1. Nick Wright CSC285 Problem 2
2. Java code

import java.io.File;  
import java.io.FileNotFoundException;  
import java.util.ArrayList;  
import java.util.Scanner;  
  
class Student1 implements Comparable{ //implementing comparable in order to override the compareTo function  
 protected int pscore; //this is the percent score for the student  
 protected int[] testScore = new int[3]; //testScore is an integer array for the test scores of a student  
 protected String id; //String id is the variable to hold the student id  
 protected String name; //String name is the variable that will hold the ame of the student  
 protected String letterGrade; //a string variable that is used to store the letter grade  
  
 public Student1(){}; //default constructor for student  
  
 public Student1(int[] test, String id, String name){ //this is the constructor for the student  
 this.id=id;  
 this.name=name;  
 for(int i=0; i<3; i++){ //for loop is used to put the test scores in the testScore array  
 this.testScore[i]=test[i];  
 }  
 pscore = (int)(((test[0]+test[1]+test[2])/3.0)+0.5); //calculates the average test score as an integer  
  
 if(pscore >= 90){ //if else statement chain that will calculate letterGrade based on the pscore variable  
 letterGrade = "A";  
 }else if((pscore<90) && (pscore>=80)){  
 letterGrade = "B";  
 }else if((pscore<80) && (pscore>=70)){  
 letterGrade = "C";  
 }else if((pscore<70) && (pscore>=60)){  
 letterGrade = "D";  
 }else{  
 letterGrade = "F";  
 }  
 }  
  
 public int getPscore() { //a getter method for pscore that is used in the comapreTo method  
 return pscore;  
 }  
  
 public static void AddStudent( ArrayList<Student1> Academic\_Class, Student1 Obj){ //static method AddStudent that adds students to the array list  
 Academic\_Class.add(Obj);  
 }  
 public static void DeleteStudent(ArrayList<Student1> Academic\_Class, String StudentID){ //static method DeleteStudent that removes students from the array list  
 for(int i=0; i<Academic\_Class.size(); i++){  
 if(Academic\_Class.get(i).id.equals(StudentID)){ //comparing the id of the current student to the id of the student being removed  
 Academic\_Class.remove(Academic\_Class.get(i)); //if the id's match, the current student gets removed  
 }  
 }  
 }  
 public static void SortLarge( ArrayList<Student1> Academic\_Class){ //static method SortLarge sorts the students from largest to smallest based on their pscore variable  
 Student1 saveStudent = new Student1(); //saveStudent and saveStudent2 are new objects that will be used to help swap objects in the array  
 Student1 saveStudent2 = new Student1();  
 int whileCondition = 1; //whileCondition is an int used to end the loop  
 while(whileCondition == 1){  
 whileCondition = 0;  
 for(int i=0; i<Academic\_Class.size()-1; i++){  
 switch (Academic\_Class.get(i).compareTo(Academic\_Class.get(i+1))){  
 case 1: //the objects are in the right order  
 break;  
 case -1: //the objects are out of order and must be changed  
 saveStudent=Academic\_Class.get(i);  
 saveStudent2=Academic\_Class.get(i+1);  
 Academic\_Class.remove(i);  
 Academic\_Class.add(i,saveStudent2);  
 Academic\_Class.remove(i+1);  
 Academic\_Class.add(i+1,saveStudent);  
 whileCondition = 1;  
 break;  
 default: //objects are equal or no change  
 }//end of switch  
 }//end of for loop  
 }//end of while loop  
 } //end of SortLarge  
  
 @Override  
 public String toString() { //changing the format for when we want to print out an object  
 return id + " " + name + " " + testScore[0] + " " + testScore[1] + " " + testScore[2] + " " + pscore + "% " + letterGrade;  
 }  
  
 @Override  
 public int compareTo(Object o) { //overriding the compareTo function  
 if(getPscore()>((Student1)o).getPscore()){  
 return 1; //returning 1 if object 1 is larger than object 2  
 } else if (getPscore()<((Student1)o).getPscore()){  
 return -1; //returning -1 if object 1 is smaller than object 2  
 } else {  
 return 0; //will return 0 if objects are the same  
 }  
 }  
} //end of Student1 class  
  
public class CodeforProb2CSC285 extends Student1{  
 public static void main(String[] args) throws FileNotFoundException { //FileNotFoundException needed for reading the input file  
 ArrayList<Student1> Academic\_Class = new ArrayList<Student1>(); //creating the array list Academic\_Class  
 Scanner input = new Scanner(new File("Input.txt")); //creating the scanner to read through the input file  
  
 while(input.hasNext()){ //while loop used to iterate through the input file and assign variables their values  
 int[] test = new int[3]; //creating array test, String Sid, and String name to hold the values that are being read in  
 String Sid = input.next();  
 String name = input.next();  
 test[0] = input.nextInt();  
 test[1] = input.nextInt();  
 test[2] = input.nextInt();  
  
 Student1 workStu = new Student1(test, Sid, name); //creating a new student object  
 Academic\_Class.add(workStu); //storing object in Academic\_Class  
 }  
  
 System.*out*.println("List the objects from the class ArrayList including the % score and the grades.");  
 System.*out*.println("Student ID | Name | Test 1 | Test 2 | Test 3 | Percent Score | Letter Grade");  
 for(int i =0; i < Academic\_Class.size(); i++){ //for loop is used to iterate through the ArrayList and print out each object  
 System.*out*.println(Academic\_Class.get(i));  
 }  
  
 *DeleteStudent*(Academic\_Class, "42P4"); //deleting the student with the same student id  
 *DeleteStudent*(Academic\_Class, "45A3"); //do not need to use dot syntax because the method is static  
  
 System.*out*.println("\n" + "List the ArrayList with the dropped student records.");  
 System.*out*.println("Student ID | Name | Test 1 | Test 2 | Test 3 | Percent Score | Letter Grade");  
 for(int i =0; i < Academic\_Class.size(); i++){ //for loop is used to iterate through the ArrayList and print out each object  
 System.*out*.println(Academic\_Class.get(i));  
 }  
  
 *AddStudent*(Academic\_Class, new Student1(new int[]{80,75,98},"67T4","Clouse,B" )); //adding new students to the array list  
 *AddStudent*(Academic\_Class, new Student1(new int[]{75,78,72},"45P5","Garrison,J" )); //do not need to use dot syntax because method AddStudent is static  
 *AddStudent*(Academic\_Class, new Student1(new int[]{85,95,99},"89P0","Singer,A" ));  
 *SortLarge*(Academic\_Class); //SortLarge used to resort the class from largest to smallest letter grade, dot syntax is not needed because method SortLarge is static  
  
 System.*out*.println("\n" + "List the ArrayList after the new students have been added and their grades have been sorted from highest to lowest.");  
 System.*out*.println("Student ID | Name | Test 1 | Test 2 | Test 3 | Percent Score | Letter Grade");  
 for(int i =0; i < Academic\_Class.size(); i++){ //for loop is used to iterate through the ArrayList and print out each object  
 System.*out*.println(Academic\_Class.get(i));  
 }  
 } //end of main method  
} //end of CodeforProb2CSC285

1. Input files

45A3 Jones,H 86 88 95

34K5 Horner,M 67 75 23

56J8 Gach,T 75 85 95

34U8 Hunter,C 100 50 75

42P4 Hinrichs,S 85 75 65

78T6 Johnson,K 80 78 89

44L2 Levitte,H 56 66 99

88I9 Garner,J 95 92 98

1. Output Files

List the objects from the class ArrayList including the % score and the grades.

Student ID | Name | Test 1 | Test 2 | Test 3 | Percent Score | Letter Grade

45A3 Jones,H 86 88 95 90% A

34K5 Horner,M 67 75 23 55% F

56J8 Gach,T 75 85 95 85% B

34U8 Hunter,C 100 50 75 75% C

42P4 Hinrichs,S 85 75 65 75% C

78T6 Johnson,K 80 78 89 82% B

44L2 Levitte,H 56 66 99 74% C

88I9 Garner,J 95 92 98 95% A

List the ArrayList with the dropped student records.

Student ID | Name | Test 1 | Test 2 | Test 3 | Percent Score | Letter Grade

34K5 Horner,M 67 75 23 55% F

56J8 Gach,T 75 85 95 85% B

34U8 Hunter,C 100 50 75 75% C

78T6 Johnson,K 80 78 89 82% B

44L2 Levitte,H 56 66 99 74% C

88I9 Garner,J 95 92 98 95% A

List the ArrayList after the new students have been added and their grades have been sorted from highest to lowest.

Student ID | Name | Test 1 | Test 2 | Test 3 | Percent Score | Letter Grade

88I9 Garner,J 95 92 98 95% A

89P0 Singer,A 85 95 99 93% A

56J8 Gach,T 75 85 95 85% B

67T4 Clouse,B 80 75 98 84% B

78T6 Johnson,K 80 78 89 82% B

34U8 Hunter,C 100 50 75 75% C

45P5 Garrison,J 75 78 72 75% C

44L2 Levitte,H 56 66 99 74% C

34K5 Horner,M 67 75 23 55% F

Process finished with exit code 0