|  |  |
| --- | --- |
| **AP Computer Science** | **TextLab11 Java Assignment** |
| **The Student Records Algorithm Program** | **80, 90, 100 & 110 Point Versions** |
| **Assignment Purpose:**  This program requires knowledge of various algorithms like searching and sorting. | |

Write a program that reads in a text file called **"students2.dat"**. (This is a similar, but different file from the one you used in a previous lab) The program will need to store all of the data (**Name**, **ID#**, **Age**, and **GPA**) into an array of **Person** records. This array – called **Student** – will be sorted and displayed by **GPA**, by **Age**, and finally by **ID#**. The user will then enter the **ID#** of one of the students. The individual **Person** record for that student will be displayed, and then that student will be deleted from the array.

The focus of this assignment is *Algorithms* and not *File Handling*. A file is used to make entering all of the information more efficient. The method to read in this file is provided to allow you to focus on the algorithms.

|  |  |
| --- | --- |
| **TextLab11 Student Version** | **Do not copy this file, which is provided.** |
| // TextLab11st.java  // The Student Records Algorithm Program  // This is the student, starting version of the TextLab11 assignment.  import java.io.\*;  import java.util.\*;  import java.text.DecimalFormat;    public class TextLab11st  {  public static void main(String args[]) throws IOException  {  List studentArray = new List(100);  studentArray.getList();  studentArray.display("UNSORTED LIST OF STUDENTS");  studentArray.pause();    studentArray.gpaSort();  studentArray.display("STUDENTS SORTED IN DESCENDING ORDER BY GPA");  studentArray.pause();    studentArray.ageSort();  studentArray.display("STUDENTS SORTED IN ASCENDING ORDER BY AGE");  studentArray.pause();    studentArray.idSort();  studentArray.display("STUDENTS SORTED IN ASCENDING ORDER BY ID#");  studentArray.pause();    // int studentID = getID();  // int index = studentArray.search(studentID);  //  // if (index == -1)  // System.out.println("There is no student with an ID# of "+studentID+".\n");  // else  // {  // studentArray.displayStudent(index); // displays the information for the found student  // studentArray.delete(index); // remove the same student from the array  // studentArray.display("STUDENTS SORTED IN ASCENDING ORDER BY ID# WITHOUT STUDENT# "+studentID);  // studentArray.pause();  // }  }    public static int getID()  {  Scanner input = new Scanner(System.in);  System.out.print("\nEnter the 6-digit ID of the student. { 100000 - 999999 } --> ");  return input.nextInt();  }  }  class Person  {  public String name;  public int id;  public int age;  public double gpa;    Person(String n, int ID, int a,double g)  {  name = n;  id = ID;  age = a;  gpa = g;  }  }  class List  {  private Person student[]; // stores array elements  private int capacity; // actual array capacity  private int size; // number of elements in the array    public List(int c)  {  capacity = c;  size = 0;  student = new Person[capacity];  }  public void getList() throws IOException  {  FileReader inFile = new FileReader("students2.dat");  BufferedReader inStream = new BufferedReader(inFile);  String s1,s2,s3,s4;  int age, id;  double gpa;  int index = 0;  while( ((s1 = inStream.readLine()) != null) &&  ((s2 = inStream.readLine()) != null) &&  ((s3 = inStream.readLine()) != null) &&  ((s4 = inStream.readLine()) != null) )  {  String name = s1;  id = Integer.parseInt(s2);  age = Integer.parseInt(s3);  gpa = Double.parseDouble(s4);  student[index] = new Person(name,id,age,gpa);  index++;  }  inStream.close();  size = index;  }    private String spaces(String name)  {  int tab = 24 - name.length();  String temp = "";  for (int j = 1; j <= tab; j++)  temp += " ";  return temp;  }    public void display(String listInfo)  {  DecimalFormat output = new DecimalFormat("0.000");  System.out.println("\nDISPLAYING "+ listInfo);  System.out.println("\nStudent ID# Student Name Age GPA");  System.out.println("============================================================");    for (int k = 0; k < size; k++)  System.out.println(student[k].id + " "+student[k].name + spaces(student[k].name) +  student[k].age + " " + output.format(student[k].gpa));  }    public void pause()  {  Scanner input = new Scanner(System.in);  String dummy;  System.out.println("\nPress <Enter> to continue.");  dummy = input.nextLine();  }  public void displayStudent(int index)  {    }  private void swap(int x, int y)  {  }  public void gpaSort()  {  }  public void ageSort()  {  }    public void idSort()  {  }    // public int search(int studentID)  // {  // }    public void delete(int index)  {  // Precondition: "index" stores the index of a student object that exists in the "student" array.  // Postcondition: The student object at index "index" is removed from the "student" array.  // All other objects in the "student" array are unaffected.      }  } | |

**Provided Student Version Output Description**

This program will read in and display all of the data in the **students2.dat** file 4 times. The first time, the data will simply be in the order it was in – in the file. The other 3 are supposed to be sorted displays, but if you have not written the sorts yet, the program will just display the same list 4 times. **80-Point Version Specifics**

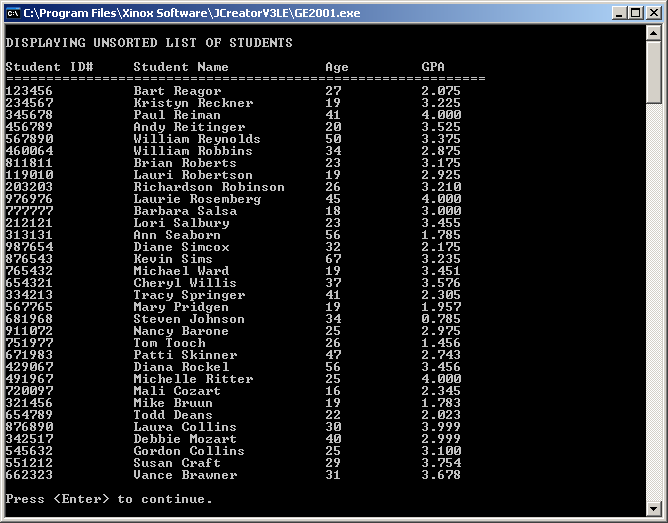
The 80-point version displays the data in the **Student** array in the original order, followed by 3 sorted displays that are sorted by **GPA** (descending), **Age** (ascending) and **ID#** (ascending).

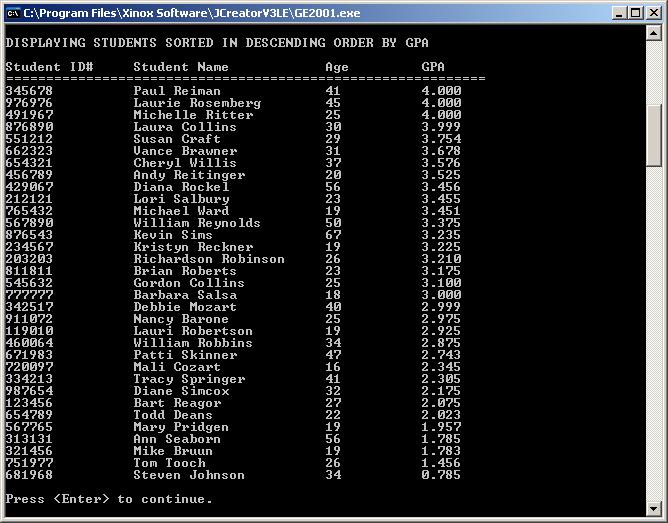
**80-Point Version Output**

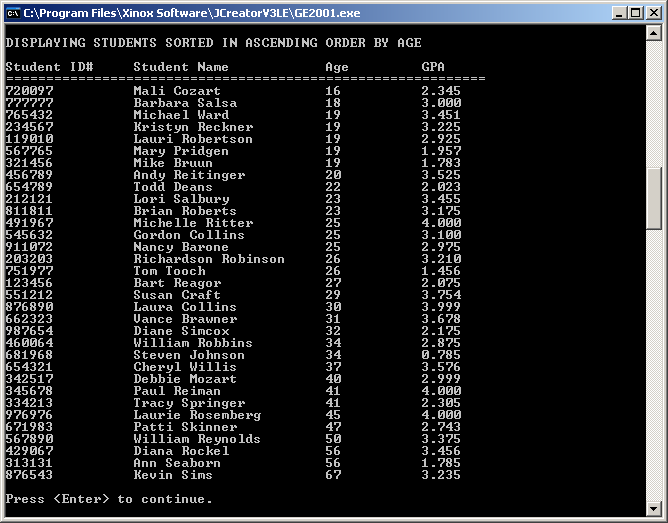
This program has a very long output. You will be shown separate screens which show individual stages of the output. Even though these are shown in separate windows, these are all part of the same output.

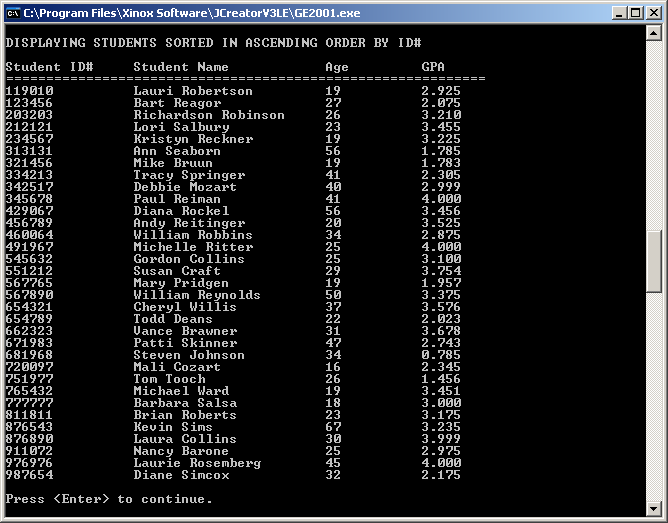
**NOTE**:

The output for the 80-point version is the same as the initial output of the 90, 100 and 110 point versions. To save paper, the windows below, and on the next 2 pages, will not be repeated for the other versions of this assignment.







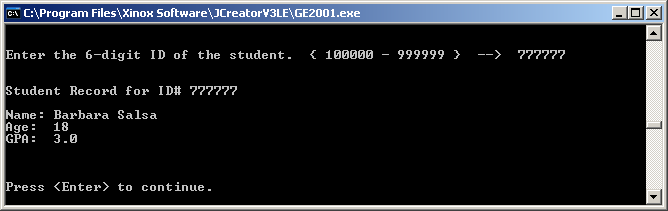


**90-Point Version Specifics**

The 90-point version requires everything from the 80-point version but adds a *linear search*. The user needs to be prompted for an **ID#** (use the provided **getID** method) and then the **search** method needs to find and return the **index** of the student with that **ID#**. Another method that needs to be written is **displayStudent**. This method uses the **index** you obtained as its parameter and then displays all of the information about the student at that particular **index**.

**90-Point Version Output**

The output will include everything you saw in the 80-point version output, plus what is below:

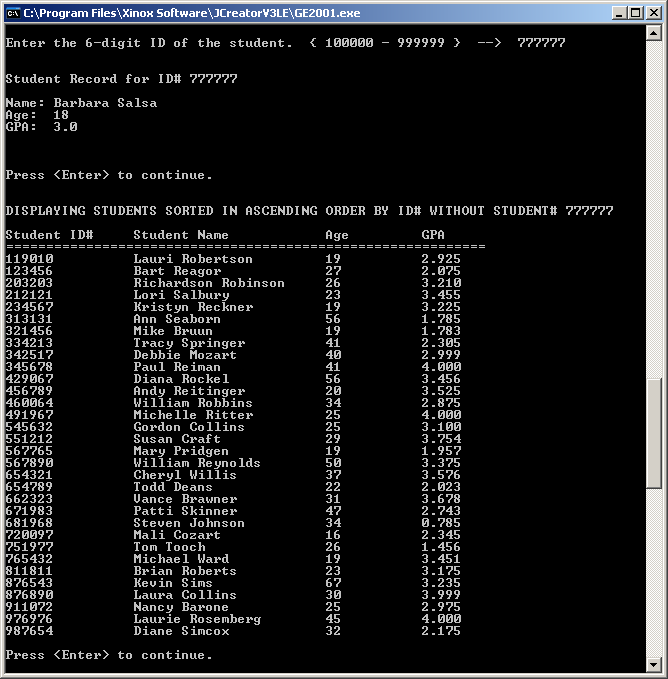


**100-Point Version Specifics**

The 100-point version requires everything from the 90-point version but adds a **delete** method to delete the student whose information you displayed at the end of the 90-point version. This **delete** method will require a search to find the index of the item that needs to be deleted. After that index is found every item after that needs to be copied to the previous index. The **size** of the array will then need to be reduced by one.

**100-Point Version Output**

The output will include everything you saw in the 80-Point Version Output, plus what is below:



**110-Point Version Specifics and Output**

The 110-point version is identical to the 100-point version except it uses the *Binary Search* instead of the *Linear Search*. The output will be identical to what you see above for the 100-point version.