

Sit-in Lab #3

CS1101 AY2009/10 Semester 1

Date: 23 October 2009, Friday

School of Computing, National University of Singapore

0 Instructions

Please follow the instructions on the first page.

There are **two** exercises in this lab. Time limit is **1 hour 30 minutes**.

Do NOT:

- Use recursion
- Create any additional .java programs (they will not be uploaded onto CM)

Please do NOT reveal the question to anybody until tomorrow. Anybody found to have leaked the question in full or in part will be subject to disciplinary actions.

1 Exercise 1: Using ArrayList

[40 marks]

There are two programs in this exercise: **Employee.java** and **Roster.java**. The program **Employee.java** is a complete program, which you are not supposed to modify even a bit. You are required to complete only **Roster.java**, which is an application program using the Employee class.

You may write additional method(s) in **Roster.java** if you wish. (But do not create additional class(es) or program(s).)

We will ignore your **Employee.java** and use ours (which is the one given to you) to test your **Roster.java** program. Hence, please treat **Employee.java** as a write-protected file.

The complete **Employee.java** program and the **Roster.java** skeleton program are loaded into the working directory, and are shown below (some lines in the print-out below are merged into a single line to save space) for your convenience.

```

/**
 * CS1101 AY2009/10 Sit-in Lab #3 Set A Ex 1
 * Employee.java
 * This defines the Employee class that contains
 * name and employee id.
 * Note: You are NOT to change this program a bit.
 */

class Employee {

    // Data members
    String name;
    int    id;

    // Default constructor
    public Employee() {
        this("unknown", 0);
    }

    // Alternative constructor
    public Employee(String newName, int newId) {
        name = newName;
        id = newId;
    }

    // Accessors
    public String getName() { return name; }

    public int getId() { return id; }

    // Mutators
    public void setName(String newName) { name = newName; }

    public void setId(int newId) { id = newId; }

    // Returns string representation of Inventory object
    public String toString() {
        return name + " <" + id + ">";
    }

    // Returns true if this object and obj object
    // contain the exact content
    public boolean equals(Object obj) {
        if (obj instanceof Employee) {
            Employee emp = (Employee) obj;
            return name.equals(emp.name) &&
                id == emp.id;
        }
        else
            return false;
    }
}

```

```

/**
 * CS1101 AY2009/10 Sit-in Lab #3 Set A Ex 1
 * Roster.java
 * This program reads a list of employee names, generates
 * the employee ids and creates an ArrayList of Employee
 * objects. It then reads a list of instructions on
 * duty swaps. It uses the Employee class.
 *
 * <Fill in your name and discussion group here>
 */
// import appropriate package(s)

class Roster {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        List<Employee> list = new ArrayList<Employee>();
        Employee emp;

        System.out.println("Enter names, "
            + "end by entering an empty string:");

        // Fill in your code
        // You must use the ArrayList list in your code

        // Read names

        // Display the list after reading in all names
        displayList(list);

        // Read swap list

    }

    // Display all elements in the list.
    // This method is provided.
    public static void displayList(List<Employee> aList) {
        Employee emp;
        Iterator<Employee> itr = aList.iterator();

        while (itr.hasNext()) {
            emp = itr.next();
            System.out.println(emp);
        }
    }
}

```

Task

You are to write **Roster.java** to perform the following:

- Read a list of employees' names. For each employee read, generate the employee-id in running order (i.e., first employee entered has employee-id of 0, second employee has id of 1, etc.), create the Employee object and add it into the ArrayList **list**. End the list of employee's names by entering an empty string.
- Display the ArrayList **list** using the given **displayList()** method. This **list** is also used as the duty roster.
- Next, read a list of instructions. An instruction can either be "swap" or "display".

- If the instruction is "swap", read two integers in the next two lines. For example:

```
swap
3
5
```

The above example means that the employee in index 3 of **list** is to be swapped with the employee in index 5. If any of the two integers refers to an invalid index, do not perform the swap. Your program must not crash.

- If the instruction is "display", display the **list** using the given **displayList()** method.
- End the list of instructions by entering an empty string for the instruction.

Sample Run

A sample run of the program is given below. Essential outputs are shown in bold.

```
Enter names, end by entering an empty string:
Lee Lily
Ang Moh Kyo
How Ah You
Brendon Chew
Muhammad Ishak
    ← user pressed enter here to end list of names

Lee Lily <0>
Ang Moh Kyo <1>
How Ah You <2>
Brendon Chew <3>
Muhammad Ishak <4>
Read swap list:
swap
0
5
display
Lee Lily <0>
Ang Moh Kyo <1>
How Ah You <2>
Brendon Chew <3>
Muhammad Ishak <4>
swap
0
3
swap
4
2
display
Brendon Chew <3>
Ang Moh Kyo <1>
Muhammad Ishak <4>
Lee Lily <0>
How Ah You <2>
    ← user pressed enter here to end list of instructions
```

Marking Scheme

This exercise constitutes 40 marks. 30 marks are allocated to the correctness, and 10 marks to design and style.

Advice: Do not spend more than 35 minutes on Exercise 1, so that you have enough time for Exercise 2.

2 Exercise 2: Using 2-dimensional Array

[60 marks]

You are to write a single program **TriangularArray.java** to perform the following:

- Read a positive integer value for *size*.
- Create a 2-dimensional triangular array whereby the first row has 1 element, second row 2 elements, ..., last row *size* elements.
- Read integers to fill in the array created above.
- Determine the maximum value in each diagonal of the array, from the shortest diagonal to the longest, as shown in the sample runs below.

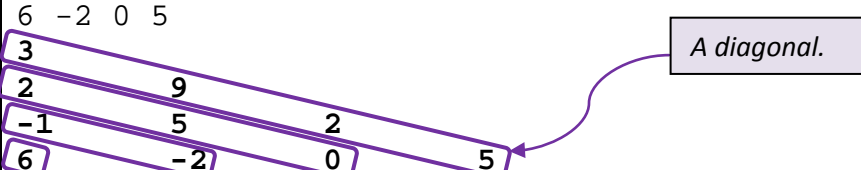
You must use array and not ArrayList for this exercise.

The skeleton **TriangularArray.java** program is loaded into the working directory. A **printTriangularArray()** method is provided in the skeleton program. Do NOT modify **printTriangularArray()**. You may write your own additional method(s) if you wish. (But do not create additional class(es) or program(s).)

Sample Runs

Three sample runs of the program are given below (with explanatory note). Essential outputs are shown in bold.

```
Enter size: 4
Enter values for array:
3
2 9
-1 5 2
6 -2 0 5
3
2 9
-1 5 2
6 -2 0 5
Maximum values in diagonals: 6 -1 5 9
```



```
Enter size: 2
Enter values for array:
7
9 7
7
9 7
Maximum values in diagonals: 9 7
```

```
Enter size: 6
Enter values for array:
-8
10 2
-2 12 3
5 0 4 9
10 10 -6 8 1
3 2 1 0 -1 -2
-8
10 2
-2 12 3
5 0 4 9
10 10 -6 8 1
3 2 1 0 -1 -2
Maximum values in diagonals: 3 10 10 0 12 9
```

Marking Scheme

This exercise constitutes 60 marks. 40 marks are allocated to the correctness of the methods that you are supposed to complete or write. 20 marks are allocated to design and style.

=== End of paper ===