Sachin Karki, Ritu Shrestha

Dr. Wen Xu

CSCI 3053-01

March 20th, 2021

Project 1: Array Based Structure

Project Outline & Introduction

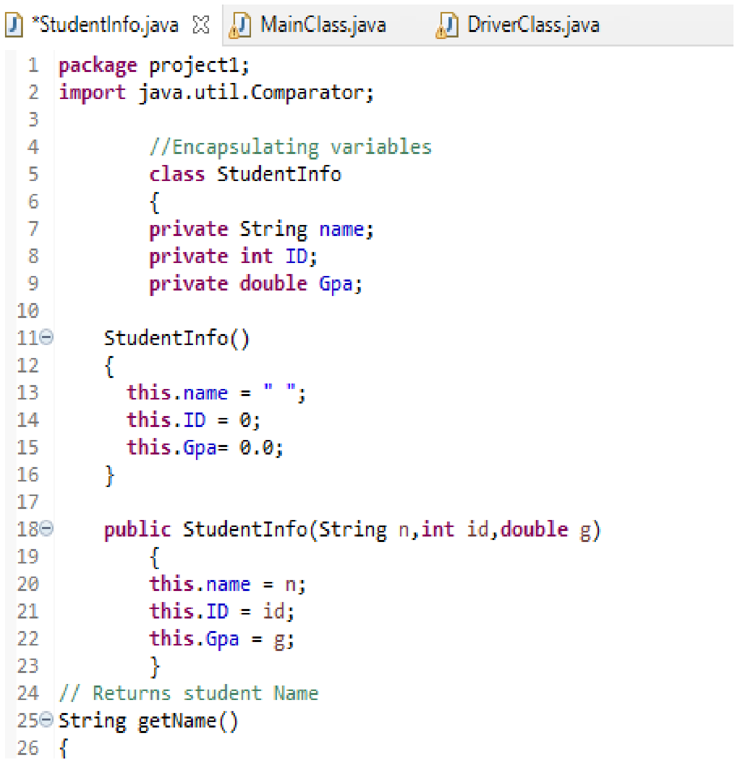
For this project, we created a program that relied on array-based structure as provided in the instructions. *The Academic Information Database* is a dynamic console-based program created in Java using sorted array-based structure method. As the name suggests, student information database stores administrative related student information like name, student ID, GPA. The functionality of the program is much more than just storing the information. It can display, update, delete and view all the records within the program.

Program Layout

The overall layout of the program is the result of usage of different functions within the Java programming language. The program utilizes various functionalities and technical options provided by the Java and Eclipse IDE. A brief elemental breakdown of those functionalities are described below.

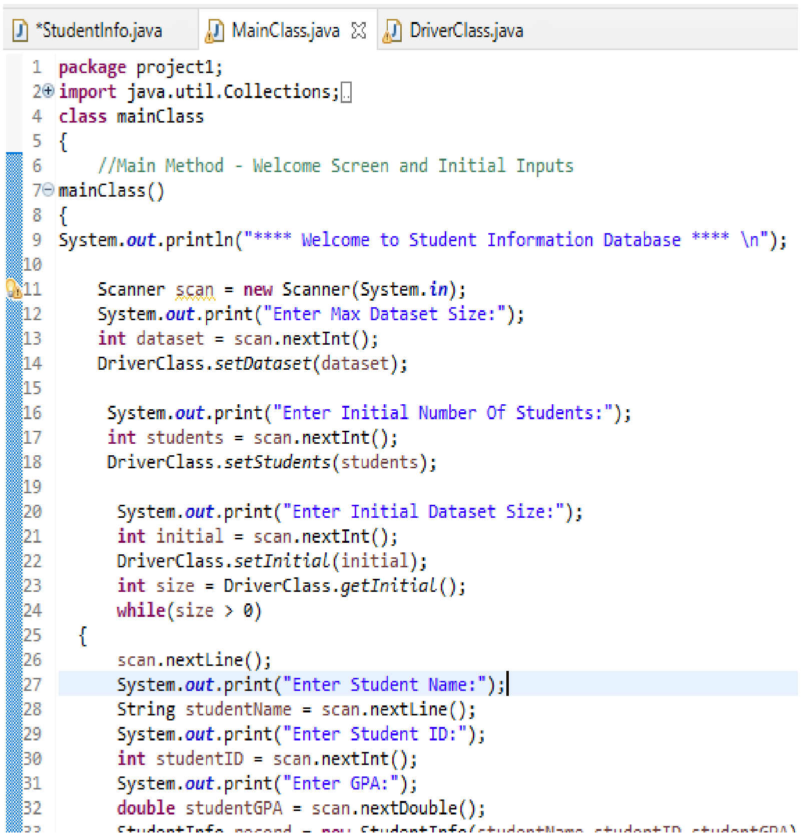
1. ***Variables:*** To begin with, a great deal of variables were used to differentiate several variables that were used within the code. Most notably variables like “name, ID, Gpa” were used to associate student’s name, ID, and grade points average respectively. These four variables would also rely on string, integer and double datatypes respectively to configure and assign them a value that they can handle.

2. ***Class and Methods***: The program utilizes three class for three specific purpose or say three functions in our case. A class named *StudentInfo* was created and used to encapsulate variables relating to student’s credentials like name, identification and their grade point averages score. Similarly, *DriverClass* was used to hold entry provided for the dataset from the ArrayList. This class also contains the method to sort and display all the records in the system. primarily there were two methods that were used within the program to divide the execution of different functions. Private and public class were helpful in achieving that. For an instance, `public class DriverClass` was used to pass and contain maximum and initial datasets size. In addition to this, a main class named `class mainClass` was used to pass the main method to execute the welcome screen and follow rest of the operations from there. Three screenshots are attached side-by-side below.



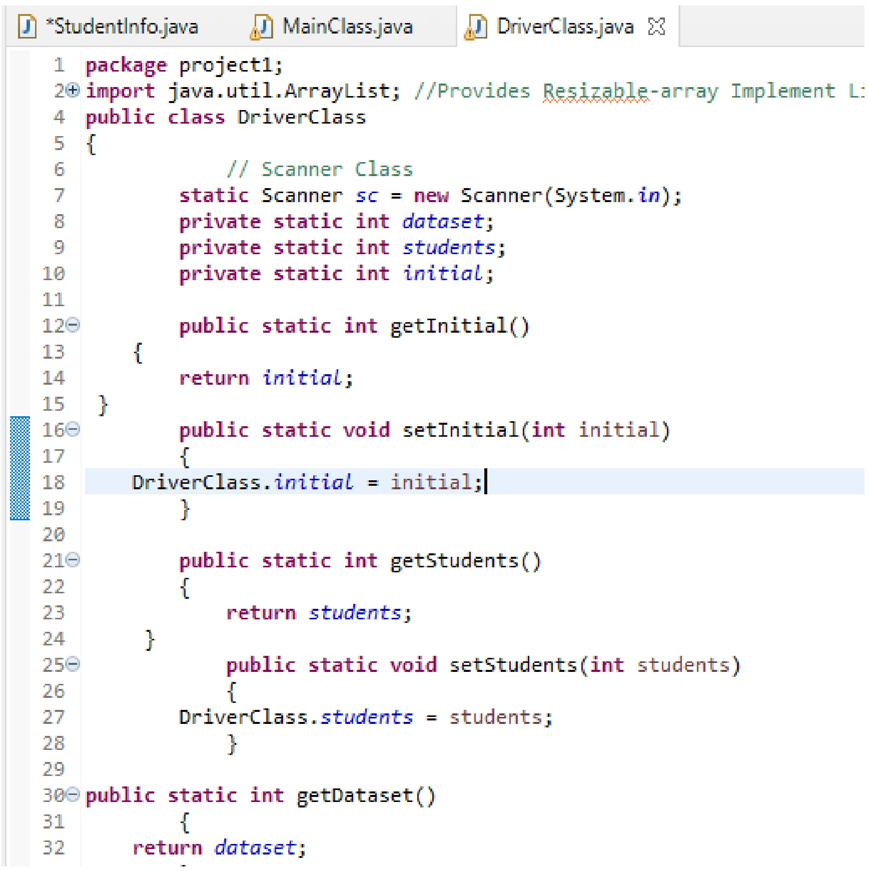
```java
package project1;
import java.util.Comparator;

        //Encapsulating variables
        class StudentInfo
        {
        private String name;
        private int ID;
        private double Gpa;

    StudentInfo()
    {
      this.name = " ";
      this.ID = 0;
      this.Gpa= 0.0;
    }

    public StudentInfo(String n,int id,double g)
        {
        this.name = n;
        this.ID = id;
        this.Gpa = g;
        }
// Returns student Name
String getName()
{
```

StudentInfo class



```java
package project1;
import java.util.Collections;
class mainClass
{
    //Main Method - Welcome Screen and Initial Inputs
mainClass()
{
System.out.println("**** Welcome to Student Information Database **** \n");

    Scanner scan = new Scanner(System.in);
    System.out.print("Enter Max Dataset Size:");
    int dataset = scan.nextInt();
    DriverClass.setDataset(dataset);

     System.out.print("Enter Initial Number Of Students:");
     int students = scan.nextInt();
     DriverClass.setStudents(students);

      System.out.print("Enter Initial Dataset Size:");
      int initial = scan.nextInt();
      DriverClass.setInitial(initial);
      int size = DriverClass.getInitial();
      while(size > 0)
  {
      scan.nextLine();
      System.out.print("Enter Student Name:");
      String studentName = scan.nextLine();
      System.out.print("Enter Student ID:");
      int studentID = scan.nextInt();
      System.out.print("Enter GPA:");
      double studentGPA = scan.nextDouble();
      StudentInfo record = new StudentInfo(studentName,studentID,studentGPA)
```

MainClass

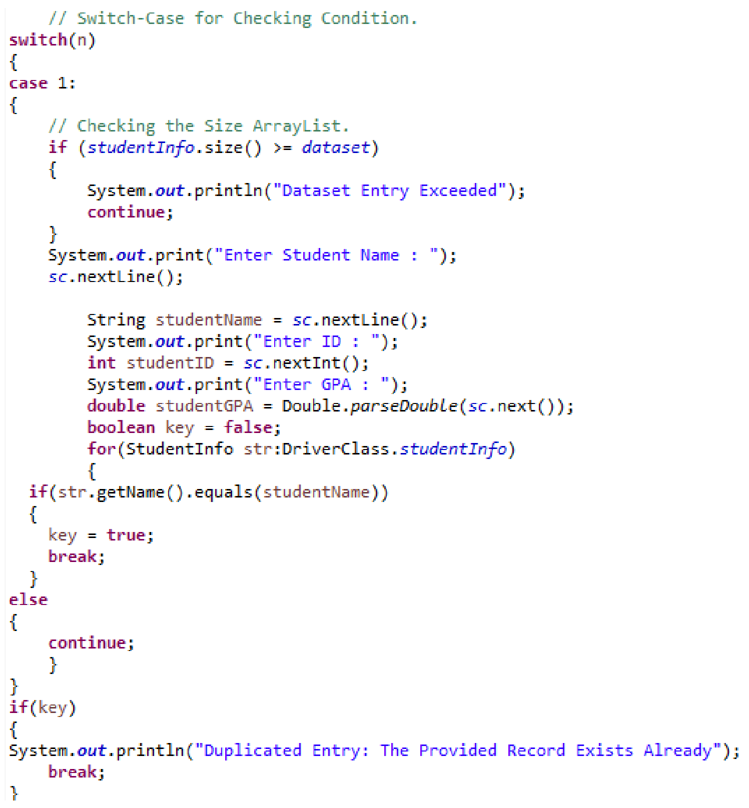


```java
package project1;
import java.util.ArrayList; //Provides Resizable-array Implement L
public class DriverClass
{
            // Scanner Class
        static Scanner sc = new Scanner(System.in);
        private static int dataset;
        private static int students;
        private static int initial;

        public static int getInitial()
    {
        return initial;
 }
        public static void setInitial(int initial)
        {
    DriverClass.initial = initial;
        }

        public static int getStudents()
        {
            return students;
     }
            public static void setStudents(int students)
            {
        DriverClass.students = students;
            }

public static int getDataset()
        {
    return dataset;
```

DriverClass

3. ***Condition Checking (If/Else):*** One of the good usages of if/else condition checking was applied in DriverClass to return the index value in array list. This would help in selecting the right part of the operation based on the prior user prompt. Futhermore, if/else were also used alongside of switch case to execute specific operation.

4. ***Switch-Case:*** Throughout the program switch-case were effectively used to make selections for different operations the program was able to make. Switch-case were applied in places where users would feed an input and based on the user prompt it would select the required operation. Pressing numerical values like 1,2,3,4,5,6 would trigger assigned method from the driver and main class. For an instance, pressing 1 would trigger insert function where you would add student information and similarly pressing 2 would trigger view where you can see the entered student information. A snippet of switch-case in the program is attached below.



```
    // Switch-Case for Checking Condition.
switch(n)
{
case 1:
{
    // Checking the Size ArrayList.
    if (studentInfo.size() >= dataset)
    {
        System.out.println("Dataset Entry Exceeded");
        continue;
    }
    System.out.print("Enter Student Name : ");
    sc.nextLine();

        String studentName = sc.nextLine();
        System.out.print("Enter ID : ");
        int studentID = sc.nextInt();
        System.out.print("Enter GPA : ");
        double studentGPA = Double.parseDouble(sc.next());
        boolean key = false;
        for(StudentInfo str:DriverClass.studentInfo)
        {
  if(str.getName().equals(studentName))
  {
    key = true;
    break;
  }
else
{
    continue;
    }
}
if(key)
{
System.out.println("Duplicated Entry: The Provided Record Exists Already");
    break;
}
```

Switch-Case

5. ***Loops:*** Every programming of this type of extent and scope uses loop and this is true in our program as well. *For* and *while* loops were used in places to repeat the process of searching through array list and display the sorted list within the mainClass and DriverClass. This would help out in retrieving the record that has been in the system and also displaying the stored records in an organized alignment in an alphabetical order.

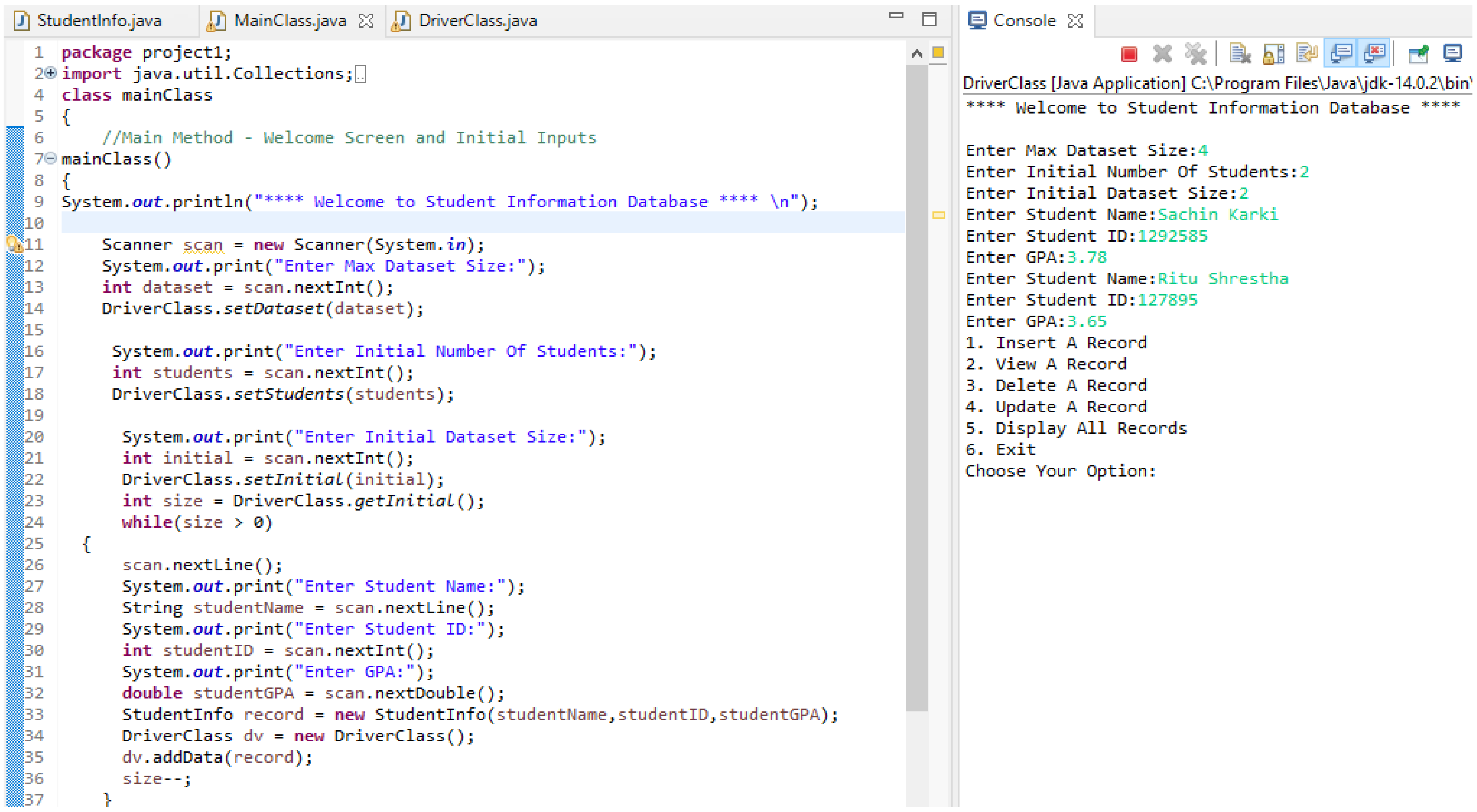
Operations Implemented

A total of six operations were implemented in the program that basically performs adding, seeing, removing and changing the contents of the information. These operations can be toggled from a keyboard with a numerical input from the user. For an instance, the operation of inserting a student information can be done by pressing 1 on the keypad and the users will be given additional instructions to test or check it. Listed below are all the operations that was implemented in the program

1. Insert: adds information
2. View (Fetch & Display): to see one specific information
3. Delete: to remove a certain information
4. Update: changing the content of one specific record
5. Display All: shows all record in alphabetical arrangement
6. Exit: terminates all operation

Execution and Testing Results

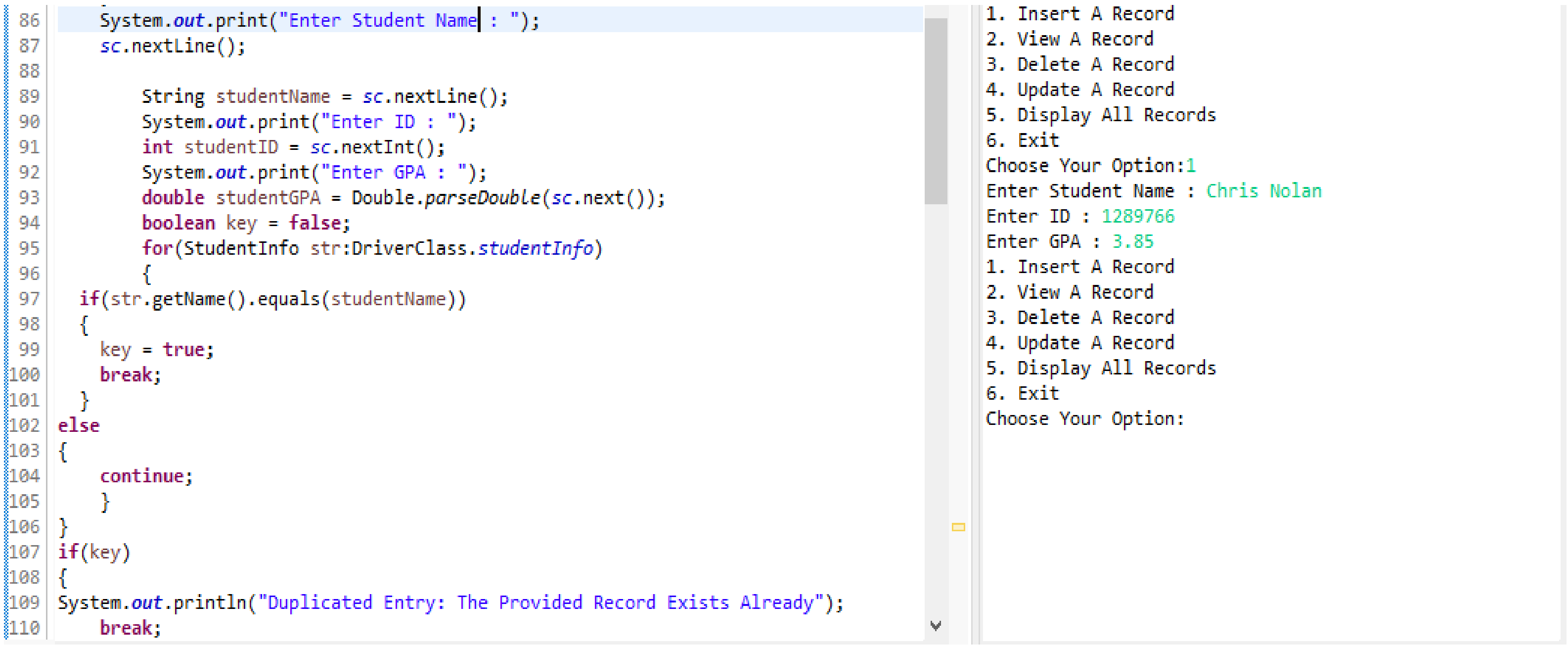
#Main Method: The program will begin with a Welcome screen created by the mainClass where users will be asked to enter the maximum number of datasets and number initial students they would like to add in the system. To begin with, for testing we used 4 sample of student information, two of them our own credentials and two based on fictional characters. Here's a picture of the relating coding and its output.



```
package project1;
import java.util.Collections;
class mainClass
{
    //Main Method - Welcome Screen and Initial Inputs
mainClass()
{
System.out.println("**** Welcome to Student Information Database **** \n");

    Scanner scan = new Scanner(System.in);
    System.out.print("Enter Max Dataset Size:");
    int dataset = scan.nextInt();
    DriverClass.setDataset(dataset);

     System.out.print("Enter Initial Number Of Students:");
     int students = scan.nextInt();
     DriverClass.setStudents(students);

      System.out.print("Enter Initial Dataset Size:");
      int initial = scan.nextInt();
      DriverClass.setInitial(initial);
      int size = DriverClass.getInitial();
      while(size > 0)
 {
      scan.nextLine();
      System.out.print("Enter Student Name:");
      String studentName = scan.nextLine();
      System.out.print("Enter Student ID:");
      int studentID = scan.nextInt();
      System.out.print("Enter GPA:");
      double studentGPA = scan.nextDouble();
      StudentInfo record = new StudentInfo(studentName,studentID,studentGPA);
      DriverClass dv = new DriverClass();
      dv.addData(record);
      size--;
    }
```

```
DriverClass [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\
**** Welcome to Student Information Database ****

Enter Max Dataset Size:4
Enter Initial Number Of Students:2
Enter Initial Dataset Size:2
Enter Student Name:Sachin Karki
Enter Student ID:1292585
Enter GPA:3.78
Enter Student Name:Ritu Shrestha
Enter Student ID:127895
Enter GPA:3.65
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:
```

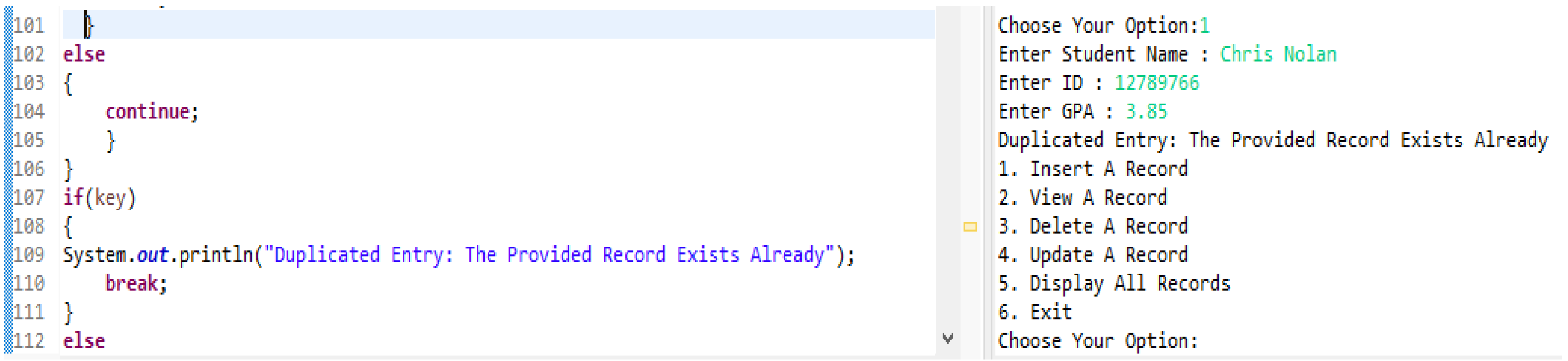
1. ***Insert a Record:*** Pressing '1' will toggle the 'insert a record' option which will prompt user to enter student's name, ID number and their GPA. New record will be stored in the program and in the case of entering a repeated information, it will prompt a notification about the duplicated entry of that data. Two examples have been demonstrated in the picture below.



```java
    System.out.print("Enter Student Name : ");
    sc.nextLine();

        String studentName = sc.nextLine();
        System.out.print("Enter ID : ");
        int studentID = sc.nextInt();
        System.out.print("Enter GPA : ");
        double studentGPA = Double.parseDouble(sc.next());
        boolean key = false;
        for(StudentInfo str:DriverClass.studentInfo)
        {
  if(str.getName().equals(studentName))
  {
    key = true;
    break;
  }
else
{
    continue;
    }
}
if(key)
{
System.out.println("Duplicated Entry: The Provided Record Exists Already");
    break;
```

```
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:1
Enter Student Name : Chris Nolan
Enter ID : 1289766
Enter GPA : 3.85
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:
```

Inserting a new record

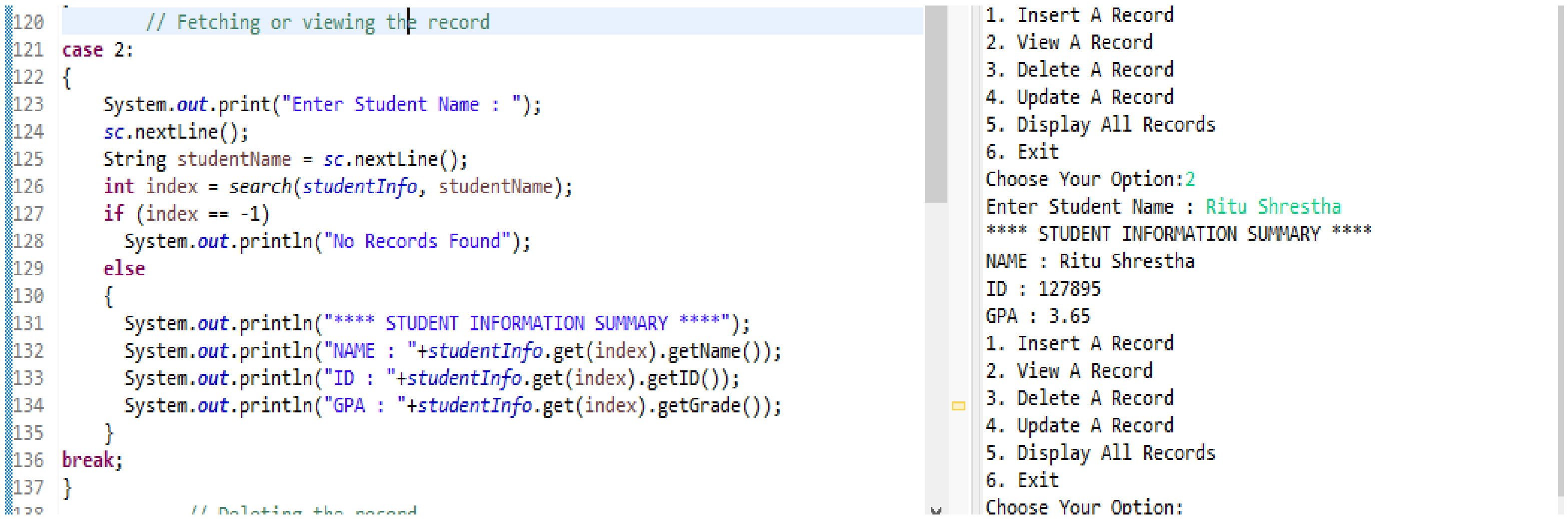


```java
  }
else
{
    continue;
    }
}
if(key)
{
System.out.println("Duplicated Entry: The Provided Record Exists Already");
    break;
}
else
```

```
Choose Your Option:1
Enter Student Name : Chris Nolan
Enter ID : 12789766
Enter GPA : 3.85
Duplicated Entry: The Provided Record Exists Already
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:
```

Inserting an existing record

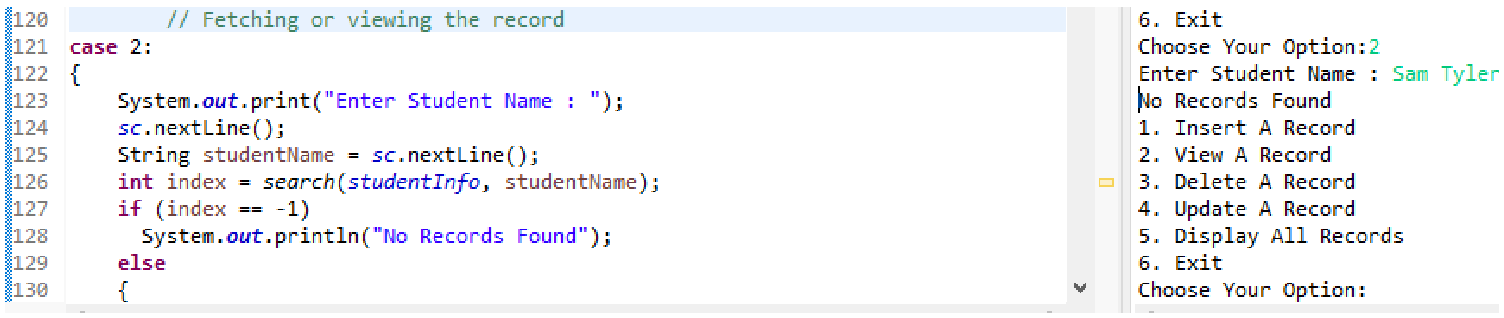
2. ***View a Record:*** Pressing ‘2’ will trigger ‘view a record’ option which will prompt user to provide the name of a student. Upon providing a valid and existing name, users will be returned the details of student information and in case of an invalid name, users will be returned with about the absence of such record. The demonstrations are pictured below.



```java
        // Fetching or viewing the record
case 2:
{
    System.out.print("Enter Student Name : ");
    sc.nextLine();
    String studentName = sc.nextLine();
    int index = search(studentInfo, studentName);
    if (index == -1)
      System.out.println("No Records Found");
    else
    {
      System.out.println("**** STUDENT INFORMATION SUMMARY ****");
      System.out.println("NAME : "+studentInfo.get(index).getName());
      System.out.println("ID : "+studentInfo.get(index).getID());
      System.out.println("GPA : "+studentInfo.get(index).getGrade());
    }
break;
}
            // Deleting the record
```

```
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:2
Enter Student Name : Ritu Shrestha
**** STUDENT INFORMATION SUMMARY ****
NAME : Ritu Shrestha
ID : 127895
GPA : 3.65
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:
```

Viewing a Record



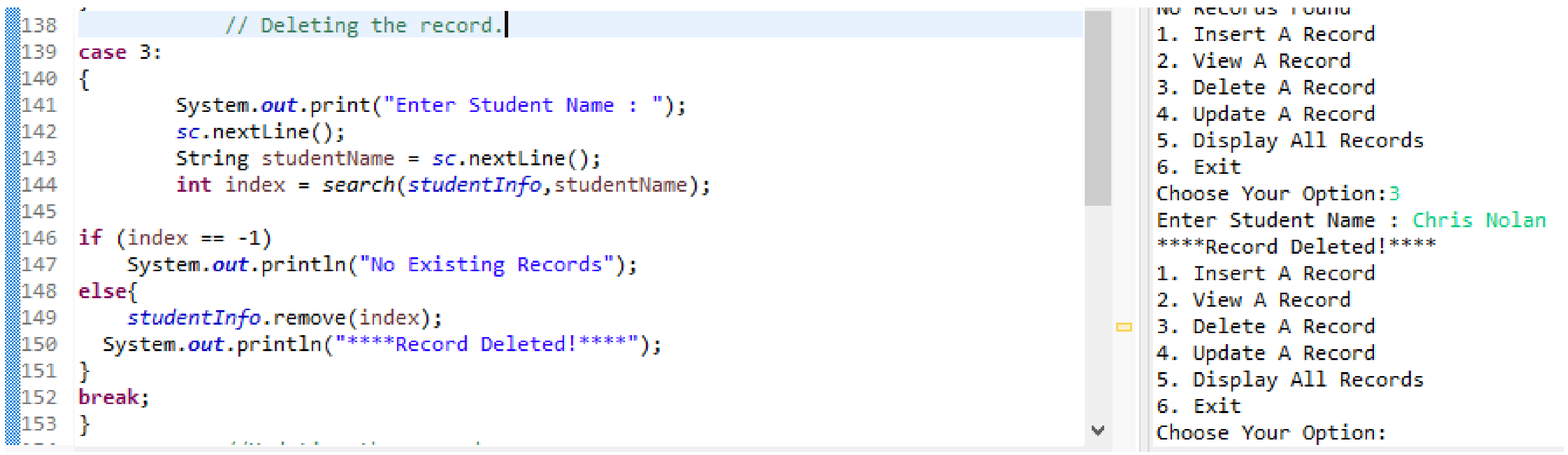
```java
        // Fetching or viewing the record
case 2:
{
    System.out.print("Enter Student Name : ");
    sc.nextLine();
    String studentName = sc.nextLine();
    int index = search(studentInfo, studentName);
    if (index == -1)
      System.out.println("No Records Found");
    else
    {
```

```
6. Exit
Choose Your Option:2
Enter Student Name : Sam Tyler
No Records Found
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:
```

Viewing a non-existing record

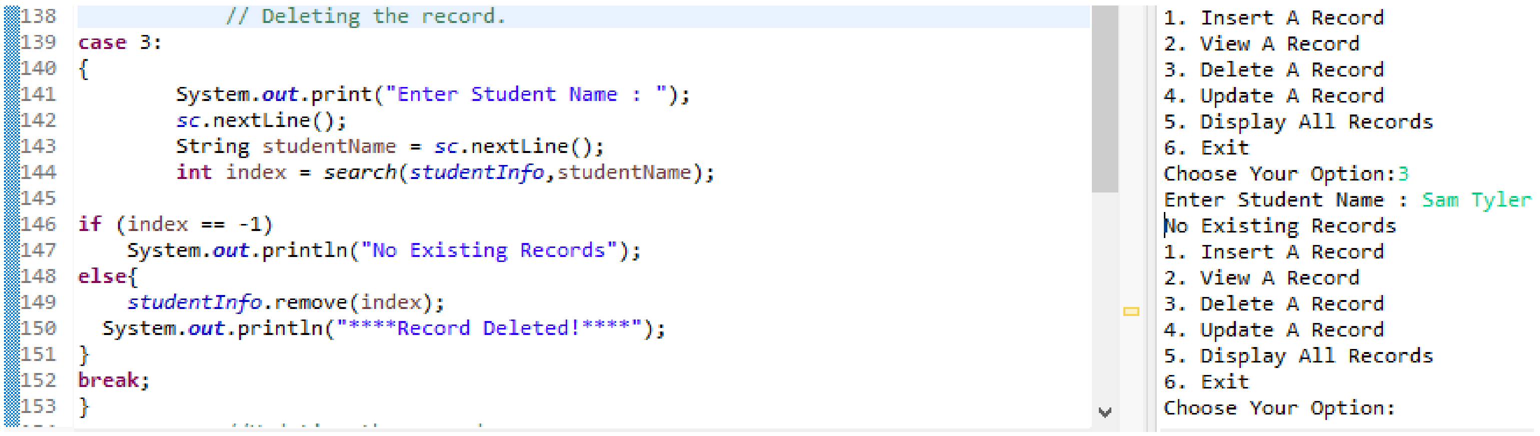
3. ***Deleting a Record:*** Pressing ‘3” will trigger ‘Delete a Record’ option which will ask user for the name of a student. Entering a existing name will erase the information associated with that name; entering a wrong name will prompt user about absence of such record in the system. Two demonstrations are given below.

Deleting a student information



```java
            // Deleting the record.
case 3:
{
        System.out.print("Enter Student Name : ");
        sc.nextLine();
        String studentName = sc.nextLine();
        int index = search(studentInfo,studentName);

if (index == -1)
    System.out.println("No Existing Records");
else{
    studentInfo.remove(index);
  System.out.println("****Record Deleted!****");
}
break;
}
```

```
No Records Found
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:3
Enter Student Name : Chris Nolan
****Record Deleted!****
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:
```

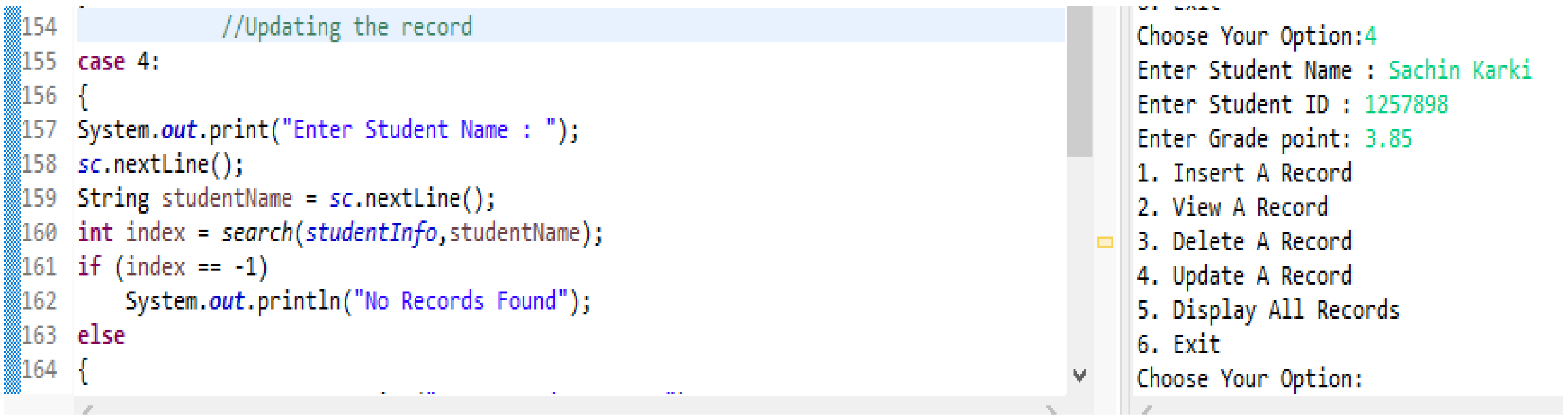


```
        // Deleting the record.
case 3:
{
        System.out.print("Enter Student Name : ");
        sc.nextLine();
        String studentName = sc.nextLine();
        int index = search(studentInfo,studentName);

if (index == -1)
    System.out.println("No Existing Records");
else{
    studentInfo.remove(index);
  System.out.println("****Record Deleted!****");
}
break;
}
```

```
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:3
Enter Student Name : Sam Tyler
No Existing Records
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:
```

Deleting a non-existent information.

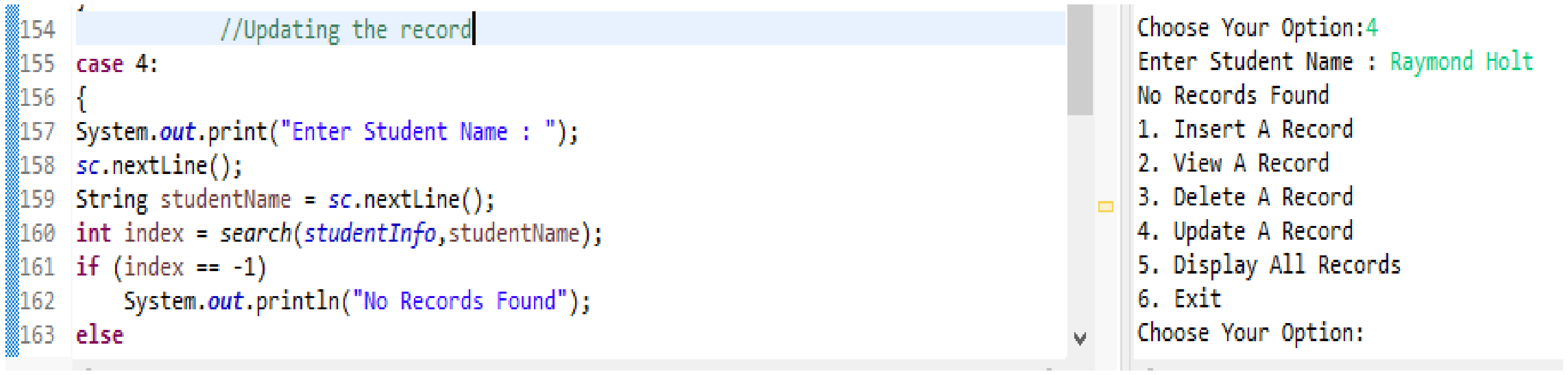
4. ***Update a Record:*** Pressing '4' will toggle to 'update a record' option, where users can update, ID and the GPA of a student if provided the right name as input. Feeding a wrong name as an input will prompt an error similar as to that of deletion process. A couple of demonstration is pictured below.



```
        //Updating the record
case 4:
{
System.out.print("Enter Student Name : ");
sc.nextLine();
String studentName = sc.nextLine();
int index = search(studentInfo,studentName);
if (index == -1)
    System.out.println("No Records Found");
else
{
```

```
Choose Your Option:4
Enter Student Name : Sachin Karki
Enter Student ID : 1257898
Enter Grade point: 3.85
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:
```

Updating a record

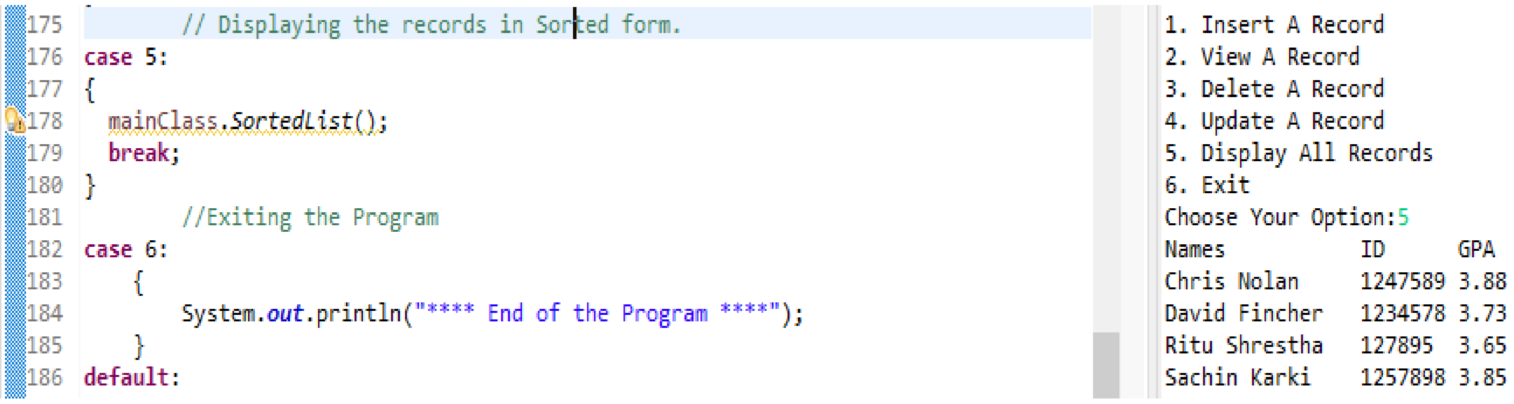


```
        //Updating the record
case 4:
{
System.out.print("Enter Student Name : ");
sc.nextLine();
String studentName = sc.nextLine();
int index = search(studentInfo,studentName);
if (index == -1)
    System.out.println("No Records Found");
else
```

```
Choose Your Option:4
Enter Student Name : Raymond Holt
No Records Found
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:
```

Updating a non-existing record

5. ***Display All Records:*** Pressing ‘5’ would toggle the ‘display all records’ option which displays all the records that is in the system. This option would list all the student information in the program in an alphabetical arrangement. A demonstration is pictured below.

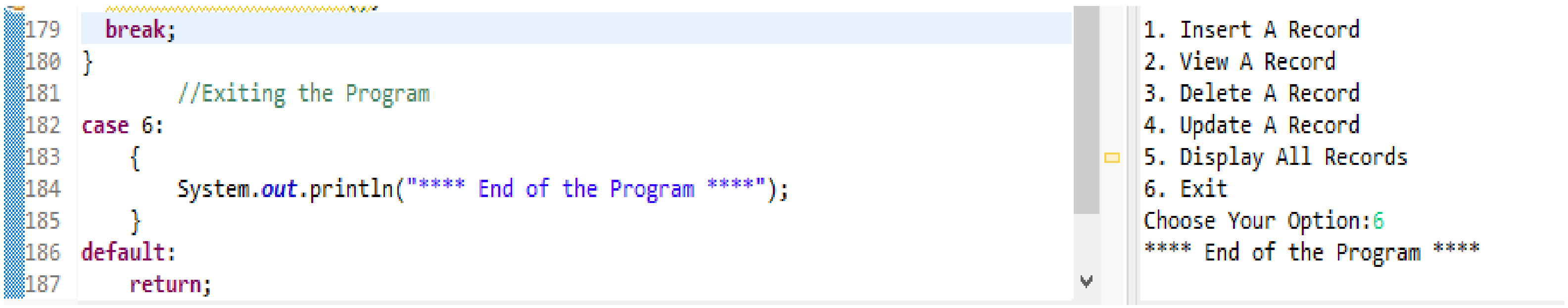


```
        // Displaying the records in Sorted form.
case 5:
{
  mainClass.SortedList();
  break;
}
        //Exiting the Program
case 6:
    {
        System.out.println("**** End of the Program ****");
    }
default:
```

```
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:5
Names           ID      GPA
Chris Nolan     1247589 3.88
David Fincher   1234578 3.73
Ritu Shrestha   127895  3.65
Sachin Karki    1257898 3.85
```

Displaying all records.

6. ***Exit:*** Pressing ‘6’ would toggle ‘exit’ option that terminates the program. This will end all operation and no any other operation can be selected. A demonstration is pictured below.



```
  break;
}
        //Exiting the Program
case 6:
    {
        System.out.println("**** End of the Program ****");
    }
default:
    return;
```

```
1. Insert A Record
2. View A Record
3. Delete A Record
4. Update A Record
5. Display All Records
6. Exit
Choose Your Option:6
**** End of the Program ****
```

Exiting the program

Summary

As described earlier, this is a console-based program so users are able to dynamically update within the console however there are few restrictions to this program as well. First of all, the information that are fed as an input will not be stored into the memory meaning the program is not able to remember the last information soon as you terminate or restart the program. Second, feeding a wrong type of datatype could trigger errors. For an instance, feeding integer value into decimal (double) form of datatype could return an error. Feeding integer (numeric) value in places of string or char (characters and letters) can also trigger errors. It is essential for smooth running of the program to provide right set of input. To sum up, the program would work fine assuming all the input datatype are provided as expected.