Q1. what is difference between Interface & Abstract Class, explain with example? Create an Abstract Class for Vehicle and inherited classes for Boat and Truck. Explain the code in each step.

Ans :- **Abstract Class**

An Abstract Class is a restricted class that can't be used to create objects. This type of class can only be used when inherited, i.e. as a superclass. The abstract keyword is used to make a class declaration abstract type.

→ Abstract Class have both abstract as well as non-abstract methods
→ They don't support multiple inheritance
→ They can have final, non-final, static, non-static variables
→ An Abstract class can extend another Java class and implement multiple Java Interfaces.
→ The "extends" keyword is used to extend an Abstract Class
→ The class members of Abstract class can be private, public, protected etc.

Syntax :- `abstract` (keyword) `class name` (name of the class) `{`

```
// code
}
```

Example :-

```
abstract class Shape{
    int area = 0;
    abstract public int getArea(); // Abstract method to calculate area
}
public class Circle extends Shape {
    public static void main(String args[]) {
        Circle c = new Circle(10);
        c.area = c.getArea();
        System.out.println("Area" + c.area);
    } int radius; // Data member
    //constructor.
    Circle(int radius) {
        super();
        this.radius = radius;
    }
    // implementing the abstract method of abstract
    // class
    public int getArea() {
        return 3.14 * (this.radius) * (this.radius);
    }
}
```

The Circle class extends the abstract class Shape, that has an abstract method getArea(), which is implemented in the Circle class.

Output :- 314

Interface

An interface is a completed abstract class that is used to group related methods with empty bodies. The interface keyword is used to create interfaces. The interface is implemented by another class using implement keyword.

→ Interface can only have abstract methods
→ They support multiple inheritance unlike Abstract classes
→ They only have static and final variables
→ An interface can extend another Java interface
→ Members of Java interface are public by default.
→ They can't be used to create object just like Abstract classes.
→ The body of the interface methods are provided by the class that implements the interface.

Syntax:-

```
interface name{
    // ~~code~~ abstract methods
}
```

Example:-

```
interface Shape {
    public void getArea(); // abstract by default
}

public class Circle implements Shape {
    public void getArea() {
        System.out.println(3.14 * this.radius * this.radius);
    }
    // Data Members
    int radius;
    // constructor
    Circle(int radius) {
        this.radius = radius;
    }
    public static void main(String args[]) {
        Circle c = new Circle(10);
        c.getArea();
    }
}
```

The Circle class implements the Shape interface that has a getArea() method which is defined by the Circle class

[Output] :- 314.

```
abstract class Vehicle {
    String name = " ";
    Vehicle(String name) {
        this.name = name;
    }
    abstract public String getName();
}
// Creating the Boat class
public class Boat extends Vehicle {
    String boatType;
    Boat(String name, String boatType){
        super(name);
        this.boatType = boatType;
    }
    public String getName(){
        return "Boat Name" + this.name;
    }
}
// Creating the Truck class
public class Truck extends Vehicle {
    String noOfWheels;
    Truck(String name, String wheels){
        super(name);
        this.noOfWheels = wheels;
    }
    public String getName(){
        return "Truck Name" + this.name;
    }
}
```

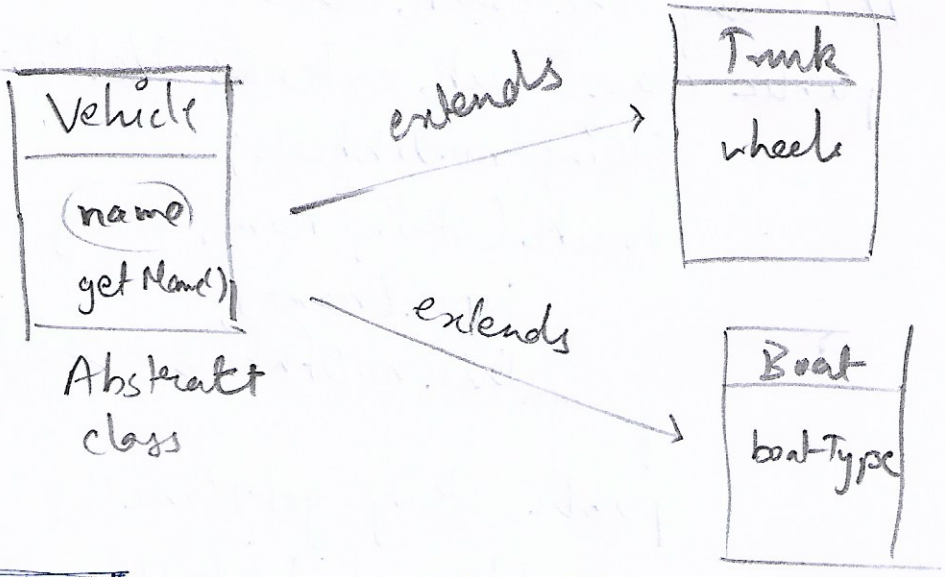
```
class Demo{
    public static void main(String args[]){
        Truck t = new Truck("T","14");
        Boat b= new Boat("B","DeikBoat");

        t.get
        System.out.println(t.getName());
        System.out.println(b.getName());
    }
}
```

The above code of Vehicle, Boat, Truck class is in a file named Demo.java.

→ In the Demo class's main method we create one Boat Object and one Truck object. Both these classes were extended by the abstract class Vehicle. We call the getName() method to get the Name of object with its type.

→



→ Output:- Truck Name T
Boat Name B

Q2. What is difference between the layout.xml & manifest.xml? Write user layout and activity code for below Application Screen of Geographical Data for states. State is a spinner with the values Delhi, Uttar Pradesh, Himachal, Bihar, Haryana and Rajisthan and all other elements as per the screen below. Population, Literary Rate, Hospital per 1000 and Poverty rate are positive and numeric types. On submitting the Data, app will toast a success or failure message based on validation in Activity Class.

Application Screen

Geographical Data - 2020 (TextView)	
State (TextView)	Spinner
Population (TextView)	Edit Text
Literary View (TextView)	Edit Text
Hospital per 1000 (TextV)	Edit Text
Poverty Rate (TextView)	Edit Text
Submit (Button)	

Ans :-

Layout.xml / Layout-XML files

These files are used to define the actual UI (User Interface) of our application. It holds all the elements (views) or the tools that we want to use in our application, like Button, EditText, TextView etc.

These files are inside the app/res/layout folder of the Android Project. These files are tied up with the different activities and fragments that our Application has.

Manifest.xml

This xml file is used to define all components of our application. It includes names of :-

- Application Packages
- Activities
- Services
- Permissions

that our Application have/needs.

This file is located in the app/manifests folder of the Android studio Project.

The UI of the given Application in the question is in a file named activity_main.xml, the Java code is in MainAcitivity.java.

⇒ activity_main.xml

```
<?xml version=1.0 encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas
    android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:padding_left="32dp"
    android:padding_right="32dp" >

    <TextView
        android:id="@+id/title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Geographical Data-2020"
    />

    <TextView
        android:id="@+id/titleState"
        android:layout_width="50dp"
        android:layout_height="0dp wrap_content"
        android:text="State"
        android:layout_below="@id/title"
        android:layout_alignParentLeft="true"
    />

    <Spinner
        android:id="@+id/StateSpinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/title"
        android:layout_alignParentRight="true"
    />
```

```
<TextView
    android:id="@+id/populationText"
    android:layout_width="50dp"
    android:layout_height="wrap_content"
    android:layout_below="@id/titleState"
    android:layout_alignParentLeft="true"
    android:text="Population"
/>
<EditText
    android:id="@+id/population"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_below="@id/StateSpinner"
    android:layout_alignParentRight="true"
    android:digits="0123456789"
    android:inputType="number"
/>
<TextView
    android:id="@+id/literacyText"
    android:layout_width="50dp"
    android:layout_height="wrap_content"
    android:layout_below="@id/populationText"
    android:layout_alignParentLeft="true"
    android:text="Literacy Rate"
/>
```

```
< Edit Text
    android : id = "@+id / literacy"
    android : layout_width = "0dp"
    android : layout_height = "wrap_content"
    android : below = "@id/ population"
    android : layout_alignParentRight = "true"
    android : digits = "0123456789"
    android : inputType = "number"

>

< Text View
    android : id = "@+id / hospitalText"
    android : layout_width = "50dp"
    android : layout_height = "wrap_content"
    android : layout_alignParentLeft = True
    android : below = "@id/ literacyText"
    android : text = "Hospital per 1000"

>

< Edit Text
    android : id = "@+id/hospital"
    android : layout_width = "0dp"
    android : layout_height = "wrap_content"
    android : below = "@id/ populaliteracy"
    android : layout_alignParentRight = "true"
    android : digits = "0123456789"
    android : inputType = "number"

>
```

```
<TextView
    android:id="@+id/poverityTent"
    android:layout_width="50dp"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@id/hospitalTent"
    android:text="Poverty Rate"
>
<EditText
    android:id="@+id/poverty"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_below="@id/hospital"
    android:layout_alignParentRight="true"
    android:digits="0123456789"
    android:InputType="number"
>
<Button
    android:id="@+id/submit"
    android:layout_width="match_parent"
    android:layout_height="wrap_parent"
    android:layout_below="@id/poverityTent"
    android:text="Submit"
    android:onClick="submit"
>
</RelativeLayout>
```

Main Activity.java

```
import android.widget.Spinner
public class MainActivity extends AppCompatActivity {
    String state;
    Spinner stateSpinner;

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

        // Create a Array for Spinner elements
        String[] states = {"Delhi","Uttar Pradesh"
,"Himachal","Bihar","Haryana","Rajasthan"};

        stateSpinner = (Spinner) findViewById(
                R.id.StateSpinner);

    ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this, android.R.layout.
simple_spinner_item, states);

    adapter.setDropDownViewResource(android.R.
layout.simple_spinner_dropdown_item);

    stateSpinner.setAdapter(adapter);

    // Spinner Selecting event handling

    stateSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?>
adapterView, View view, int i, long l) {
```

```
            state = adapterView.getItemAtPosition(i)
                    .toString();
        });
    }
    // OnClick handler for submit button.
    public void submit(View view){
        EditText E_population = (EditText) findViewById(
                R.id.population);
        EditText E_literary = (EditText) findViewById(
                R.id.literary);
        EditText E_hospital = (EditText) findViewById(
                R.id.hospital);
        EditText E_poverty = (EditText) findViewById(
                R.id.poverty);
        // getting the text from Edit Text views.
        String population = E_population.getText().toString();
        String literary = E_literary.getText().toString();
        String hospital = E_hospital.getText().toString();
        String poverty = E_poverty.getText().toString();
        // validating and displaying Toast.
        if (validate(state, population, literary,
                hospital, poverty)){
            Toast.makeText(this, "Success", Toast.
                    LENGTH_LONG).show();
        }
```

```
        else {
            Toast.makeText(this, "Failure", Toast.
                    LENGTH_LONG).show();
        }
    }
}
public int validate(String state, String population
    String literacy, String hospital, String poverty)
    { //Validation Logic

    }
```

Q3. Write the code for the creation of a Database named "Unix" and a table named "dir" using SQLite Database for Android Application, then write the code for insertion of data into database by passing ContentValues Object

Structure of Table.

Column Name	Data Type	Attributes
_id	INTEGER	Primary Key, Auto increment
dir_name	TEXT	
dir_size	INTEGER	
dir_location	TEXT	
creation_date	TEXT	

Insert the below data in Dir Table

dir_name	dir_size	dir_location	creation_date
"users"	10056	"/users/"	"2020-02-01 05:52:25.055"
"system"	10024	"/lib/system/"	"2020-02-01 05:52:25.055"

Ans :-

DatabaseHelper.java

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.content.ContentValues;
public class DatabaseHelper extends SQLiteOpenHelper {

    // Database Name & Version
    public static final String DB_Name = "Unin";
    public static final int v = "1";
                                VERSION

    // Table Name
    public static final String TABLE = "dir";

    // Table Columns
    public static final String _id = "_id";
    public static final String dir_name = "dir_name";
    public static final String dir_size = "dir_size";
    public static final String dir_loc = "dir_location";
    public static final String cre_date = "creation date";

    // Query to Create the Database & the Table.
    private static final String CREATE_TABLE =
        "CREATE TABLE " + TABLE + "( " + _id
        + " INTEGER PRIMARY KEY AUTO INCREMENT," +
        dir_name + " TEXT," + dir_size + " INTEGER TEXT," +
        dir_loc + " TEXT," + cre_date + " TEXT );";
    );

    // constructor
    public DatabaseHelper(Context context) {
        super(context, DB_NAME, null, VERSION);
    }
```

```
// On create method.
@Override
public void onCreate (SQLiteDatabase db) {
        db.execSQL(CREATE_TABLE);
        // inserting records on db creation
        insertDir("users", 10056, "/users/", "2020-02-01
                                    50:52:25:055");
        insertDir("system", 10024, "lib/system/", "2020-02
                                    -01-05:52:25:055");
    }

// insertDir method that uses ContentValue Object
// to perform SQL insert operation.
private static void insertDir(String name, int size,
        String location, String date, SQLiteDatabase db)
{   ContentValues toInsert = new ContentValues();
    toInsert.put("dir_name, name);
    toInsert.put(dir_size, size);
    toInsert.put(dir_loc, location);
    toInsert.put(dir cre_date, date);
    // performing Insert
    db.insert(TABLE_NAME, null, toInsert);
    }
}
```


We can create the object of this class in our Main Activity or any activity to use the SQLiteDatabase, as well the methods of this class to perform CRUD operation.