Hands-on Experiment # 9-1 : Worksheet

Section_____2_____ Date_____29/03/2018_____

No more than 3 students per one submission of this worksheet.

Student ID _____6031851521_____ Name___Sarun Nuntaviriyakul_____

Student ID _____6031848721_____ Name___Watcharin Kriengwatana_____

Student ID _____6031847021_____ Name___Wasuthon Klyhirun_____

Part A: Searching or Sorting? Pick one.

We are revisiting the processing of *score.csv* again. Now that we know how to use array, things should be much simpler. Also, this time, you are also given with a class called *StudentScore* which is designed so that all scores of a student with a specific ID are stored in a single object of *StudentScore*.

First, study, compile and run *TestStudentScore.java* so that you get some idea of how the *StudentScore* class can be used.

Explain what each statement in TestStudentScore.java does.

1	Set Boolean withHeader to true;
2	Create a new StudentScore object called s1 with attribute("5630000021,10,10,9,8,7");
	s1 will have id = 5630000021 and scores = {10,10,9,8,7}
3	call function printScore from s1, passing true as an withHeader(true).
	s1's scores, including total score, will be printed with a header
4	
5	Set Boolean withHeader to false;
6	Create a new StudentScore object called s2 with attribute ("5630000121,8,8,10,6,5");
	s2 will have id = 5630000121 and scores = {8,8,10,6,5}
7	call function printScore from s1, passing withHeader(false) as an argument.
	s2's scores, including total score, will be printed without a header

Complete either *ScoreSort.java* or *ScoreLookUp.java* (only one of them) so that the program you choose works according to what are described in *Hands-on Experiment 9-1.pdf*.

The specification of the *StudentScore* class is also given in *L09-1.pdf*.

Which choice did you choose? Why did you prefer the choice over the other one?

Both. Because I can do both.

Have you been able to complete the program? If not, what were the problems?

Yes

Does it work correctly in all cases? If not, what are the cases those your program does not work correctly?

(ScoreSort)

No, if many students have the same score, the sore will be printed in increasing order of id.

For example, 10 students got 40 points but some are left out if you print top 20.

Include the screenshots below.

```
C:\Users\him10\Com Prog\week9>java ScoreSort
Student ID      Q0      Q1      Q2      Q3      Q4      Total
5639110921      10      10      8       10      8       46
5676237921      10      10      9       9       7       45
5698907421      10      10      6       10      7       43
5635209721      9       10      8       10      5       42
5692820921      7       10      8       9       8       42
5696614321      5       10      9       10      8       42
5616166921      7       10      6       10      8       41
5631371621      9       9       6       9       8       41
5654331721      8       8       7       10      8       41
5663510921      8       7       8       10      8       41
5672488021      8       10      6       10      7       41
5683947321      9       9       8       7       8       41
5690418521      8       9       9       9       6       41
5694033521      8       8       8       9       8       41
5605082021      9       9       9       10      3       40
5607383721      10      10      5       7       8       40
5609460721      8       8       8       9       7       40
5622997221      9       10      4       9       8       40
5626572621      10      10      8       7       5       40
5633380221      8       10      8       10      4       40
```

List all your source code here.

```
import java.util.*;

import java.io.*;

public class ScoreSort{


        public static void main(String [] args){

                StudentScore [] scores = readScoreFile();

                sortByTotal(scores);

                listTop(scores, 20);

        }
```

```
	public static StudentScore [] readScoreFile(){
		List <StudentScore> list = new ArrayList<>();
		try{
			Scanner sc = new Scanner(new File("score.csv"));
			sc.nextLine();
			while(sc.hasNextLine())
				list.add(new StudentScore(sc.nextLine()));
		}
		catch (IOException e) {
			e.printStackTrace();
		}
		return list.toArray( new StudentScore[list.size()] );
	}

	public static void sortByTotal(StudentScore [] data){
		Arrays.sort(data, (s1,s2) -> s2.getTotalScore() - s1.getTotalScore());
	}

	public static void listTop(StudentScore[] sortedScores, int n){
		sortedScores[0].printScore(true);
		for(int i = 1 ; i < n ; i++)
			sortedScores[i].printScore(false);
	}

}
```

Part B (Optional): Complete the other choice

Do this part if you want to.

Complete the other program.

List all your source code here.

```
import java.util.*;

import java.io.*;

public class ScoreLookup2 {

	public static void main(String [] args) {

		StudentScore[] StudentScore = readScoreFile();

		boolean toQuit = false;

		do{

			toQuit = false;

			String choice = getStudentID();

			if(choice.equals("quit"))

				toQuit = true;

			else

				showScoreOf(StudentScore, choice);

		}while(!toQuit);

	}


	public static StudentScore[] readScoreFile() {

		List <StudentScore> list = new ArrayList<>();

		try{

			Scanner sc = new Scanner(new File("score.csv"));

			sc.nextLine();

			while(sc.hasNextLine())

				list.add(new StudentScore(sc.nextLine()));
```

```
                }
                catch (IOException e) {
                        e.printStackTrace();
                }
                return list.toArray( new StudentScore[list.size()] );
        }

        public static String getStudentID() {
                Scanner kb = new Scanner(System.in);
                System.out.print("Enter Student ID\n>> ");
                return kb.next();
        }

        public static void showScoreOf(StudentScore[] data, String id) {
                for(StudentScore student : data)
                        if(student.id.equals(id)){
                                student.printScore(true);
                                return;
                        }
                System.out.println("Student ID not found.");
        }
}
```

Submit this worksheet (by only one member of the group) via http://www.myCourseVille.com (Assignments > Hands-on Experiment # 9-1) **within the day after your lecture**.