Concordia University
Computer Science & Software Engineering
Faculty of Engineering and Computer Science

# Concordia University
# COMP 248 – Winter 2020
# Assignment 3

| | |
|---|---|
| **Due Date:** | By 11:55 PM, March 13, 2020 |
| **Evaluation:** | 5% of final mark (see marking rubric at the end of handout) |
| **Late Submission:** | none accepted |
| **Purpose:** | The purpose of this assignment is to help you learn Java arrays and simple classes |
| **CEAB/CIPS ATTRIBUTES:** | Design/Problem analysis/Communication Skills |

## General Guidelines When Writing Programs:

Refer to assignment #1 handout.

## Question 1: Rank the IMO Participants

International Mathematical Olympiad (IMO) is held each year since 1959. It is the oldest of the kind. Each team contains 6 members. The rank of a team is decided by the sum of the scores of all its 6 members. The rank of a participants is decided by his/her individual scores among all the participants.

You are asked to develop a program to rank the teams and the participants. Some more detailed requirements:

- The score of a participant is an integer and in the range of [0, 50].
- 2D array is used to store a predefined data set.
- Ranking an array is different from sorting an array. Assume an array is {3, 4, 2}, the ranks of the elements in ascent order are {2,3,1}. If two elements are the same, they are ranked the same. The next rank after the duplicate elements should be added by 1. Assume an array is {3, 3, 4, 2}, the ranks are {2, 2, 4, 1}. Intensively use array and loop. No need to use any sorting algorithm.
- Print the results. You do not need to sort the results.

Sample Testing Case:

The raw data:

| | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| Team 1 | 39 | 40 | 17 | 35 | 42 | 6 |
| Team 2 | 40 | 41 | 27 | 41 | 42 | 36 |
| Team 3 | 42 | 40 | 26 | 42 | 42 | 35 |

The printed results:

```
The ranking list:
          P1     P2     P3     P4    P5    P6     Total Rank
Team 1 39(11) 40 (8) 17(17) 35(13) 42 (1)  6(18)   179  3
Team 2 40 (8) 41 (6) 27(15) 41 (6) 42 (1) 36(12)   227  1
Team 3 42 (1) 40 (8) 26(16) 42 (1) 42 (1) 35(13)   227  1
```

Please reference to the 2019 IMO result page if you need more testing data (https://www.imo-official.org/year_country_r.aspx?year=2019)

**Question 2: Create Simple Classes**

You are asked to create an application to manage employees. Two classes, Date and Employee, are designed.

The details of the Date class:

- The properties: day (type of int), month (type of String), year (type of int). The months are "January", "February", …, "December". The day is between 1 and 31. The year is between 1000 to 9999. Validation is needed for user input data.
- Constructor 1: a constructor without parameter. You need to give a valid default value for day, month, and year
- Constructor 2: a constructor takes three parameters: String monthString, int day, int year. You need to validate the actual parameters and initialize the properties. Use the dateOK( ) method given below to do the validation.
- Constructor 3: a constructor takes three parameters: int monthInt, int day, int year. You need to validate the actual parameters and initialize the properties. You need to convert the int monthInt to String, e.g. "January", and store the string to the property month.
- public boolean equals(Date otherDate): return true when otherDate has the same day, month and year.
- private String monthString(int): convert an integer month to its corresponding name (String)
- private boolean dateOK(String, int, int): validate if the actual parameters represent a valid date. See the definition of the properties for validate range. Leap year(*) is checked for February. No other special rules need to apply.
- private boolean dateOK(int, int, int): validate if the actual parameters represent a valid date. (different than the above one). See the definition of the properties for validate range. Leap year is checked for February. No other special rules need to apply
- public setDate(int, int, int): validate the actual parameters and set the values of the properties
- public setDate(String, int, int): validate the actual parameters and set the values of the properties

- public int getDay( ), public int getMonth( ), public void getYear( ): the accessors of the properties. Notice that getMonth( ) converts the string of month to integer and returns the integer.
- public void setDay(int), public void setMonth(int), public void setMonth(String), public setYear(int): the mutators of the properties.

(*) Leap years occur in years exactly divisible by four, except that years ending in 00 are leap years only if they are divisible by 400.

The details of the Employee class:

- The properties:
  - The employee name: type of String
  - The hiring date: type of Date
- Constructor 1: a constructor without parameter. You need to give a valid default value for the employee name and the hiring date
- Constructor 2: a constructor takes two parameters: String aName, Date aDate..
- public int seniority(Employee e): compare the seniority of this Employee and the Employee e. The seniority is decided by the hiring date. It returns 0, if the two employees are hired on the same date. It returns -1, if this Employee is hired before the Employee e. It return 1, if this Employee is hired after the Employee e.
- public boolean equals(Employee e): return true when Employee e has the same day, month, year.
- public String getName( ), public Date getHireDate( ): the accessor of the properties.
- public void setHireDate (Date), public void setName(String): the mutator of the properties.

A driver class to test the two classes:

- Create two Employee instances with different data: call them e1 and e2.
- Compare the seniority of e1and e2: whoever hired earlier is more senior.
- Use the e3 is a second instance of e1 that the day, the month, and the year are the same as e1.
- Use equals(Employee e) to compare e1 vs e2 , e1 vs e3.
- Use == to compare e1 vs e2, e1 vs e3.
- Print the status of e1, e2, and e3: print the values of the properties

You can add any methods to implement the designed functions.

A sample output can be:

```
The first employee (e1):
Jack October 5, 2019

The second employee (e2):
Peter December 3, 2008

Peter is more senior than Jack

The duplicated employee (e3):
Jack October 5, 2019

e1.equals(e3) => true
e2.equals(e3) => false
(e1 == e3) => false
(e2 == e3) => false
```

## Submitting Assignment 3

**What to submit:**

Zip the 2 source codes (the .java files only please, **not** the entire project) of this assignment as a .ZIP file (**NOT** .RAR) using the following naming convention:

*a#_studentID*, where # is the number of the assignment and *studentID* is your student ID number.

For example, for this third assignment, student 123456 would submit a zip file named a3_123456.zip

**How to submit:**

For sections U & W, please check your Moodle course webpage and for section EC please check your eConcordia webpage for instructions on how to submit your assignment.

| Evaluation Criteria for Assignment 3 | |
|---|---|
| Source Code | |
| **Comments for all 2 questions (3 pts.)** | |
| Description of the program (authors, date, purpose) | 1 pt. |
| Description of variable and constants | 1 pt. |
| Description of the algorithm | 1 pt. |
| **Programming styles for all 2 questions (3 pts.)** | |
| Use of significant names for identifiers | 1 pt. |
| Indentation and readability | 1 pt. |
| Welcome Banner/Closing message | 1 pt. |
| **Question 1 (7 pts.)** | |
| The usage of array | 2 pts. |
| The correctness of ranking a team | 2 pts. |
| The correctness of ranking a participant | 2 pts. |
| Display correct results | 1 pt. |
| **Question 2 (7 pts.)** | |
| Implementation of Date class | 2 pts. |
| Implementation of Employee class | 2 pts. |
| The correctness of seniority | 1 pts. |
| The correctness of comparison with equals | 1 pt. |
| The correctness of comparison with == | 1 pt. |
| **TOTAL** | **20 pts.** |