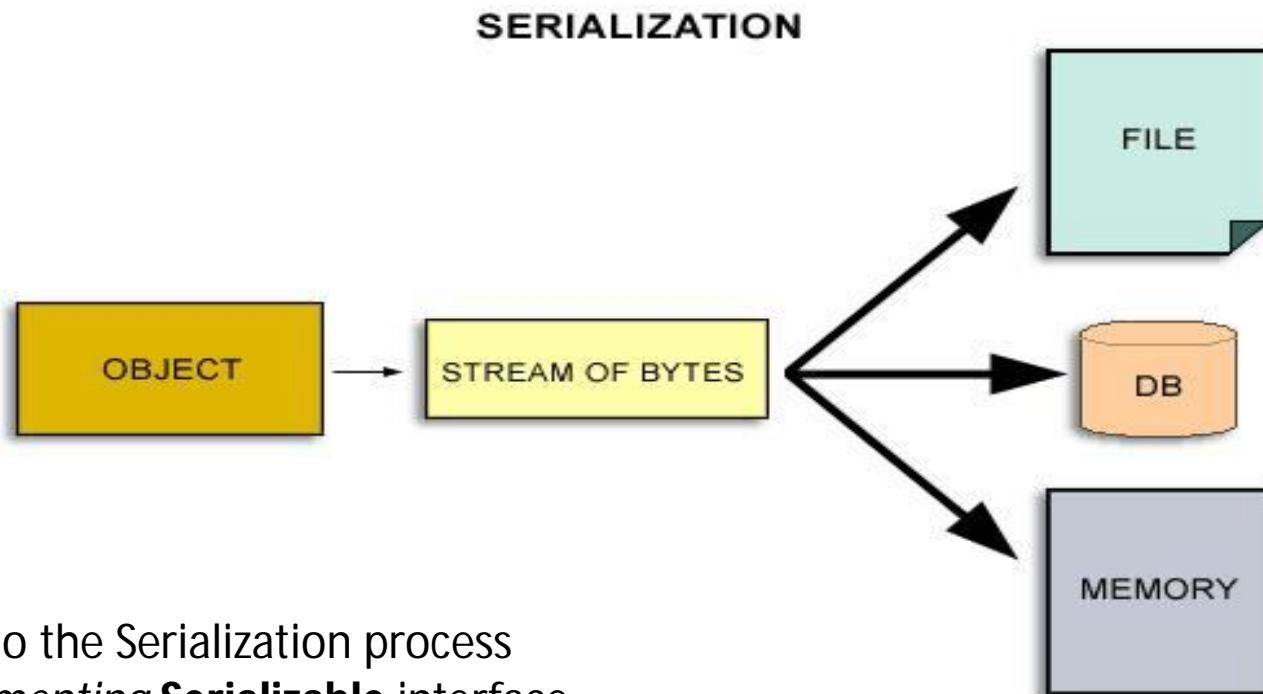


# Serialization in JAVA.

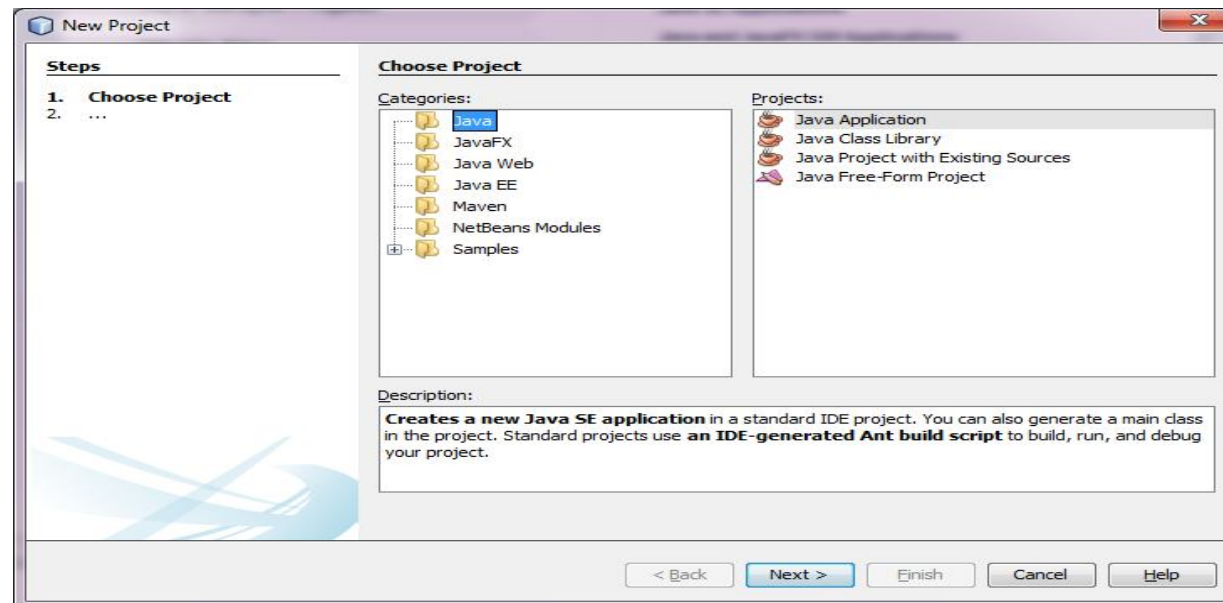
"**Serialization** is the process of converting the data(Objects) into stream of bytes and storing in to the files or database."



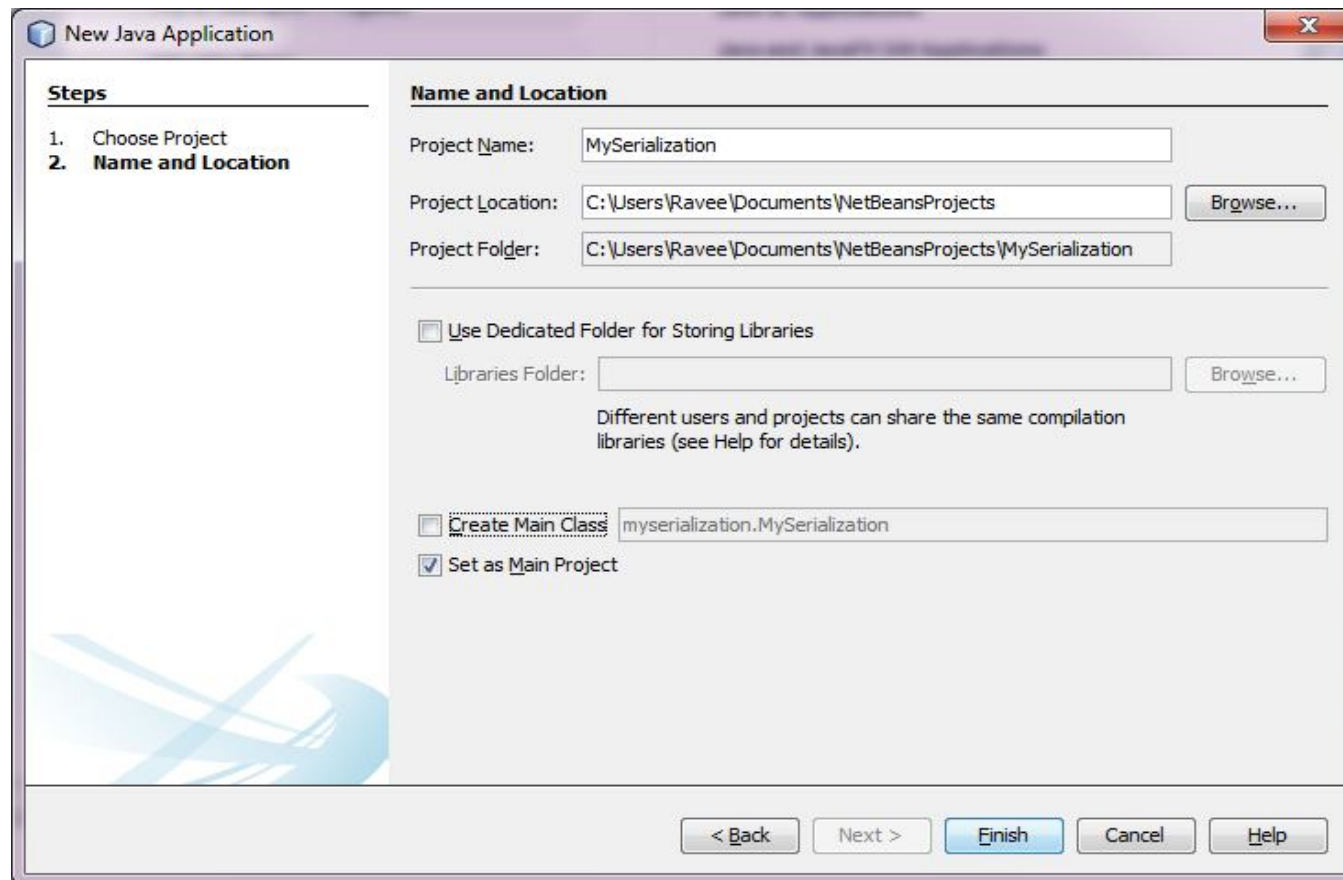
We can do the Serialization process by *implementing* **Serializable** interface.

# How to Create Serialized Object?

- Step 1: First lets create a new project in netbeans.
- File → New Project
- You will be to see following window.



Select java application from the above form then click next.



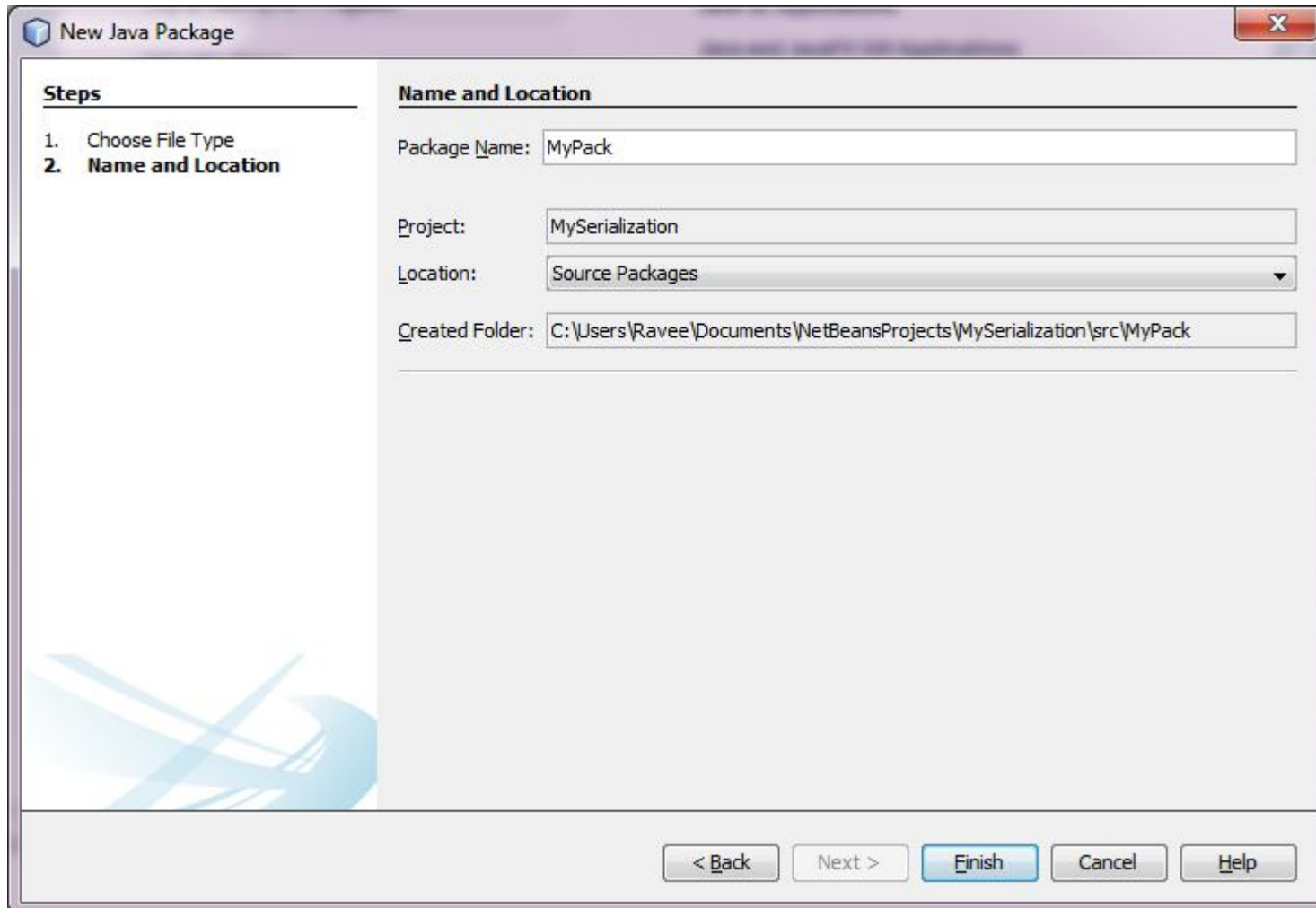
Give an appropriate name to the application in this ,I have given "MySerialization" select the required location and folder for the project.

Then click on Finish.

Step 2:- Then create a Package which will help in managing the project.

Right click on (Default Package) → Add new → Java Package.

Give an appropriate name to the package and click on Finish.



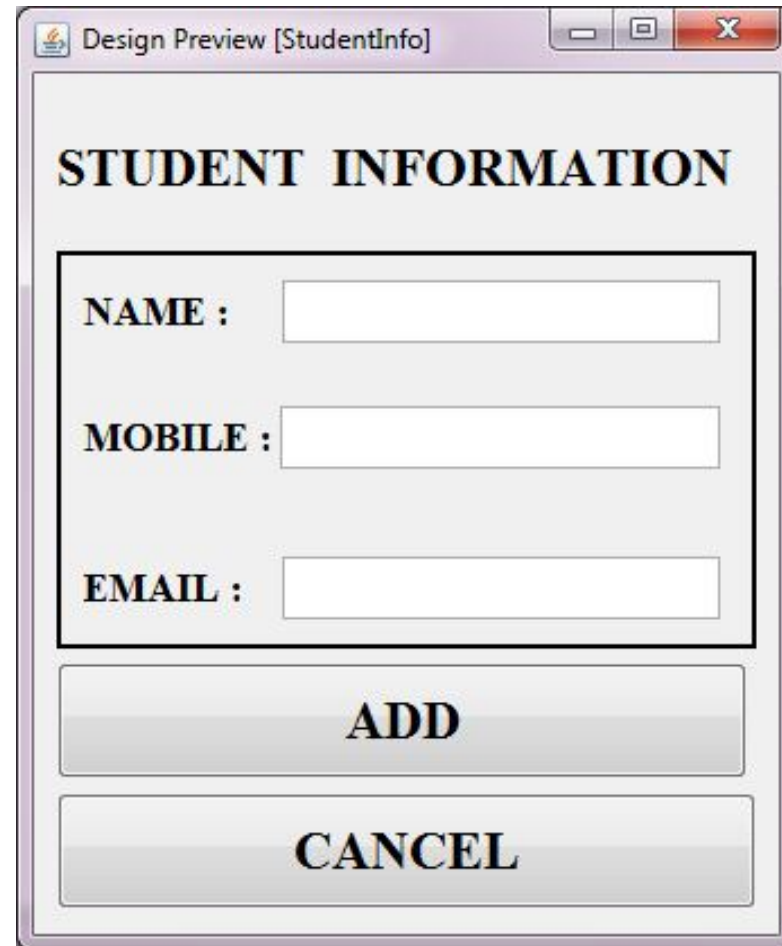
The screenshot shows the 'New Java Package' dialog box. On the left, under the 'Steps' section, step 2 'Name and Location' is selected. The main area on the right is titled 'Name and Location' and contains the following fields:

- Package Name:** MyPack
- Project:** MySerialization
- Location:** Source Packages (selected from a dropdown menu)
- Created Folder:** C:\Users\Ravee\Documents\NetBeansProjects\MySerialization\src\MyPack

At the bottom of the dialog, there are five buttons: '< Back', 'Next >', 'Finish' (highlighted in blue), 'Cancel', and 'Help'.

- Step 3: Now lets create a GUI form. Using which we will be adding data to serialized object.
- Lets first add a from to our package.  
(Right click on package)→New →JFrame Form  
Give a name to the form and click on finish.
- Lets take a example of student information for  
I have designed a form for the same as follows. Using the Palette provided in netbeans.

# GUI FORM FOR STUDENT INFORMATION



Design Preview [StudentInfo]

**STUDENT INFORMATION**

**NAME :**

**MOBILE :**

**EMAIL :**

**ADD**

**CANCEL**

Step 4:-After we have created GUI form. Now lets design a class.

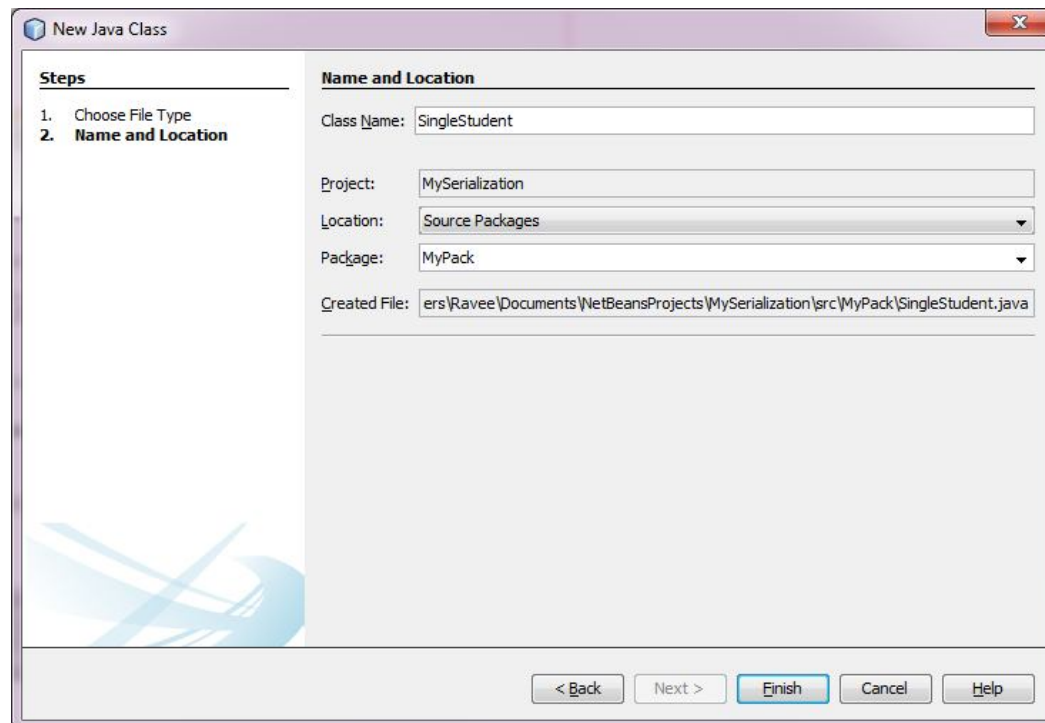
- Now we require a class for defining the fields of data we will be storing for a single student.
- As per our example we have three field name, mobile and email.
- Lets create a class Library for this.



# Now we will be adding a class. As following

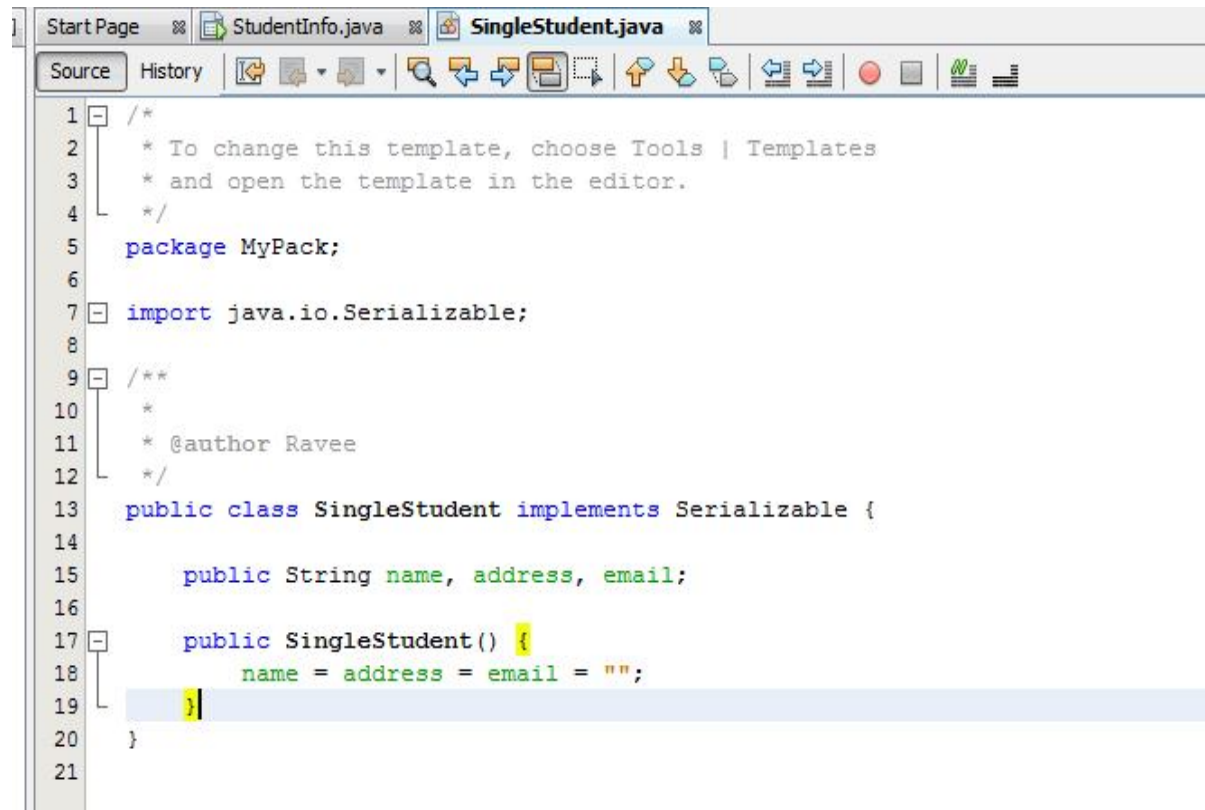
- Right click on package → New → Java Class.
- Give a name to this class as I have given singleStudent.

This class will be used as structure of data.



## Step 5: Now lets implement this class with interface "Serializable".

- After implementing, now lets add the fields to this class. So the structure of the class will be as follows.



The screenshot shows an IDE window with two tabs: 'StudentInfo.java' and 'SingleStudent.java'. The 'SingleStudent.java' tab is active, displaying the following Java code:

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5  package MyPack;
6
7  import java.io.Serializable;
8
9  /**
10   *
11   * @author Ravee
12   */
13  public class SingleStudent implements Serializable {
14
15      public String name, address, email;
16
17      public SingleStudent() {
18          name = address = email = "";
19      }
20  }
21
```

- Now we will be writing code for reading and writing data to file.

```
public void writeToFile() {           //Function for writing to file
    String file = System.getProperty("user.dir") + "\\studentData.dat"; //Name and path of the file to
    which we will be writing. System.getProperty("user.dir") this function give the relative path from the
    CWD(Currently Working Derectory) , and studentData.dat is name of the file.
    try { //As this lines are capable of throwing an exception so we have written them in try catch block
        ObjectOutputStream oo = new ObjectOutputStream(new FileOutputStream(file)); //creating an
        outputStream object.
        oo.writeObject(allStudent); // writing allStudent Vector Object to the file.
        oo.close(); // after writing closing the stream object.
        System.out.println("Written OK");
    } catch (Exception e) {
        System.out.println("ERROR IN WRITING "); // if any exception occurs printing a message.
    }
}
```

```

public void readFromFile() {                                     //Function for reading from file.
    String file = System.getProperty("user.dir") + "\\studentData.dat"; // Name and path of the
    file from which we will be reading . System.getProperty("user.dir") this function give the
    relative path from the CWD(Currently Working Derectory) , and "studentData.dat" is name of
    the file.
    try {
// creating an input stream object using which we will be able to read from file.
        ObjectInputStream oi = new ObjectInputStream(new FileInputStream(file));
// using the readObject () function. This function will return the object, so we have to typeCast it
    to vector.
        allStudent = (Vector<SingleStudent>) oi.readObject();
        oi.close();
        System.out.println("Reading OK");
    } catch (Exception e) {
        System.out.println("ERROR IN READING");
    }
}

```