:** 2.3.3
    c. **Application Name:** GraphPrim
    d. **Package Name**: com. GraphPrim.example
    e. **Create Activity:** GraphPrim
5. Click on **Finish.**

Coding the Application

1. Open **AndroidMainfest.xml** which is located in **res->values->AndroidMainfest.xml.** This file will hold all of the text that our layout will use.
2. Click on the **AndroidMainfest**.xml at the bottom to bring up the raw xml file.

Editing the the java code

1. Open SampleApp.java from the left hand side.
2. Save the files.

Running the Application

1. Click on the green circle with the white arrow.
2. Choose the AVD that we created in a previous step.
3. The android AVD will load and the program will run.

PROGRAMS

File Name: MainActivity.java

```
package comlab.draws_basic_graphics;

import android.app.Activity;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
```

```
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;

public class MainActivity extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      final ImageView img = (ImageView) findViewById(R.id.img);

      Button btnCircle = (Button) findViewById(R.id.btnCircle);
      btnCircle.setOnClickListener(new OnClickListener() {

                                    @Override
                                    public void onClick(View v) {
                                          // TODO Auto-generated method stub
                                          final Bitmap bmp = Bitmap.createBitmap(100, 100,
Bitmap.Config.ARGB_8888);
                                          final Paint paint = new Paint();
                                          paint.setAntiAlias(true);
                                          paint.setColor(Color.YELLOW);
                                          Canvas c = new Canvas(bmp);
                                          c.drawCircle(10,50,35, paint);
                                          img.setImageBitmap(bmp);
                                    }
                              });

      Button btnRect = (Button) findViewById(R.id.btnRect);
      btnRect.setOnClickListener(new OnClickListener() {

                                    @Override
                                    public void onClick(View v) {
                                          // TODO Auto-generated method stub
                                          final Bitmap bmp = Bitmap.createBitmap(100, 100,
Bitmap.Config.ARGB_8888);
                                          final Paint paint = new Paint();
                                          paint.setAntiAlias(true);
                                          paint.setColor(Color.GREEN);
```

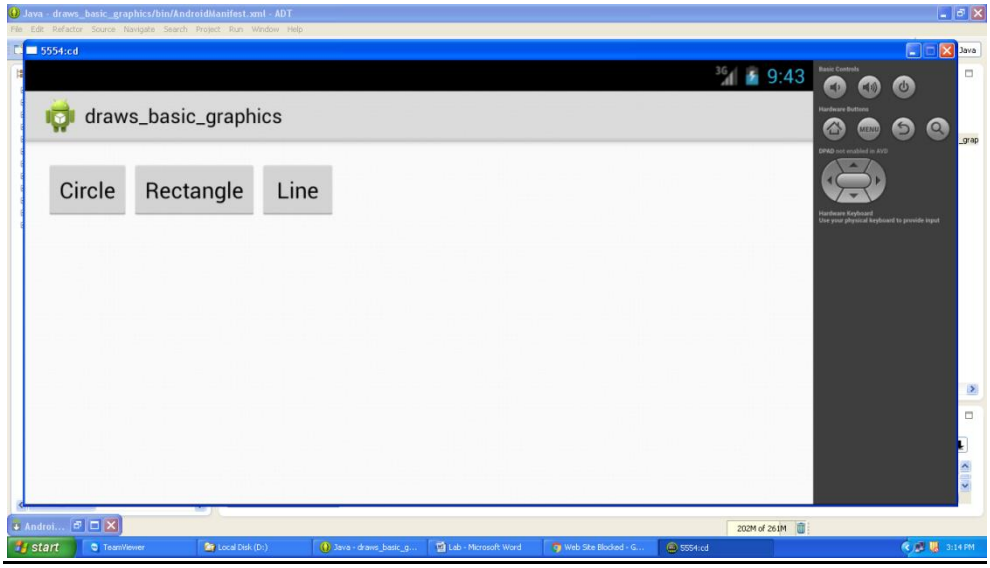
```
                Canvas c = new Canvas(bmp);
                //c.drawCircle(60,50,25, paint);
                c.drawRect(0, 0, 100, 100, paint);
                img.setImageBitmap(bmp);
            }
        });

    Button btnLine = (Button) findViewById(R.id.btnLine);
    btnLine.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                final Bitmap bmp = Bitmap.createBitmap(100, 100,
Bitmap.Config.ARGB_8888);
                final Paint paint = new Paint();
                paint.setAntiAlias(true);
                paint.setColor(Color.RED);

                Canvas c = new Canvas(bmp);
                c.drawLine(10, 10, 100, 100, paint);
                img.setImageBitmap(bmp);
            }
        });
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
  }

  public class CustomView extends View {

    private Paint paint;
    private Canvas _canvas;

    public CustomView(Context context) {
      super(context);

      // create the Paint and set its color
      paint = new Paint();
      paint.setColor(Color.GRAY);
    }
```

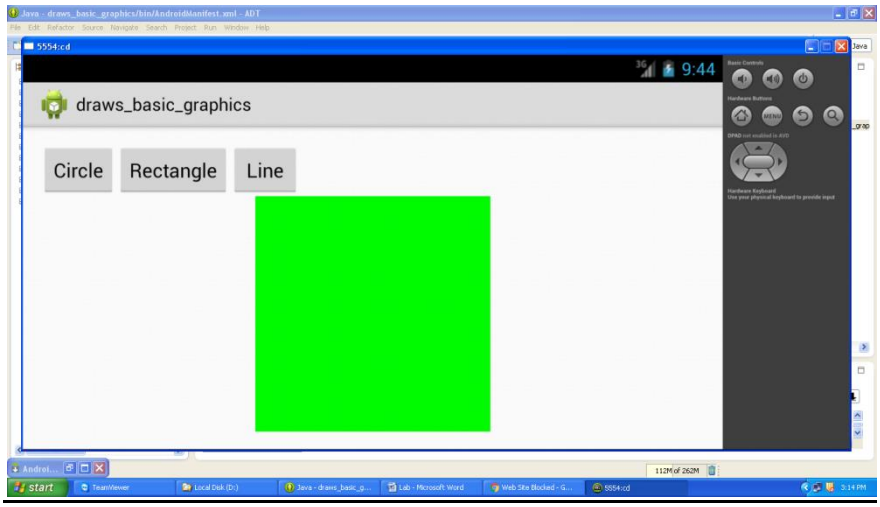
```
        @Override
        protected void onDraw(Canvas canvas) {
           //canvas.drawColor(Color.BLUE);
                                canvas.drawCircle(200, 200, 100, paint);
        }

        private void drawCircle()
        {

                                invalidate();
        }

    }

}
```

File Name : AndroidMainfest.Xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="comlab.draws_basic_graphics"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="comlab.draws_basic_graphics.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

OUTPUT:





RESULT:

Thus the application that draws basic graphical primitives on the screen.is executed successfully by using Eclipse.

Ex.No.5 Develop an application that makes use of database.

Date :

AIM:

To develop an application that makes use of database..

PROCEDURE:

Create a new Android Application

1. In Eclipse go to **File->New->Project**
2. Select an **Android Project** from the Android Folder and press **Next.**
3. Fill in the details of your Android application.
    a. **Project Name**: The project name and folder that Eclipse will store the project files
    b. **Build Target:** The version of the Android SDK that will be used when you build your program. Select a platform that is equal to or lower than the target chosen for the AVD.
    c. **Application Name:** This is the name of the application.
    d. **Package Name**: The namespace that all of the source code will reside under.
    e. **Create Activity:** The name for that class stub that is generated by the plugin.
4. The values that are used in this example are:
    a. **Project Name**: DBase
    b. **Build Target:** 2.3.3
    c. **Application Name:** DBase
    d. **Package Name**: com.DBase.example
    e. **Create Activity:** DBase
5. Click on **Finish.**

Coding the Application

1. Open **AndroidMainfest.xml** which is located in **res->values-> AndroidMainfest.xml.** This file will hold all of the text that our layout will use.
2. Click on the **AndroidMainfest**.xml at the bottom to bring up the raw xml file.

Editing the the java code

1. Open SampleApp.java from the left hand side.
2. Save the files.

Running the Application

1. Click on the green circle with the white arrow.
2. Choose the AVD that we created in a previous step.
3. The android AVD will load and the program will run.

PROGRAMS

File Name : Main Activity.java

```
package com.lap.database;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
```

```
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.Toast;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;

public class MainActivity extends Activity {

                        private EditText txtStudentName, txtDepartment;
                        private Button btnAdd;
                        private ListView listView1;

                        @Override
                        protected void onCreate(Bundle savedInstanceState) {
                            super.onCreate(savedInstanceState);
                            setContentView(R.layout.activity_main);

                            listView1 = (ListView) findViewById(R.id.listView1);
                            btnAdd = (Button) findViewById(R.id.btnAdd);
                            txtStudentName = (EditText) findViewById(R.id.txtStudentName);
                            txtDepartment = (EditText) findViewById(R.id.txtDepartment);

                            btnAdd.setOnClickListener(new OnClickListener() {

                                    @Override
                                    public void onClick(View arg0) {
                                            // TODO Auto-generated method stub
                                            insertRecord();
                                    }
                            });
                        }

                        private void insertRecord() {
                            SQLiteDatabase db = openOrCreateDatabase("dbTest",
MODE_PRIVATE, null);

                                db.execSQL("CREATE TABLE IF NOT EXISTS Students(StudentName
VARCHAR,Department VARCHAR);");

                                String strDept = txtDepartment.getText().toString();
                                String strName = txtStudentName.getText().toString();
```

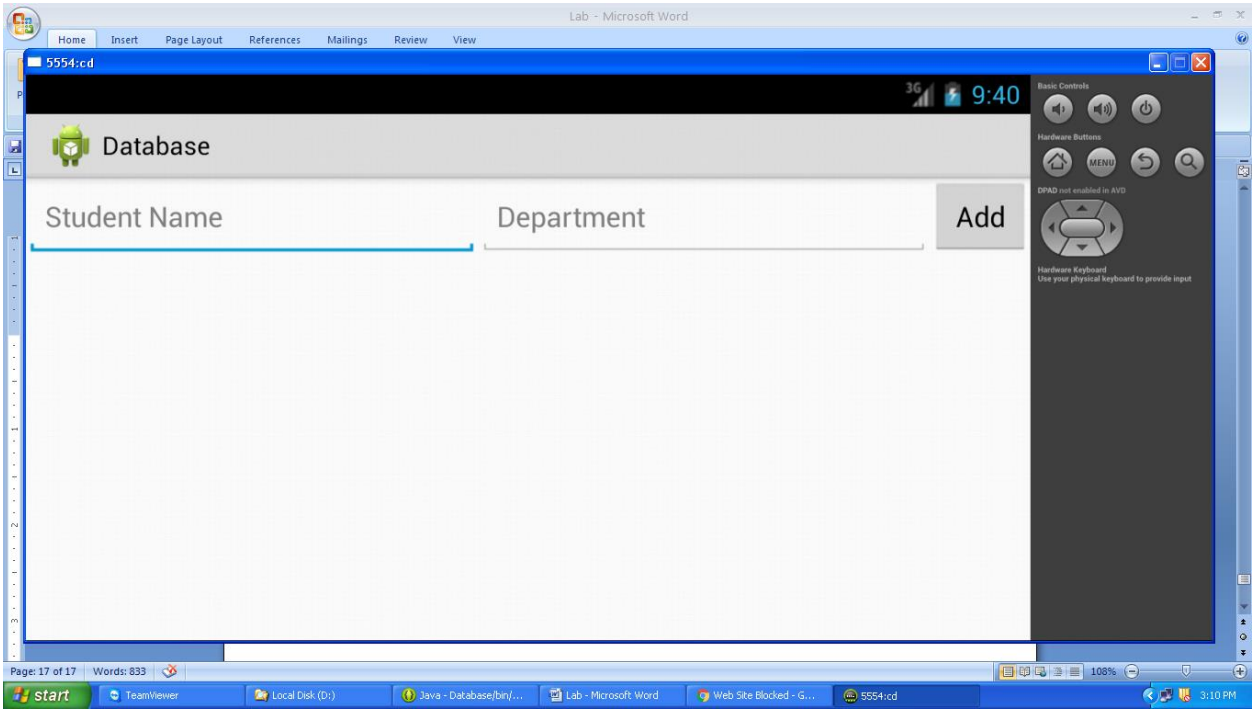
```
                        db.execSQL("INSERT INTO Students VALUES('" + strName + "','" +
strDept
                                + "');");

                        Cursor resultSet = db
                                .rawQuery("Select rowid _id,* from Students", null);

                        int nRecordCount = resultSet.getCount();

                        if (nRecordCount > 0) {
                                String[] strColumns = new String[] { "StudentName",
"Department" };
                                int[] nListCol = new int[] { R.id.text1, R.id.text2 };

                                SimpleCursorAdapter mAdapter = new SimpleCursorAdapter(this,
                                        R.layout.listitem, resultSet, strColumns, nListCol);

                                listView1.setAdapter(mAdapter);
                        }
                }
}
```

File Name : AndroidMainfest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.lap.database"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="18" />

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
         android:name="com.lap.database.MainActivity"
         android:label="@string/app_name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>
```

</manifest>

OUTPUT:



RESULT:

Thus the application that makes use of database is executed successfully by using Eclipse.

Ex.No.6 Develop an application that makes use of RSS Feed

Date:

AIM:

To develop an application that makes use of RSS Feed.

PROCEDURE:

Create a new Android Application

1. In Eclipse go to **File->New->Project**
2. Select an **Android Project** from the Android Folder and press **Next.**
3. Fill in the details of your Android application.
    a. **Project Name**: The project name and folder that Eclipse will store the project files
    b. **Build Target:** The version of the Android SDK that will be used when you build your program. Select a platform that is equal to or lower than the target chosen for the AVD.
    c. **Application Name:** This is the name of the application.
    d. **Package Name**: The namespace that all of the source code will reside under.
    e. **Create Activity:** The name for that class stub that is generated by the plugin.
4. The values that are used in this example are:
    a. **Project Name**: RSSFeed
    b. **Build Target:** 2.3.3
    c. **Application Name:** RSSFeed
    d. **Package Name**: com. RSSFeed.example
    e. **Create Activity:** RSSFeed
5. Click on **Finish.**

Coding the Application

1. Open **AndroidMainfest.xml** which is located in **res->values-> AndroidMainfest.xml.** This file will hold all of the text that our layout will use.
2. Click on the **AndroidMainfest**.xml at the bottom to bring up the raw xml file.

Editing the the java code

1. Open SampleApp.java from the left hand side.
2. Save the files.

Running the Application

1. Click on the green circle with the white arrow.
2. Choose the AVD that we created in a previous step.
3. The android AVD will load and the program will run.

PROGRAMS

FileName :MainActivity.java

```
package com.lab.rss_feed;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
```

```
import java.net.URL;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserFactory;

import com.lab.rss_feed.R.string;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.webkit.WebView;
import android.widget.Toast;

public class MainActivity extends Activity {

                        private Context context;
                        private WebView webview;

                        @Override
                        protected void onCreate(Bundle savedInstanceState) {
                            // TODO Auto-generated method stub
                            super.onCreate(savedInstanceState);

                            context = this;

                            webview = (WebView) findViewById(R.id.webview);

                            DownloadRSSfeed feed = new DownloadRSSfeed();
                            feed.execute();
                        }

                        private InputStream readXML() {
                            String strURL = "http://rss.cnn.com/rss/edition.rss";
                            // String strURL = "http://www.google.com";

                            try {
                                    URL url = new URL(strURL);
                                    HttpURLConnection conn = (HttpURLConnection)
url.openConnection();

                                    Log.e("err", "1");

                                    conn.setAllowUserInteraction(false);
```

```
                conn.setInstanceFollowRedirects(true);
                conn.setRequestMethod("GET");

                /*
                 * conn.setReadTimeout(10000); conn.setConnectTimeout(15000);
                 */

                Log.e("err", "2");

                conn.connect();

                int resCode = -1;
                resCode = conn.getResponseCode();

                InputStream is = null;

                if (resCode == HttpURLConnection.HTTP_OK) {
                        Log.e("err", "2.1");
                        is = conn.getInputStream();
                }

                Log.e("err", "3");

                BufferedReader reader = new BufferedReader(new
InputStreamReader(
                                is, "UTF-8"));
                String webPage = "", data = "";

                while ((data = reader.readLine()) != null) {
                        webPage += data + "\n";
                        Log.e("data", " " + data);
                }

                Log.e("err", "4");

                /* parseXML(myparser); */

                Log.e("err", "7");

                //is.close();
                return is;
        } catch (Exception ex) {
                Log.e("err", "err : " + ex.getMessage() +
ex.getLocalizedMessage());
        }
        return null;
```

```
                }

                private void parseXML(InputStream is) {
                        int event;
                        String text = null;
                        String title = "";
                        String link = "";
                        String description = "";

                        try {
                                XmlPullParserFactory xmlFactoryObject = XmlPullParserFactory
                                                .newInstance();
                                XmlPullParser myparser = xmlFactoryObject.newPullParser();

                                Log.e("err", "5");


                myparser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES,
false);
                                myparser.setInput(is, null);

                                Log.e("err", "6");

                                event = myparser.getEventType();
                                Log.e("err", "6.00; event=" + event);

                                while (event != XmlPullParser.END_DOCUMENT) {
                                        String name = myparser.getName();

                                        switch (event) {
                                        case XmlPullParser.START_TAG:
                                                Log.e("err", "6.11");
                                                break;

                                        case XmlPullParser.TEXT:
                                                text = myparser.getText();
                                                Log.e("err", "6.22");
                                                break;

                                        case XmlPullParser.END_TAG:
                                                Log.e("err", "6.33;" + name);
                                                if (name.equals("title")) {
                                                        title = text;
                                                }

                                                else if (name.equals("link")) {
```

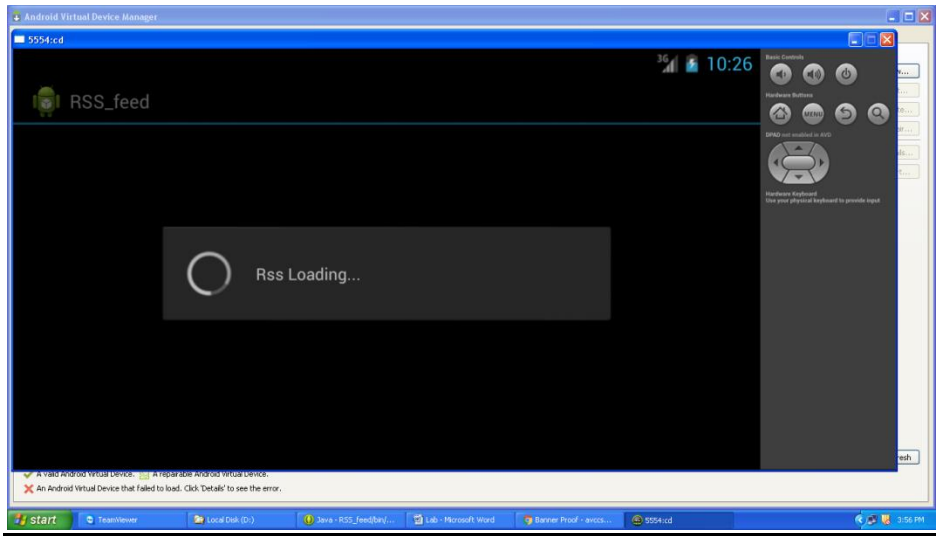
```
                                        link = text;
                                    }

                                    else if (name.equals("description")) {
                                        description = text;
                                    }

                                    else {
                                    }

                                    break;
                                }

                                event = myparser.next();
                            }

                            // parsingComplete = false;
                            Toast.makeText(context, title + " ; " + link,
Toast.LENGTH_LONG)
                                    .show();
                        }

                        catch (Exception e) {
                            Log.e("err", "6.1" + e.getMessage());
                        }
                    }

                    private class DownloadRSSfeed extends AsyncTask<String, Void,
InputStream> {
                        private ProgressDialog Dialog;
                        String response = "";

                        @Override
                        protected void onPreExecute() {
                            Dialog = new ProgressDialog(MainActivity.this);
                            Dialog.setMessage("Rss Loading...");
                            Dialog.show();
                        }

                        @Override
                        protected InputStream doInBackground(String... urls) {
                            return readXML();

                            //return response;
                        }
                        @Override
```

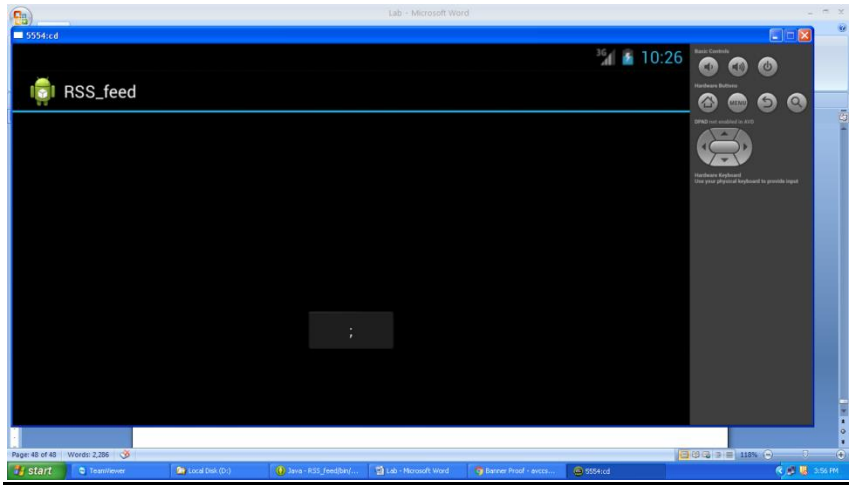
```
                protected void onPostExecute(InputStream is) {
                        parseXML(is);
                        Dialog.dismiss();
                }       } }
```

File Name: AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.lab.rss_feed"
   android:versionCode="1"
   android:versionName="1.0" >

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

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="21" />

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
         android:name=".MainActivity"
         android:label="@string/app_name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>

</manifest>
```

OUTPUT:





RESULT:

Thus the application that makes use of RSS Feed is executed successfully by using Eclipse.

Ex.No.7 Implement an application that implements Multi threading

Date:

AIM:

To develop an application that implements Multi threading.

PROCEDURE:

Create a new Android Application

1. In Eclipse go to **File->New->Project**
2. Select an **Android Project** from the Android Folder and press **Next.**
3. Fill in the details of your Android application.
    a. **Project Name**: The project name and folder that Eclipse will store the project files
    b. **Build Target:** The version of the Android SDK that will be used when you build your program. Select a platform that is equal to or lower than the target chosen for the AVD.
    c. **Application Name:** This is the name of the application.
    d. **Package Name**: The namespace that all of the source code will reside under.
    e. **Create Activity:** The name for that class stub that is generated by the plugin.
4. The values that are used in this example are:
    a. **Project Name**: Mthred
    b. **Build Target:** 2.3.3
    c. **Application Name:** Mthred
    d. **Package Name**: com. Mthred.example
    e. **Create Activity:** Mthred
5. Click on **Finish.**

Coding the Application

1. Open **AndroidMainfest.xml** which is located in **res->values->AndroidMainfest.xml.** This file will hold all of the text that our layout will use.
2. Click on the **AndroidMainfest**.xml at the bottom to bring up the raw xml file.

Editing the the java code

1. Open SampleApp.java from the left hand side.
2. Save the files.

Running the Application

1. Click on the green circle with the white arrow.
2. Choose the AVD that we created in a previous step.
3. The android AVD will load and the program will run.

PROGRAMS

FileName :MainActivity.java

```
package com.lab.multithreading;

import android.app.Activity;
import android.opengl.Visibility;
import android.os.Bundle;
```

```
import android.os.Handler;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.ImageView;
import android.widget.TextView;

public class MainActivity extends Activity {

		final Handler mHandler = new Handler(); // A handler allows you to post
                                                                                                                                                //
runnables to execute on a
                                                                                                                                                //
specific thread. Behind the
                                                                                                                                                //
scenes, runOnUi Thread queues
                                                                                                                                                //
your Runnable up with Android's
                                                                                                                                                // Ui
Handler so your runnable can
                                                                                                                                                //
execute safely on the UI thread.
		private TextView txtNum1, txtNum2;

		@Override
		protected void onCreate(Bundle savedInstanceState) {
			super.onCreate(savedInstanceState);
			setContentView(R.layout.activity_main);

			txtNum1 = (TextView) findViewById(R.id.txtNum1);
			txtNum2 = (TextView) findViewById(R.id.txtNum2);

			Incr();
			Decr();
		}

		int nPlus,nMinus;
		Runnable r = new Runnable() {
			public void run() {
				txtNum1.setText(" " + nPlus);
				txtNum2.setText(" " + nMinus);
			}
		};

		protected void Incr() {
```

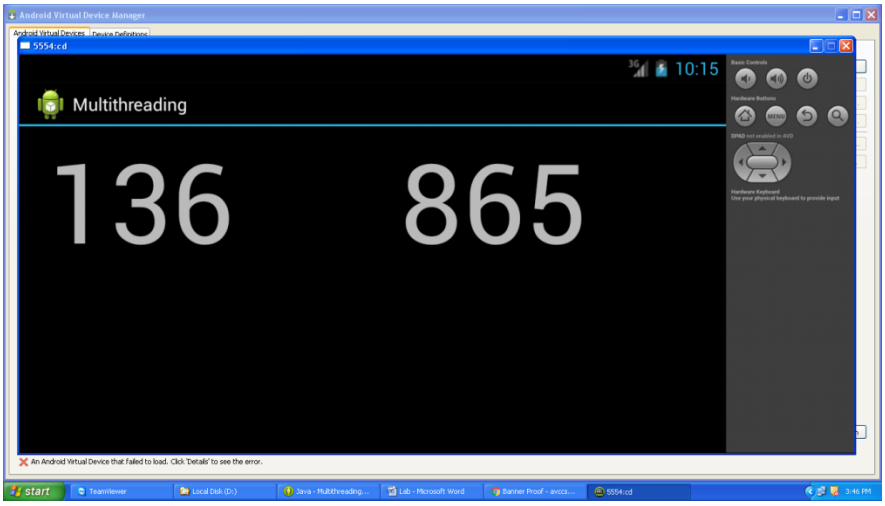
```
                Thread incrThread = new Thread() {
                    public void run() {
                        try {
                            for (int i = 0; i < 1000; i++) {
                                nPlus = i;
                                mHandler.post(r);
                                sleep(100);
                            }
                        } catch (Exception e) {

                        }
                    }
                };
                incrThread.start();
            }

            protected void Decr() {
                Thread decrThread = new Thread() {
                    public void run() {
                        try {
                            for (int i = 1000; i > 0; i--) {
                                nMinus = i;
                                mHandler.post(r);
                                sleep(100);
                            }

                        } catch (Exception e) {

                        }
                    }
                };
                decrThread.start();
            }
}
```

Filename :AndroidMainfest.xml

```
?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.lab.multithreading"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="21" />

   <application
```

```
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

OUTPUT:



RESULT:

Thus the application that implements Multi threading is executed successfully by using Eclipse.

Ex.No.8 Develop a native application that uses GPS location information.

Date :

AIM:

To develop a native application that uses GPS location information.

PROCEDURE:

Create a new Android Application

1. In Eclipse go to **File->New->Project**
2. Select an **Android Project** from the Android Folder and press **Next.**
3. Fill in the details of your Android application.
    a. **Project Name**: The project name and folder that Eclipse will store the project files
    b. **Build Target:** The version of the Android SDK that will be used when you build your program. Select a platform that is equal to or lower than the target chosen for the AVD.
    c. **Application Name:** This is the name of the application.
    d. **Package Name**: The namespace that all of the source code will reside under.
    e. **Create Activity:** The name for that class stub that is generated by the plugin.
4. The values that are used in this example are:
    a. **Project Name**: GPSl
    b. **Build Target:** 2.3.3
    c. **Application Name:** GPSl
    d. **Package Name**: com. GPSl.example
    e. **Create Activity:** GPSl
5. Click on **Finish.**

Coding the Application

1. Open **AndroidMainfest.xml** which is located in **res->values-> AndroidMainfest.xml.** This file will hold all of the text that our layout will use.
2. Click on the **AndroidMainfest**.xml at the bottom to bring up the raw xml file.

Editing the the java code

1. Open SampleApp.java from the left hand side.
2. Save the files.

Running the Application

1. Click on the green circle with the white arrow.
2. Choose the AVD that we created in a previous step.
3. The android AVD will load and the program will run.

PROGRAMS

FileName :MainActivity.java

```
package com.lab.gps_location;

import android.app.Activity;
import android.content.Context;
import android.location.Location;
import android.location.LocationListener;
```

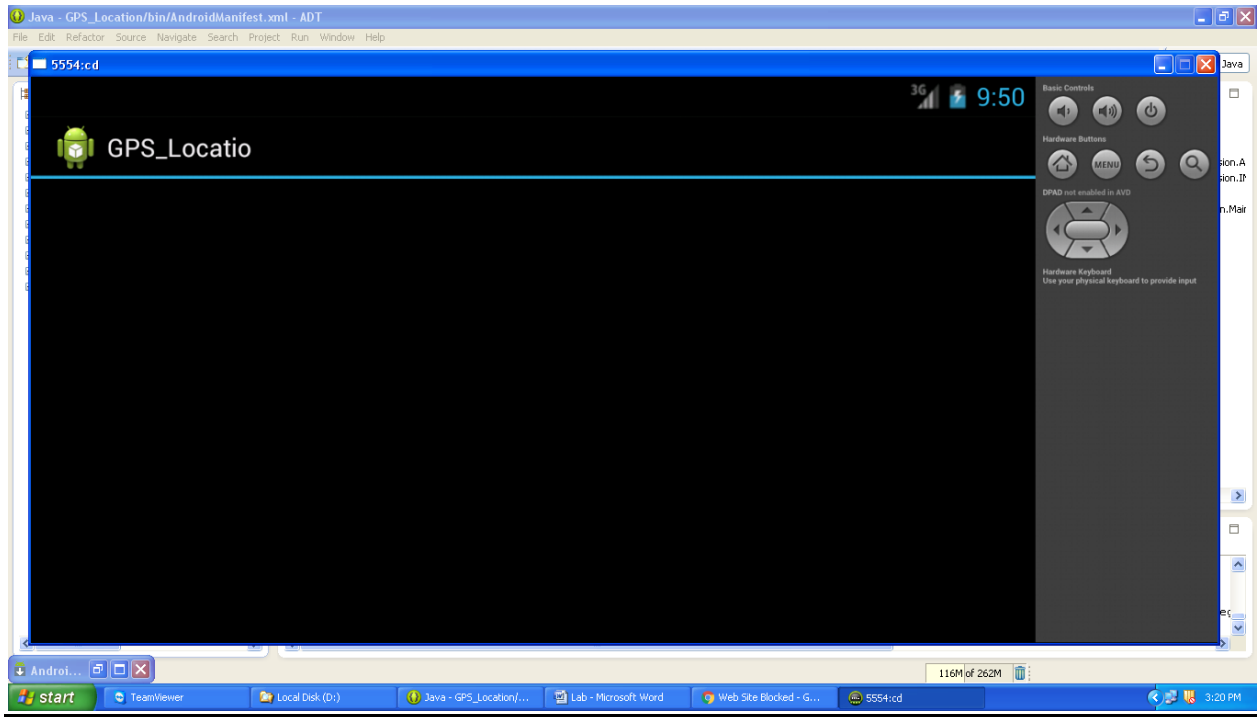
```
import android.location.LocationManager;
import android.os.Bundle;
import android.widget.Toast;

public class MainActivity extends Activity implements LocationListener {

                        private Context context;

                        @Override
                        protected void onCreate(Bundle savedInstanceState) {
                            // TODO Auto-generated method stub
                            super.onCreate(savedInstanceState);
                            setContentView(R.layout.activity_main);

                            context = this;

                            LocationManager locationManager;

                            locationManager = (LocationManager)
context.getSystemService(LOCATION_SERVICE);


                        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
0,                      0, this);

                        }

                        @Override
                        public void onLocationChanged(Location location) {
                            // TODO Auto-generated method stub
                            String str = "Latitude: " + location.getLatitude() + "Longitude: " +
location.getLongitude();

                            Toast.makeText(getBaseContext(), str, Toast.LENGTH_LONG).show();
                        }

                        @Override
                        public void onStatusChanged(String provider, int status, Bundle extras) {
                            // TODO Auto-generated method stub

                        }

                        @Override
                        public void onProviderEnabled(String provider) {
                            // TODO Auto-generated method stub
```

```
                    }

                    @Override
                    public void onProviderDisabled(String provider) {
                        // TODO Auto-generated method stub

                    }
}
```

File Name: AndroidMainfest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lab.gps_location"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />

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

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.lab.gps_location.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

OUTPUT:



RESULT:

Thus a native application that uses GPS location information is executed successfully by using Eclipse.

Ex.No.9 Implement an application that writes data to the SD card.

Date:

AIM:

To develop an application that writes data to the SD card..

PROCEDURE:

Create a new Android Application

1. In Eclipse go to **File->New->Project**
2. Select an **Android Project** from the Android Folder and press **Next.**
3. Fill in the details of your Android application.
    a. **Project Name**: The project name and folder that Eclipse will store the project files
    b. **Build Target:** The version of the Android SDK that will be used when you build your program. Select a platform that is equal to or lower than the target chosen for the AVD.
    c. **Application Name:** This is the name of the application.
    d. **Package Name**: The namespace that all of the source code will reside under.
    e. **Create Activity:** The name for that class stub that is generated by the plugin.
4. The values that are used in this example are:
    a. **Project Name**: SDcard
    b. **Build Target:** 2.3.3
    c. **Application Name:** SDcard
    d. **Package Name**: com. SDcard.example
    e. **Create Activity:** SDcard
5. Click on **Finish.**

Coding the Application

1. Open **AndroidMainfest.xml** which is located in **res->values->AndroidMainfest.xml.** This file will hold all of the text that our layout will use.
2. Click on the **AndroidMainfest**.xml at the bottom to bring up the raw xml file.

Editing the the java code

1. Open SampleApp.java from the left hand side.
2. Save the files.

Running the Application

1. Click on the green circle with the white arrow.
2. Choose the AVD that we created in a previous step.
3. The android AVD will load and the program will run.

PROGRAMS

FileName :MainActivity.java

```
package com.lab.sdcard;
import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;

import android.app.Activity;
import android.os.Bundle;
import android.os.Environment;
```

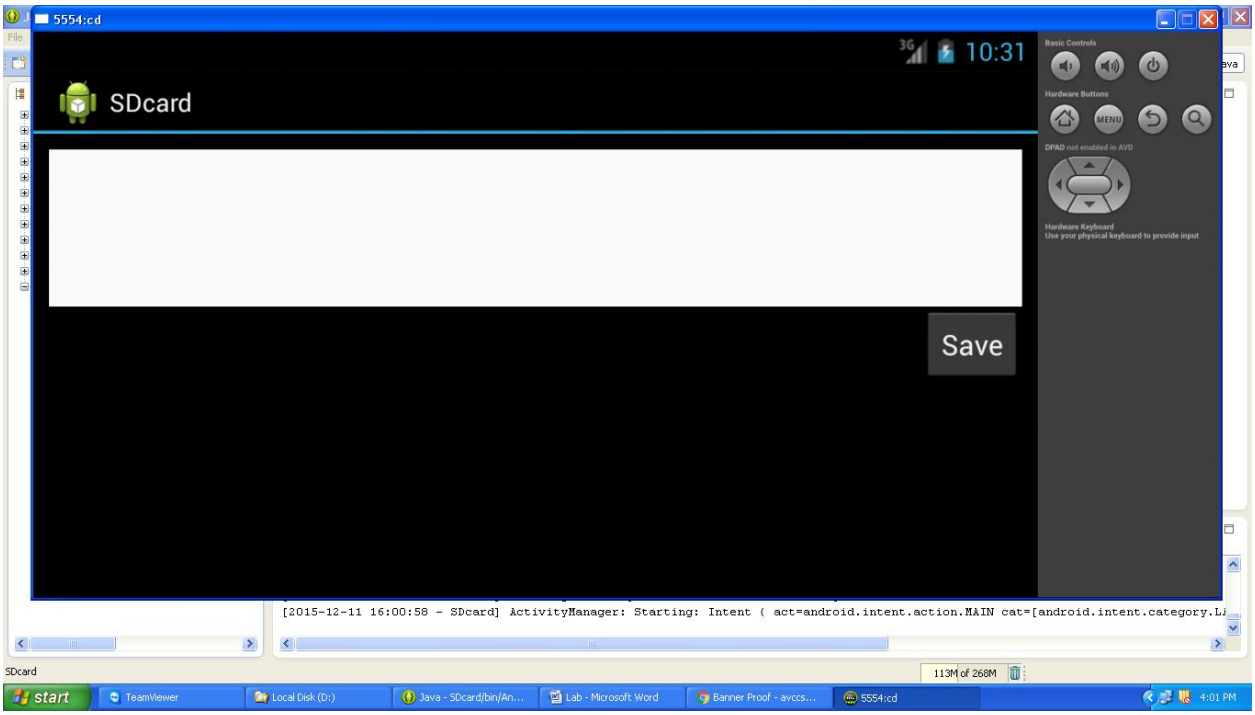
```
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends Activity {


                @Override
                protected void onCreate(Bundle savedInstanceState) {
                    // TODO Auto-generated method stub
                    super.onCreate(savedInstanceState);
                    setContentView(R.layout.activity_main);

                    final EditText txtContent = (EditText) findViewById(R.id.txtContent);

                    Button btnSave = (Button) findViewById(R.id.btnSave);
                    btnSave.setOnClickListener(new OnClickListener() {

                        @Override
                        public void onClick(View v) {
                            // TODO Auto-generated method stub
                            try {
                                String strFileName = "test.txt";

                                File SDCardRoot =
Environment.getExternalStorageDirectory();

                                File file = new File(SDCardRoot + "/" +
strFileName);

                                FileOutputStream fileOutput = new
FileOutputStream(file);

                                OutputStreamWriter outputStreamWriter = new
OutputStreamWriter(
                                        fileOutput);

                outputStreamWriter.append(txtContent.getText().toString());
                                outputStreamWriter.close();

                                fileOutput.close();

                            } catch (Exception e) {
                                Log.e("err", e.getMessage());
```

```
                                        }
                                    }
                                });

                            }

}
```

File Name : MainActivity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lab.sdcard"
    android:versionCode="1"
    android:versionName="1.0" >

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

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="14" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

OUTPUT:



RESULT:

Thus an application that writes data to the SD card is executed successfully by using Eclipse.

Ex.No.10 Implement an application that creates an alert upon receiving a message.

Date:

AIM:

To develop an application that creates an alert upon receiving a message..

PROCEDURE:

Create a new Android Application

1. In Eclipse go to **File->New->Project**
2. Select an **Android Project** from the Android Folder and press **Next.**
3. Fill in the details of your Android application.
    a. **Project Name**: The project name and folder that Eclipse will store the project files
    b. **Build Target:** The version of the Android SDK that will be used when you build your program. Select a platform that is equal to or lower than the target chosen for the AVD.
    c. **Application Name:** This is the name of the application.
    d. **Package Name**: The namespace that all of the source code will reside under.
    e. **Create Activity:** The name for that class stub that is generated by the plugin.
4. The values that are used in this example are:
    a. **Project Name**: RevMsg
    b. **Build Target:** 2.3.3
    c. **Application Name:** RevMsg
    d. **Package Name**: com.RevMsg.example
    e. **Create Activity:** RevMsg
5. Click on **Finish.**

Coding the Application

1. Open **AndroidMainfest.xml** which is located in **res->values-> AndroidMainfest.xml.** This file will hold all of the text that our layout will use.
2. Click on the **AndroidMainfest**.xml at the bottom to bring up the raw xml file.

Editing the the java code

1. Open SampleApp.java from the left hand side.
2. Save the files.

Running the Application

1. Click on the green circle with the white arrow.
2. Choose the AVD that we created in a previous step.
3. The android AVD will load and the program will run.

PROGRAMS

FileName :MainActivity.java

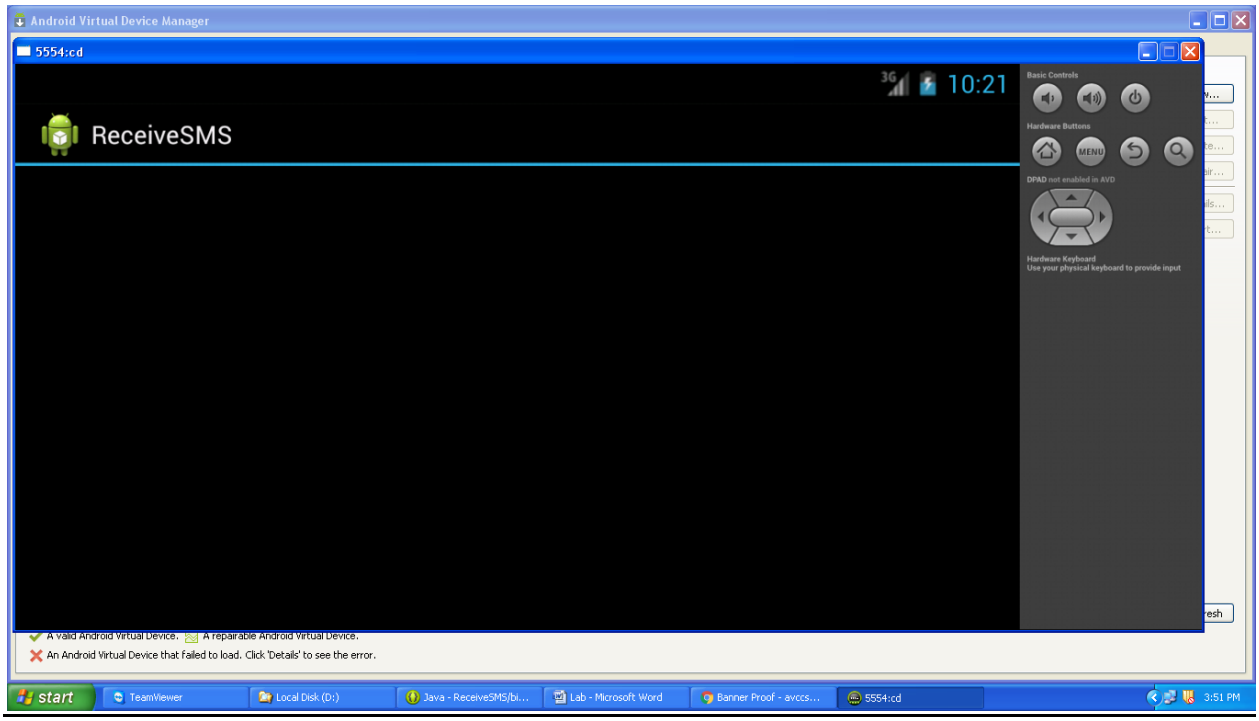
```
package com.lab.receivesms;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {
        @Override
        protected void onCreate(Bundle savedInstanceState) {
```

```
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


    }
}
```

FileName: AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.lab.receivesms"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="21" />

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
         android:name=".MainActivity"
         android:label="@string/app_name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>

</manifest>
```

OUTPUT:



RESULT:

Thus the application that creates an alert upon receiving a message is executed successfully by using Eclipse.

Ex.No.11 Write a mobile application that creates alarm clock

Date:

AIM:

To Write a mobile application that creates alarm clock.

PROCEDURE:

Create a new Android Application

1. In Eclipse go to **File->New->Project**
2. Select an **Android Project** from the Android Folder and press **Next.**
3. Fill in the details of your Android application.
    a. **Project Name**: The project name and folder that Eclipse will store the project files
    b. **Build Target:** The version of the Android SDK that will be used when you build your program. Select a platform that is equal to or lower than the target chosen for the AVD.
    c. **Application Name:** This is the name of the application.
    d. **Package Name**: The namespace that all of the source code will reside under.
    e. **Create Activity:** The name for that class stub that is generated by the plugin.
4. The values that are used in this example are:
    a. **Project Name**: Alarm
    b. **Build Target:** 2.3.3
    c. **Application Name:** Alarm
    d. **Package Name**: com. Alarm.example
    e. **Create Activity:** Alarm
5. Click on **Finish.**

Coding the Application

1. Open **AndroidMainfest.xml** which is located in **res->values-> AndroidMainfest.xml.** This file will hold all of the text that our layout will use.
2. Click on the **AndroidMainfest**.xml at the bottom to bring up the raw xml file.

Editing the the java code

1. Open SampleApp.java from the left hand side.
2. Save the files.

Running the Application

1. Click on the green circle with the white arrow.
2. Choose the AVD that we created in a previous step.
3. The android AVD will load and the program will run.

PROGRAMS

FileName :MainActivity.java

```
package com.lab.alarmclock;
import java.util.Calendar;
import android.app.Activity;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
```

```
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TimePicker;

public class AlarmActivity extends Activity {

	private TimePicker timepicker;
	private Context context;
	private Button btnSetAlarm;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);

		context = this;

		timepicker = (TimePicker) findViewById(R.id.timepicker);

		btnSetAlarm = (Button) findViewById(R.id.btnSetAlarm);
		btnSetAlarm.setOnClickListener(new OnClickListener() {

			@Override
			public void onClick(View v) {
				// TODO Auto-generated method stub

				Calendar calendar = Calendar.getInstance();
				calendar.set(Calendar.HOUR_OF_DAY, timepicker.getCurrentHour());
				calendar.set(Calendar.MINUTE, timepicker.getCurrentMinute());

				Intent myIntent = new Intent(context, AlarmReceiver.class);
				PendingIntent pendingIntent = PendingIntent.getBroadcast(
						context, 0, myIntent, 0);

				AlarmManager alarmManager = (AlarmManager)
getSystemService(ALARM_SERVICE);
				alarmManager.set(AlarmManager.RTC, calendar.getTimeInMillis(),
						pendingIntent);

			}
		});

	};
```

```
}
```

File Name: Alaram Reciever.java

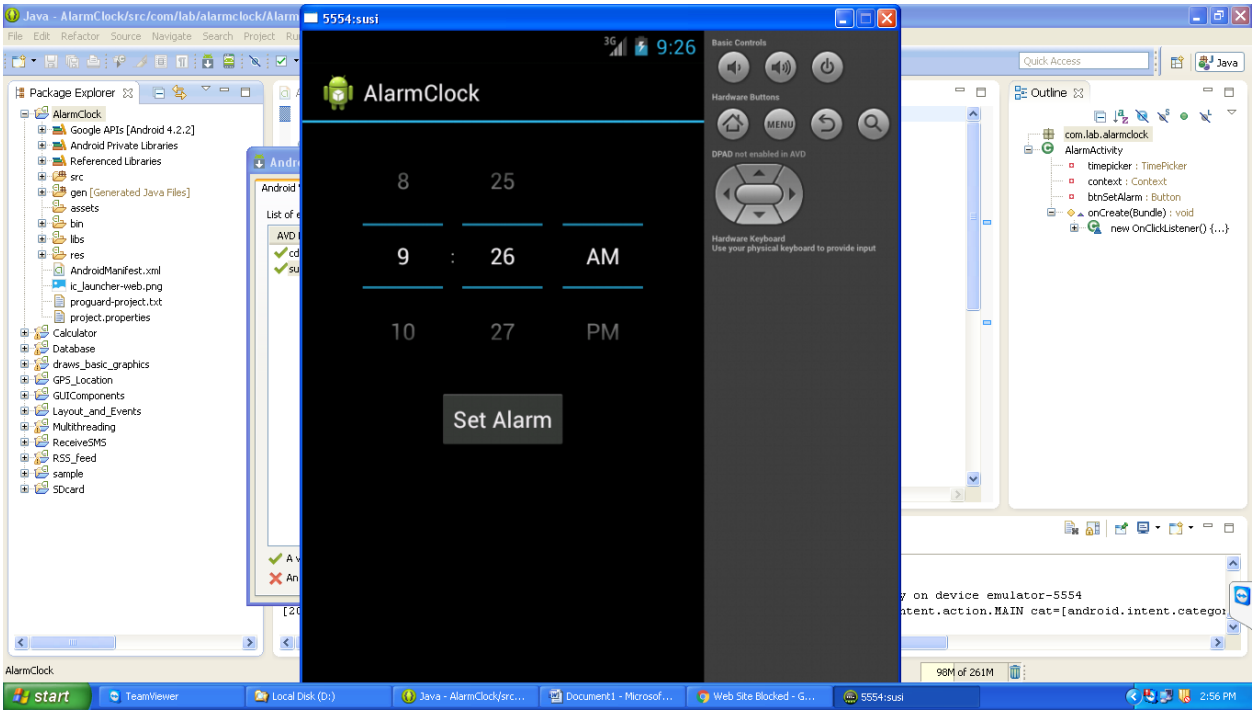
```
package com.lab.alarmclock;

import android.content.Context;
import android.content.Intent;
import android.media.Ringtone;
import android.media.RingtoneManager;
import android.net.Uri;
import android.support.v4.content.WakefulBroadcastReceiver;
import android.util.Log;
import android.widget.Toast;

public class AlarmReceiver extends WakefulBroadcastReceiver {

	@Override
	public void onReceive(Context context, Intent intent) {
		// TODO Auto-generated method stub

		Log.e("alarmreceiver", "alarmreceiver");
		Toast.makeText(context, "alarmreceiver", Toast.LENGTH_LONG).show();

		Uri alarmUri = RingtoneManager
				.getDefaultUri(RingtoneManager.TYPE_ALARM);

		Ringtone ringtone = RingtoneManager.getRingtone(context, alarmUri);

		ringtone.play();

	}
}
```

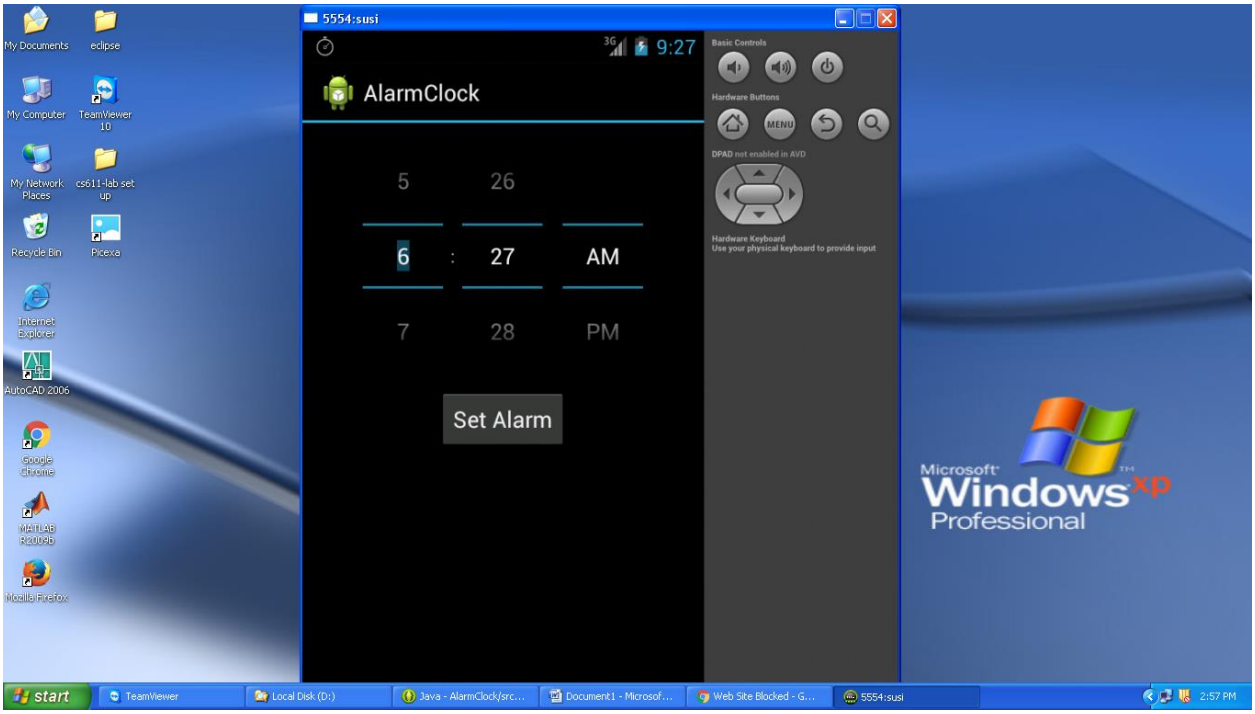
File Name: Androidmainfest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.lab.alarmclock"
   android:versionCode="1"
   android:versionName="1.0" >

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

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="21" />
```

```
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".AlarmActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".AlarmReceiver" />
    </application>

</manifest>
```

OUTPUT:





RESULT:

Thus the mobile application that creates alarm clock is executed successfully by using Eclipse.

VIVA QUESTIONS AND ANSWERS

1) What is Android?

It is an open-sourced operating system that is used primarily on mobile devices, such as cell phones and tablets. It is a Linux kernel-based system that's been equipped with rich components that allows developers to create and run apps that can perform both basic and advanced functions.

2) What Is the Google Android SDK?

The Google Android SDK is a toolset that developers need in order to write apps on Android enabled devices. It contains a graphical interface that emulates an Android driven handheld environment, allowing them to test and debug their codes.

3) What is the Android Architecture?

Android Architecture is made up of 4 key components:
– Linux Kernel
– Libraries
– Android Framework
– Android Applications

4) Describe the Android Framework.

The Android Framework is an important aspect of the Android Architecture. Here you can find all the classes and methods that developers would need in order to write applications on the Android environment.

5) What is AAPT?

AAPT is short for Android Asset Packaging Tool. This tool provides developers with the ability to deal with zip-compatible archives, which includes creating, extracting as well as viewing its contents.

6) What is the importance of having an emulator within the Android environment?

The emulator lets developers "play" around an interface that acts as if it were an actual mobile device. They can write and test codes, and even debug. Emulators are a safe place for testing codes especially if it is in the early design phase.

7) What is the use of an activityCreator?

An activityCreator is the first step towards the creation of a new Android project. It is made up of a shell script that will be used to create new file system structure necessary for writing codes within the Android IDE.

8) Describe Activities.

Activities are what you refer to as the window to a user interface. Just as you create windows in order to display output or to ask for an input in the form of dialog boxes, activities play the same role, though it may not always be in the form of a user interface.

9) What are Intents?

Intents displays notification messages to the user from within the Android enabled device. It can be used to alert the user of a particular state that occurred. Users can be made to respond to intents.

10) Differentiate Activities from Services.

Activities can be closed, or terminated anytime the user wishes. On the other hand, services are designed to run behind the scenes, and can act independently. Most services run continuously, regardless of whether there are certain or no activities being executed.

11) What items are important in every Android project?

These are the essential items that are present each time an Android project is created:
– AndroidManifest.xml
– build.xml
– bin/
– src/
– res/
– assets/

12) What is the importance of XML-based layouts?

The use of XML-based layouts provides a consistent and somewhat standard means of setting GUI definition format. In common practice, layout details are placed in XML files while other items are placed in source files.

13) What are containers?

Containers, as the name itself implies, holds objects and widgets together, depending on which specific items are needed and in what particular arrangement that is wanted. Containers may hold labels, fields, buttons, or even child containers, as examples.

14) What is Orientation?

Orientation, which can be set using setOrientation(), dictates if the LinearLayout is represented as a row or as a column. Values are set as either HORIZONTAL or VERTICAL.

15) What is the importance of Android in the mobile market?

Developers can write and register apps that will specifically run under the Android environment. This means that every mobile device that is Android enabled will be able to support and run these apps. With the growing popularity of Android mobile devices, developers can take advantage of this trend by creating and uploading their apps on the Android Market for distribution to anyone who wants to download it.

16) What do you think are some disadvantages of Android?

Given that Android is an open-source platform, and the fact that different Android operating systems have been released on different mobile devices, there's no clear cut policy to how applications can adapt with various OS versions and upgrades. One app that runs on this particular version of Android OS may or may not run on another version. Another disadvantage is that since mobile devices such as phones and tabs come in different sizes and forms, it poses a challenge for developers to create apps that can adjust correctly to the right screen size and other varying features and specs.

17) What is adb?

Adb is short for Android Debug Bridge. It allows developers the power to execute remote shell commands. Its basic function is to allow and control communication towards and from the emulator port.

18) What are the four essential states of an activity?

– Active – if the activity is at the foreground
– Paused – if the activity is at the background and still visible
– Stopped – if the activity is not visible and therefore is hidden or obscured by another activity
– Destroyed – when the activity process is killed or completed terminated

19) What is ANR?

ANR is short for Application Not Responding. This is actually a dialog that appears to the user whenever an application have been unresponsive for a long period of time.

20) Which elements can occur only once and must be present?

Among the different elements, the and elements must be present and can occur only once. The rest are optional, and can occur as many times as needed.

21) How are escape characters used as attribute?

Escape characters are preceded by double backslashes. For example, a newline character is created using '\\n'

22) What is the importance of settings permissions in app development?

Permissions allow certain restrictions to be imposed primarily to protect data and code. Without these, codes could be compromised, resulting to defects in functionality.

23) What is the function of an intent filter?

Because every component needs to indicate which intents they can respond to, intent filters are used to filter out intents that these components are willing to receive. One or more intent filters are possible, depending on the services and activities that is going to make use of it.

24) Enumerate the three key loops when monitoring an activity

– Entire lifetime – activity happens between onCreate and onDestroy
– Visible lifetime – activity happens between onStart and onStop
– Foreground lifetime – activity happens between onResume and onPause

25) When is the onStop() method invoked?

A call to onStop method happens when an activity is no longer visible to the user, either because another activity has taken over or if in front of that activity.

26) Is there a case wherein other qualifiers in multiple resources take precedence over locale?

Yes, there are actually instances wherein some qualifiers can take precedence over locale. There are two known exceptions, which are the MCC (mobile country code) and MNC (mobile network code) qualifiers.

27) What are the different states wherein a process is based?

There are 4 possible states:
– foreground activity
– visible activity
– background activity
– empty process

28) How can the ANR be prevented?

One technique that prevents the Android system from concluding a code that has been responsive for a long period of time is to create a child thread. Within the child thread, most of the actual workings of the codes can be placed, so that the main thread runs with minimal periods of unresponsive times.

29) What role does Dalvik play in Android development?

Dalvik serves as a virtual machine, and it is where every Android application runs. Through Dalvik, a device is able to execute multiple virtual machines efficiently through better memory management.

30) What is the AndroidManifest.xml?

This file is essential in every application. It is declared in the root directory and contains information about the application that the Android system must know before the codes can be executed.

31) What is the proper way of setting up an Android-powered device for app development?

The following are steps to be followed prior to actual application development in an Android-powered device:
-Declare your application as “debuggable” in your Android Manifest.
-Turn on “USB Debugging” on your device.
-Set up your system to detect your device.

32) Enumerate the steps in creating a bounded service through AIDL.

1. create the .aidl file, which defines the programming interface
2. implement the interface, which involves extending the inner abstract Stub class as well as implanting its methods.
3. expose the interface, which involves implementing the service to the clients.

33) What is the importance of Default Resources?

When default resources, which contain default strings and files, are not present, an error will occur and the app will not run. Resources are placed in specially named subdirectories under the project res/ directory.

34) When dealing with multiple resources, which one takes precedence?

Assuming that all of these multiple resources are able to match the configuration of a device, the ‘locale’ qualifier almost always takes the highest precedence over the others.

35) When does ANR occur?

The ANR dialog is displayed to the user based on two possible conditions. One is when there is no response to an input event within 5 seconds, and the other is when a broadcast receiver is not done executing within 10 seconds.

36) What is AIDL?

AIDL, or Android Interface Definition Language, handles the interface requirements between a client and a service so both can communicate at the same level through interprocess communication or IPC. This process involves breaking down objects into primitives that Android can understand. This part is required simply because a process cannot access the memory of the other process.

37) What data types are supported by AIDL?

AIDL has support for the following data types:
-string

-charSequence
-List
-Map
-all native Java data types like int,long, char and Boolean

38) What is a Fragment?

A fragment is a part or portion of an activity. It is modular in a sense that you can move around or combine with other fragments in a single activity. Fragments are also reusable.

39) What is a visible activity?

A visible activity is one that sits behind a foreground dialog. It is actually visible to the user, but not necessarily being in the foreground itself.

40) When is the best time to kill a foreground activity?

The foreground activity, being the most important among the other states, is only killed or terminated as a last resort, especially if it is already consuming too much memory. When a memory paging state has been reach by a foreground activity, then it is killed so that the user interface can retain its responsiveness to the user.

41) Is it possible to use or add a fragment without using a user interface?

Yes, it is possible to do that, such as when you want to create a background behavior for a particular activity. You can do this by using add(Fragment,string) method to add a fragment from the activity.

42) How do you remove icons and widgets from the main screen of the Android device?

To remove an icon or shortcut, press and hold that icon. You then drag it downwards to the lower part of the screen where a remove button appears.

43) What are the core components under the Android application architecture?

There are 5 key components under the Android application architecture:
– services
– intent
– resource externalization
– notifications
– content providers

44) What composes a typical Android application project?

A project under Android development, upon compilation, becomes an .apk file. This apk file format is actually made up of the AndroidManifest.xml file, application code, resource files, and other related files.

45) What is a Sticky Intent?

A Sticky Intent is a broadcast from sendStickyBroadcast() method such that the intent floats around even after the broadcast, allowing others to collect data from it.

46) Do all mobile phones support the latest Android operating system?

Some Android-powered phone allows you to upgrade to the higher Android operating system version. However, not all upgrades would allow you to get the latest version. It depends largely on the capability and specs of the phone, whether it can support the newer features available under the latest Android version.

47) What is portable wi-fi hotspot?

Portable Wi-Fi Hotspot allows you to share your mobile internet connection to other wireless device. For example, using your Android-powered phone as a Wi-Fi Hotspot, you can use your laptop to connect to the Internet using that access point.

48) What is an action?

In Android development, an action is what the intent sender wants to do or expected to get as a response. Most application functionality is based on the intended action.

49) What is the difference between a regular bitmap and a nine-patch image?

In general, a Nine-patch image allows resizing that can be used as background or other image size requirements for the target device. The Nine-patch refers to the way you can resize the image: 4 corners that are unscaled, 4 edges that are scaled in 1 axis, and the middle one that can be scaled into both axes.

50) What language is supported by Android for application development?

The main language supported is Java programming language. Java is the most popular language for app development, which makes it ideal even for new Android developers to quickly learn to create and deploy applications in the Android environment.