Panchayat Ghar Student Management System

Techniques used to create solution

Class & Methods

This solution uses extension of classes and methods (i.e. functions) within it. I've used the following techniques to make this solution functional:

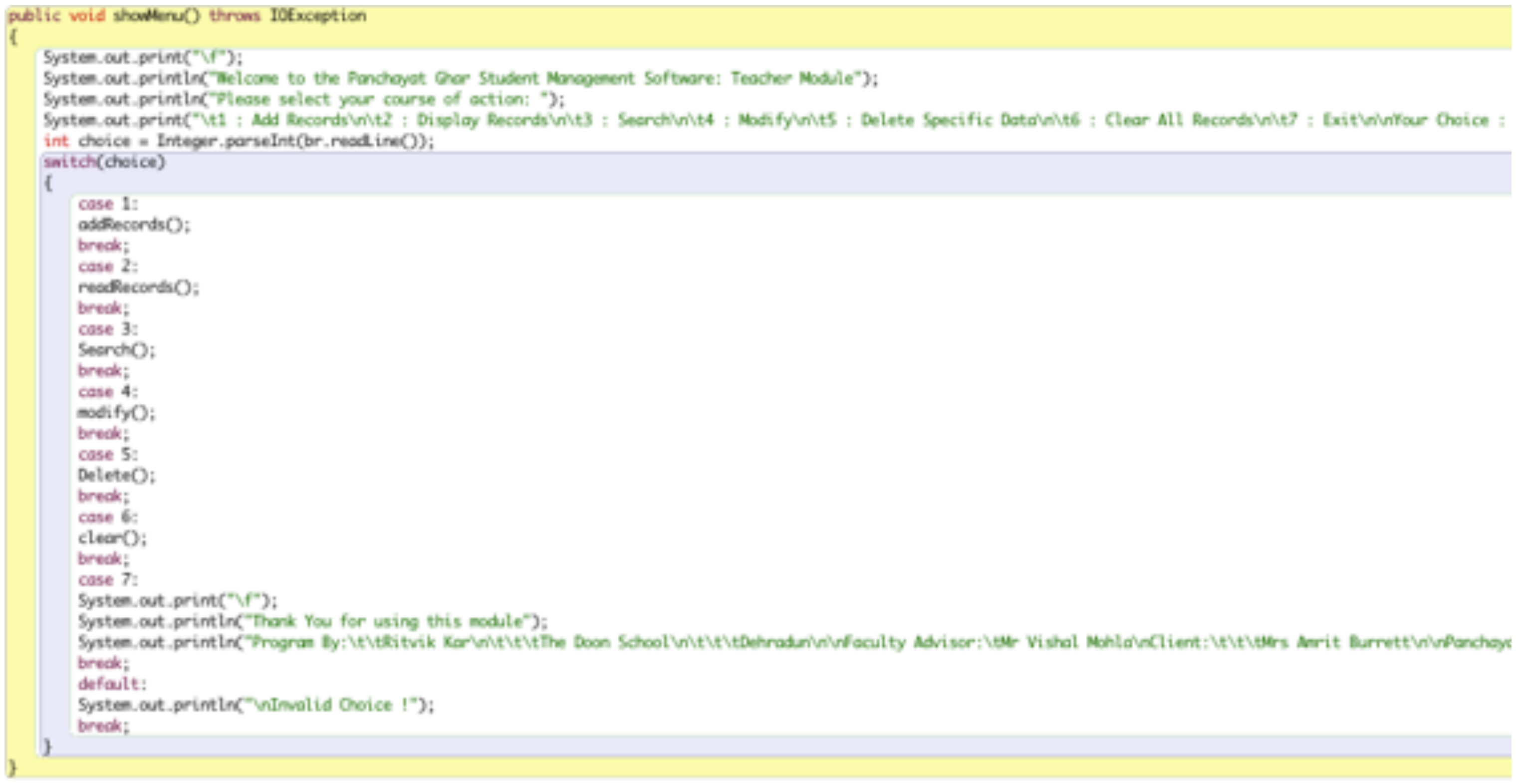
- Making a database engine using sequential file handling
- Usage of different classes to access each other
- PrintWriter and BufferedReader classes being used to accept input from keyboard and print data on .dbf file.

Every function holds a specific property which is enabled by encapsulation and abstraction. Each method will provide a specific part of the solution, shown below through the screenshots of the code, so it is simple to add new features to the solution or make amendments to the solution as each component is separated for the others.

Additional features are added by adding a separate function and calling and passing values to the others by the technique of class.method() calling operation.

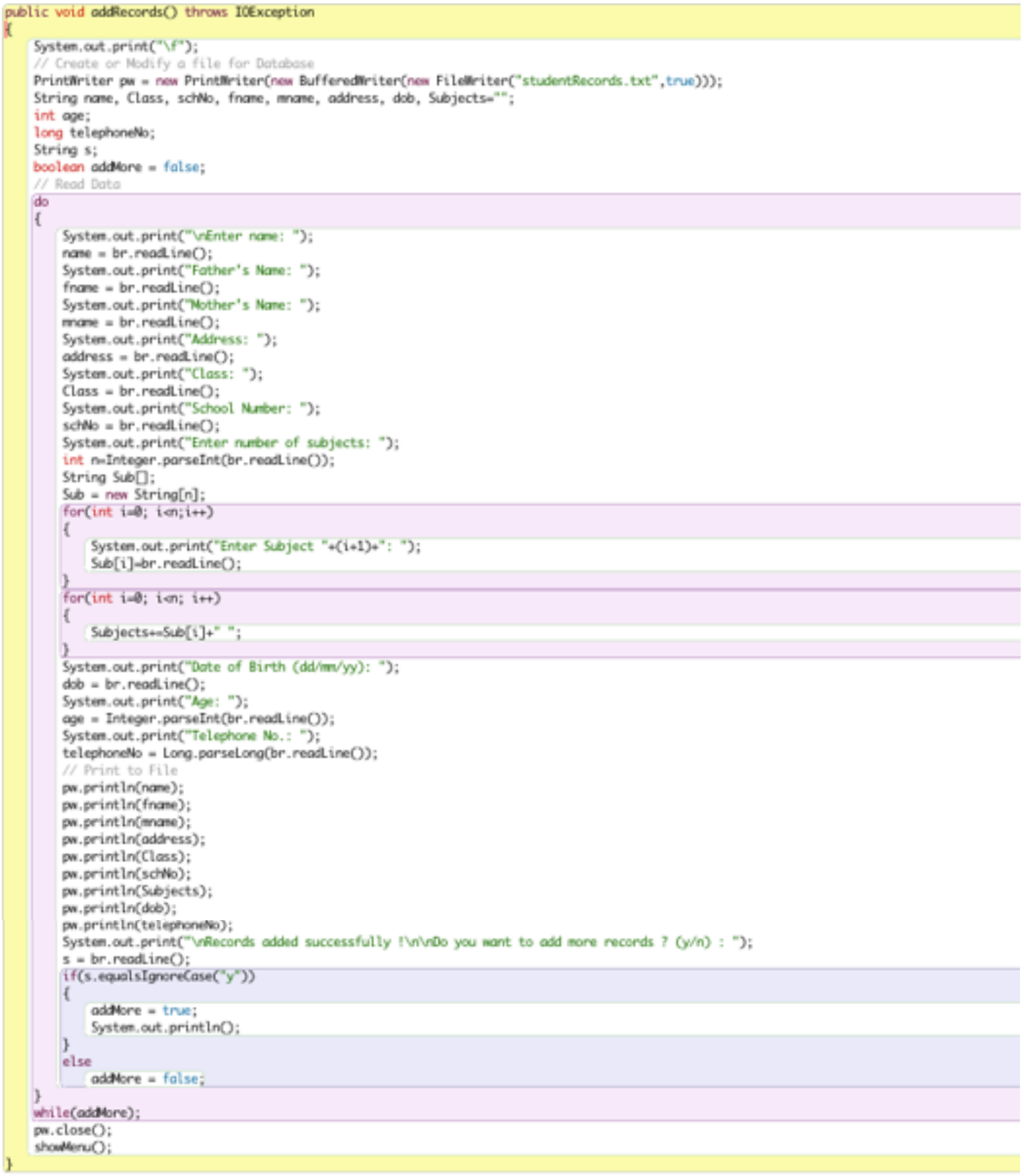
Below are examples of how functions are used in this solution to perform different functions:

(Below: `showMenu()` *function to produce the menu through out the program*)



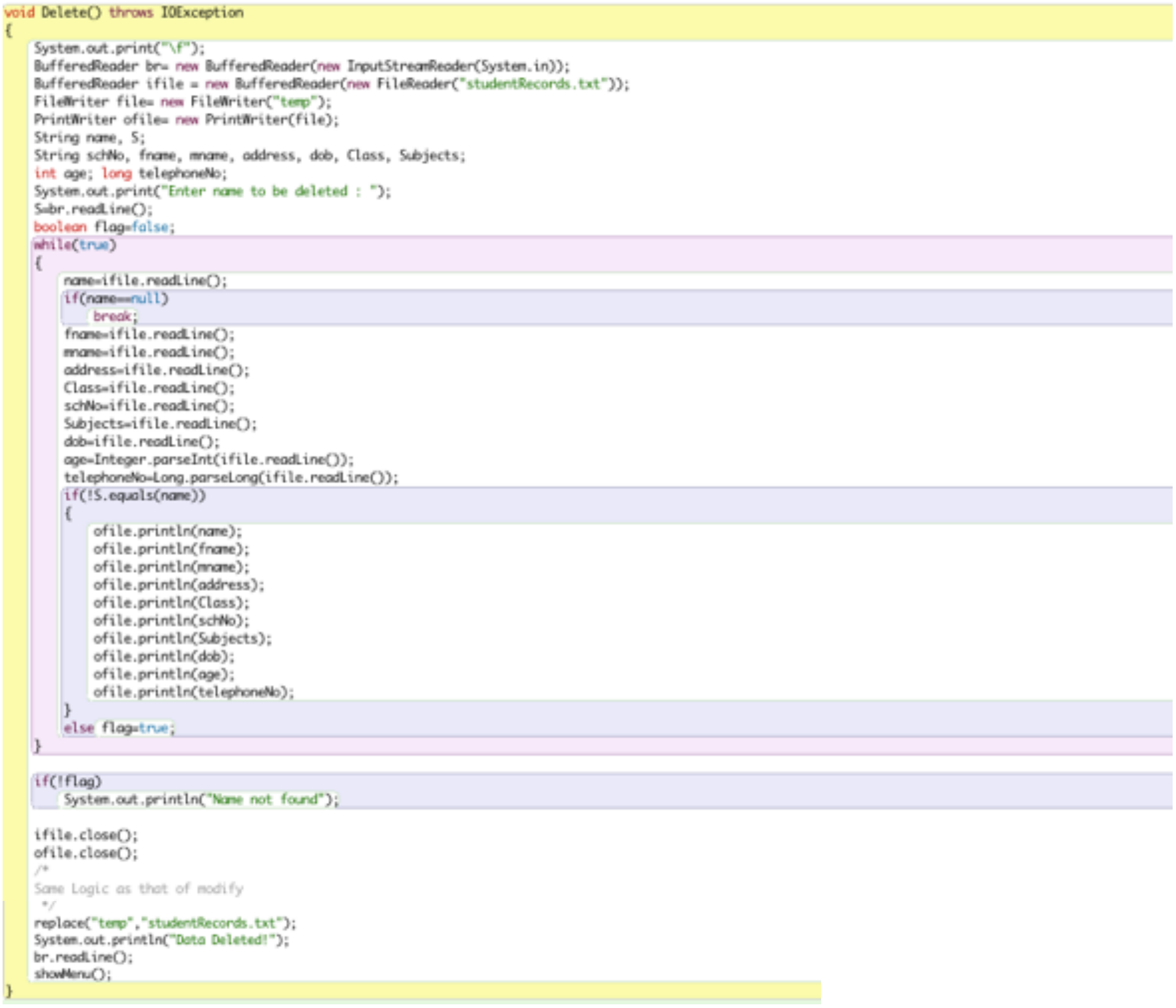
```
public void showMenu() throws IOException
{
    System.out.print("\f");
    System.out.println("Welcome to the Panchayat Ghar Student Management Software: Teacher Module");
    System.out.println("Please select your course of action: ");
    System.out.print("\t1 : Add Records\n\t2 : Display Records\n\t3 : Search\n\t4 : Modify\n\t5 : Delete Specific Data\n\t6 : Clear All Records\n\t7 : Exit\n\nYour Choice :
    int choice = Integer.parseInt(br.readLine());
    switch(choice)
    {
        case 1:
        addRecords();
        break;
        case 2:
        readRecords();
        break;
        case 3:
        Search();
        break;
        case 4:
        modify();
        break;
        case 5:
        Delete();
        break;
        case 6:
        clear();
        break;
        case 7:
        System.out.print("\f");
        System.out.println("Thank You for using this module");
        System.out.println("Program By:\t\tRitvik Kar\n\t\t\tThe Doon School\n\t\t\tDehradun\n\nFaculty Advisor:\tMr Vishal Mohla\nClient:\t\t\tMrs Amrit Burrett\n\nPanchayo
        break;
        default:
        System.out.println("\nInvalid Choice !");
        break;
    }
}
```

(Below: `addRecords()` *function to make a new document and accept individual elements)*



```
public void addRecords() throws IOException
{
    System.out.print("\f");
    // Create or Modify a file for Database
    PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter("studentRecords.txt",true)));
    String name, Class, schNo, fname, mname, address, dob, Subjects="";
    int age;
    long telephoneNo;
    String s;
    boolean addMore = false;
    // Read Data
    do
    {
        System.out.print("\nEnter name: ");
        name = br.readLine();
        System.out.print("Father's Name: ");
        fname = br.readLine();
        System.out.print("Mother's Name: ");
        mname = br.readLine();
        System.out.print("Address: ");
        address = br.readLine();
        System.out.print("Class: ");
        Class = br.readLine();
        System.out.print("School Number: ");
        schNo = br.readLine();
        System.out.print("Enter number of subjects: ");
        int n=Integer.parseInt(br.readLine());
        String Sub[];
        Sub = new String[n];
        for(int i=0; i<n;i++)
        {
            System.out.print("Enter Subject "+(i+1)+": ");
            Sub[i]=br.readLine();
        }
        for(int i=0; i<n; i++)
        {
            Subjects+=Sub[i]+" ";
        }
        System.out.print("Date of Birth (dd/mm/yy): ");
        dob = br.readLine();
        System.out.print("Age: ");
        age = Integer.parseInt(br.readLine());
        System.out.print("Telephone No.: ");
        telephoneNo = Long.parseLong(br.readLine());
        // Print to File
        pw.println(name);
        pw.println(fname);
        pw.println(mname);
        pw.println(address);
        pw.println(Class);
        pw.println(schNo);
        pw.println(Subjects);
        pw.println(dob);
        pw.println(telephoneNo);
        System.out.print("\nRecords added successfully !\n\nDo you want to add more records ? (y/n) : ");
        s = br.readLine();
        if(s.equalsIgnoreCase("y"))
        {
            addMore = true;
            System.out.println();
        }
        else
            addMore = false;
    }
    while(addMore);
    pw.close();
    showMenu();
}
```

(Below: `Delete()` *function to remove student attributes)*

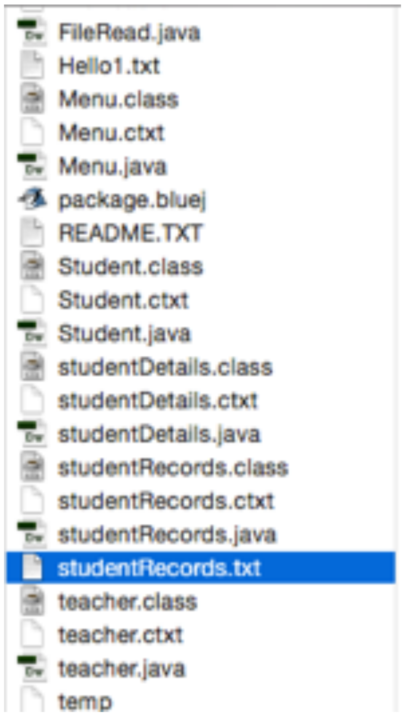


```
void Delete() throws IOException
{
    System.out.print("\f");
    BufferedReader br= new BufferedReader(new InputStreamReader(System.in));
    BufferedReader ifile = new BufferedReader(new FileReader("studentRecords.txt"));
    FileWriter file= new FileWriter("temp");
    PrintWriter ofile= new PrintWriter(file);
    String name, S;
    String schNo, fname, mname, address, dob, Class, Subjects;
    int age; long telephoneNo;
    System.out.print("Enter name to be deleted : ");
    S=br.readLine();
    boolean flag=false;
    while(true)
    {
        name=ifile.readLine();
        if(name==null)
            break;
        fname=ifile.readLine();
        mname=ifile.readLine();
        address=ifile.readLine();
        Class=ifile.readLine();
        schNo=ifile.readLine();
        Subjects=ifile.readLine();
        dob=ifile.readLine();
        age=Integer.parseInt(ifile.readLine());
        telephoneNo=Long.parseLong(ifile.readLine());
        if(!S.equals(name))
        {
            ofile.println(name);
            ofile.println(fname);
            ofile.println(mname);
            ofile.println(address);
            ofile.println(Class);
            ofile.println(schNo);
            ofile.println(Subjects);
            ofile.println(dob);
            ofile.println(age);
            ofile.println(telephoneNo);
        }
        else flag=true;
    }

    if(!flag)
        System.out.println("Name not found");

    ifile.close();
    ofile.close();
    /*
    Same Logic as that of modify
     */
    replace("temp","studentRecords.txt");
    System.out.println("Data Deleted!");
    br.readLine();
    showMenu();
}
```

Data files

This program creates and makes use of a .txt file to store names, standards, age, school number, subjects and activities of students. This is a separate file which is stored in the computer in the program file. The .txt file is accused by the program while running.
The file named 'studentRecords.txt' stores essential information pertaining to the solution such as:

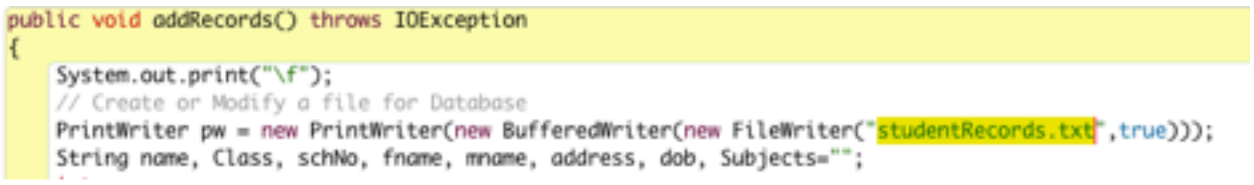
- Name
- Father's Name
- Mother's Name
- Address
- Class
- School Number
- Subjects
- Date of Birth
- Age
- Tel. No.



All this information is stored in plain text in the order given above. The program is able to read and write on this .txt file using FileWriter, PrintWriter and FileReader library functions. It is also possible to open the file using textedit or notepad and manually add the information but it is required in the above order for the program to read the data correctly. Thus it is possible to import already electronic data onto this database by simply placing it in the file in the given syntax. This will save a large amount of time as there will be no need to re-enter already available data. Again since the values are stored in plain text, the data is saved line by line in the exact order as given above (i.e. name, school number, age, class, subjects etc.). In developing this solution, it is possible to include data encryption in this format to secure the data and ensure no unauthorized alteration.

A second .txt file is created while switching values called temp.txt. This file is used to store temporarily the location of certain values when they are being edited. This is also called suing FileReader and PrintWriter functions (these are part of the imported java.io library).

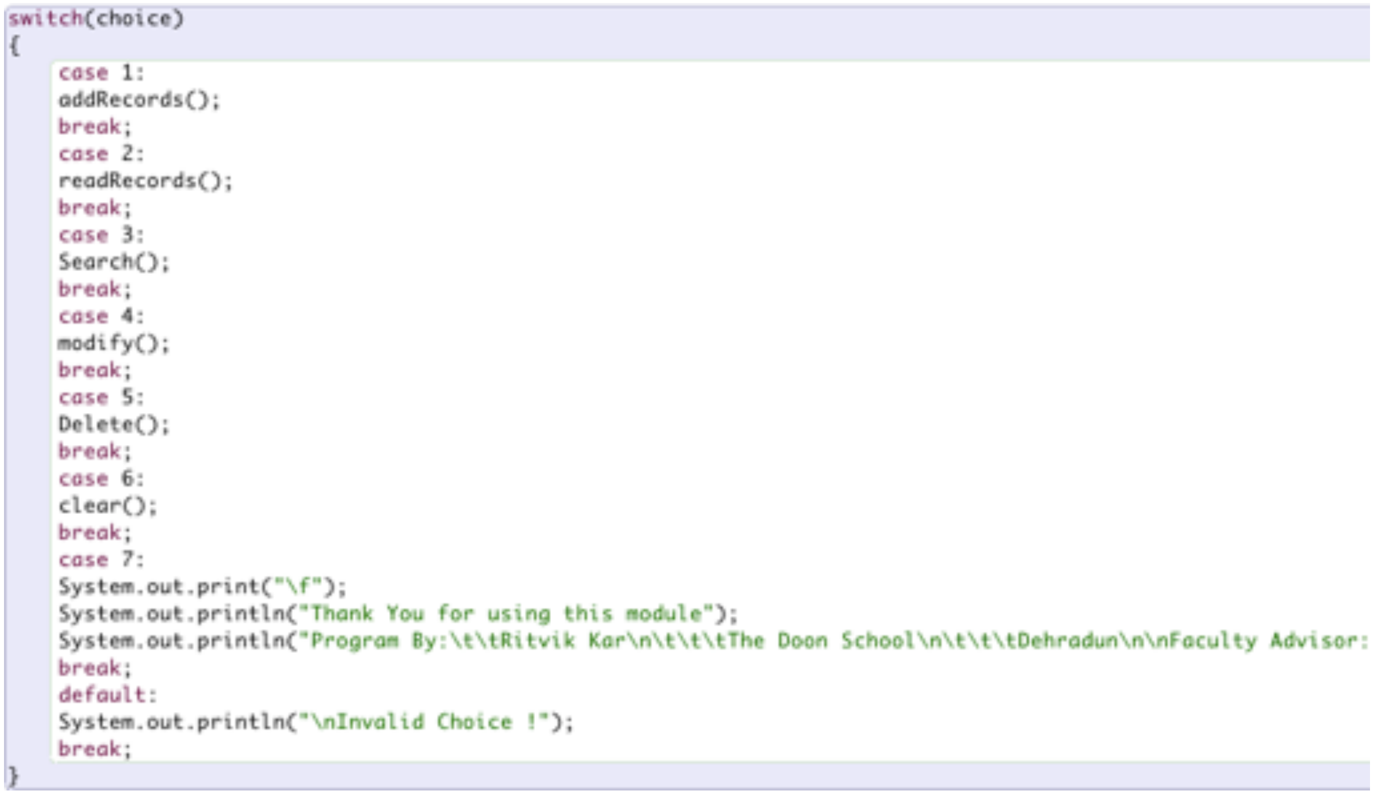
(Below: `write()` *function using the studentRecords to read the values using the BufferedReader library function)*



```
public void addRecords() throws IOException
{
    System.out.print("\f");
    // Create or Modify a file for Database
    PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter("studentRecords.txt",true)));
    String name, Class, schNo, fname, mname, address, dob, Subjects="";
```

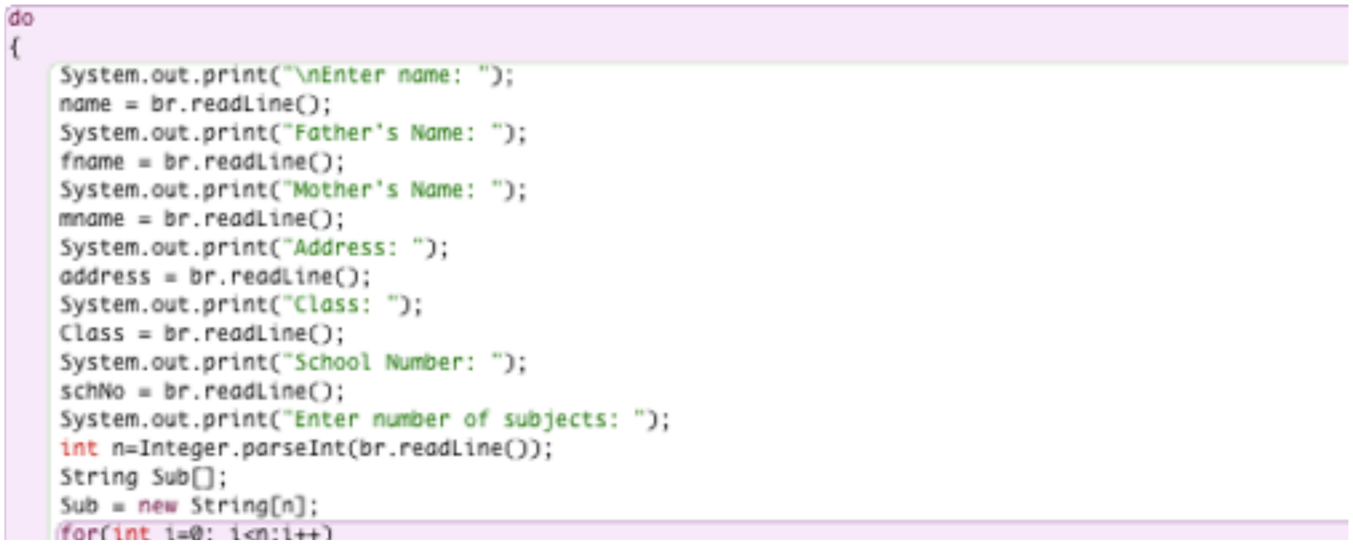
Conditional Statements

The solution also extensively makes use of conditional statements and other logical statements and operators.



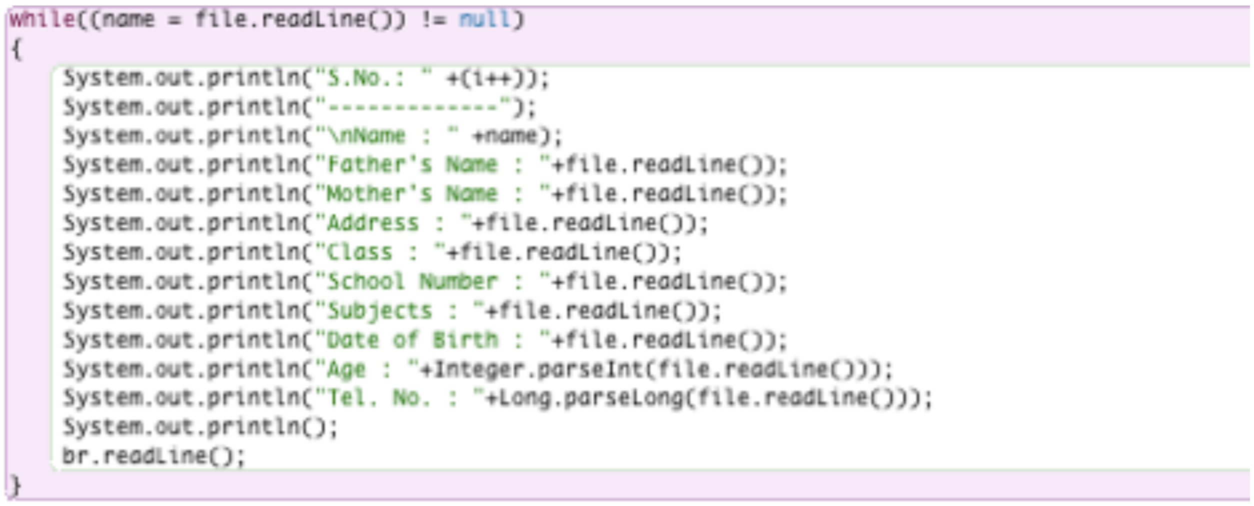
```
switch(choice)
{
    case 1:
    addRecords();
    break;
    case 2:
    readRecords();
    break;
    case 3:
    Search();
    break;
    case 4:
    modify();
    break;
    case 5:
    Delete();
    break;
    case 6:
    clear();
    break;
    case 7:
    System.out.print("\f");
    System.out.println("Thank You for using this module");
    System.out.println("Program By:\t\tRitvik Kar\n\t\t\tThe Doon School\n\t\t\tDehradun\n\nFaculty Advisor:
    break;
    default:
    System.out.println("\nInvalid Choice !");
    break;
}
```

The first example of this usage of conditional statements is in the form of 'switch-case' operators in the menu. The statement will allow the usr to choose which action to proceed with and calling the prevalent function thereafter.



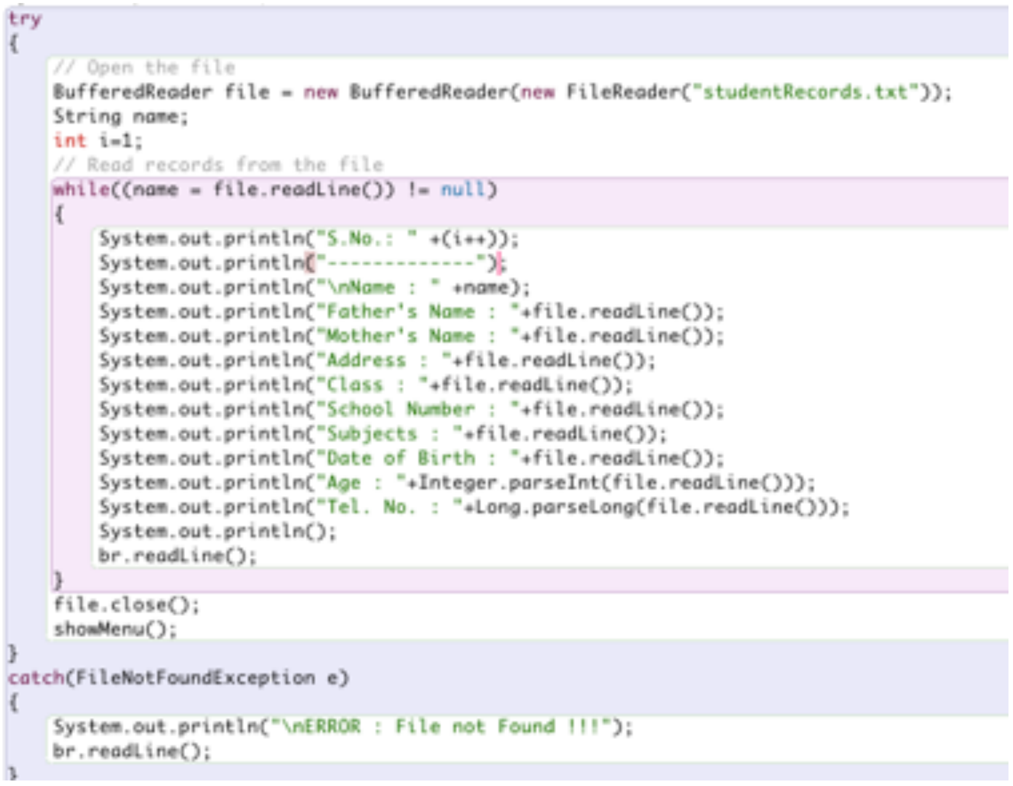
```
do
{
    System.out.print("\nEnter name: ");
    name = br.readLine();
    System.out.print("Father's Name: ");
    fname = br.readLine();
    System.out.print("Mother's Name: ");
    mname = br.readLine();
    System.out.print("Address: ");
    address = br.readLine();
    System.out.print("Class: ");
    Class = br.readLine();
    System.out.print("School Number: ");
    schNo = br.readLine();
    System.out.print("Enter number of subjects: ");
    int n=Integer.parseInt(br.readLine());
    String Sub[];
    Sub = new String[n];
    for(int i=0; i<n;i++)
```

Conditional statements in the form of loops such as ‘do-while’, where the loop will add student data to the database by prompting the user to continue. If an input of ‘Y’ or ‘y’ is received then the loop continues, else exiting and returning to the main menu.



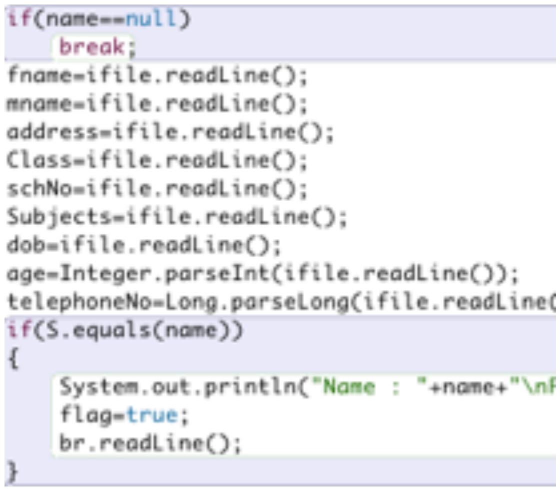
```
while((name = file.readLine()) != null)
{
    System.out.println("S.No.: " +(i++));
    System.out.println("-------------");
    System.out.println("\nName : " +name);
    System.out.println("Father's Name : "+file.readLine());
    System.out.println("Mother's Name : "+file.readLine());
    System.out.println("Address : "+file.readLine());
    System.out.println("Class : "+file.readLine());
    System.out.println("School Number : "+file.readLine());
    System.out.println("Subjects : "+file.readLine());
    System.out.println("Date of Birth : "+file.readLine());
    System.out.println("Age : "+Integer.parseInt(file.readLine()));
    System.out.println("Tel. No. : "+Long.parseLong(file.readLine()));
    System.out.println();
    br.readLine();
}
```

Other loops such as ‘while’ loops are also utilized over the course of this solution. The while loop is set to carry on infinitely as long as conditions remain ‘true’ which will only break if(N=Null) statement is fulfilled. This would imply that the loop will automatically stop at the end of the file and not cause an OutOfBoundsException leading to an error. Additionally, ‘try’ and ‘catch’ exception handling has been used in the solution. Below is a screen shot of the procedure of this exception handling:



```
try
{
    // Open the file
    BufferedReader file = new BufferedReader(new FileReader("studentRecords.txt"));
    String name;
    int i=1;
    // Read records from the file
    while((name = file.readLine()) != null)
    {
        System.out.println("S.No.: " +(i++));
        System.out.println("-------------");
        System.out.println("\nName : " +name);
        System.out.println("Father's Name : "+file.readLine());
        System.out.println("Mother's Name : "+file.readLine());
        System.out.println("Address : "+file.readLine());
        System.out.println("Class : "+file.readLine());
        System.out.println("School Number : "+file.readLine());
        System.out.println("Subjects : "+file.readLine());
        System.out.println("Date of Birth : "+file.readLine());
        System.out.println("Age : "+Integer.parseInt(file.readLine()));
        System.out.println("Tel. No. : "+Long.parseLong(file.readLine()));
        System.out.println();
        br.readLine();
    }
    file.close();
    showMenu();
}
catch(FileNotFoundException e)
{
    System.out.println("\nERROR : File not Found !!!");
    br.readLine();
}
```

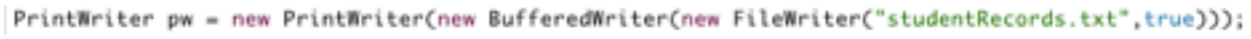
The conditional statements also utilize boolean logic. An example is visible in the search() function where the loop uses flags to decide whether the student is found to not. This in turn determines when the loop will eventually stop. Beside is a screenshot of this code:



```
if(name==null)
    break;
fname=ifile.readLine();
mname=ifile.readLine();
address=ifile.readLine();
Class=ifile.readLine();
schNo=ifile.readLine();
Subjects=ifile.readLine();
dob=ifile.readLine();
age=Integer.parseInt(ifile.readLine());
telephoneNo=Long.parseLong(ifile.readLine(
if(S.equals(name))
{
    System.out.println("Name : "+name+"\nF
    flag=true;
    br.readLine();
}
```

Library Functions

The solution also makes use of a plethora of library functions for reading and writing from and on the file. An example of this usage is in the append() function where it is seen (after the importing of the java.io library) that a BufferedReader initialized as 'br' is used to accept from the user, while the BufferedReader 'ifile' (i.e. input file) is used to accept values fro the datafile txt. The PrintWriter instance initialized as 'ofile' is used yo implement the FileWriter instance to write values in the 'temp' file. This is actually the process of file handling.



```
PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter("studentRecords.txt",true)));
```

Also lines such as System.out.println(""); are used to print instructions on screen while reading values through readLine (called using BufferedReader instance 'br'). Integer or doubles values are often parsed using library functions, example scn=Integer.parseInt(br.readLine());.

At all times the 'throws IOException' function library is used to catch errors of unexpected input.