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

Concordia University
COMP 248 – Fall 2018
Assignment 3

Due Date:	By 11:59pm November 9, 2018
Evaluation:	6% 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 primitive type arrays, strings arrays, 2-dimensional arrays and simple class design.
CEAB/CIPS Attributes:	Design/Problem analysis/Communication Skills

General Guidelines When Writing Programs:

- Please refer to the handout of Assignment #1.

Question 1 (One and Two Dimensional Arrays)

This program is about generating a road map of matches and related goal statistics in the FIFA World Cup Final. The game is repeated until your favourite soccer team (country) wins the Final in a tournament, or until a maximum number of tournaments. For simplicity, we will start from the *Round of 16*. Assume that you are given the list of 16 countries in an array. You will ask the user for his/her favourite team (keyboard input). Your program will then output a possible road map of 8 matches in the *Round of 16*, 4 *Quarter Finals*, 2 *Semi-Finals*, 1 *Final*, and the final winner of a tournament. A tournament is repeated (with the same list of countries) until the user's chosen team wins, or the max number of tournaments specified in the program has reached (e.g., 20 tournaments). At the end, you will show total scores in each match, the average goals for each tournament, the overall average for all tournaments in the game, and the total number of matches where the number of goals is greater than the overall average.

Note that a ***match*** is played between two teams/countries, a ***tournament*** consists of all the matches leading to a final winner (i.e., all matches including the final), and ***the game*** includes all the tournaments until your favourite team wins, or the max tournament count is reached.

Your program should proceed as follows:

1. The names of the teams are provided as an array of Strings.

```
String[] teams16 = {"Uruguay", "Portugal", "France", "Argentina", "Brazil", "Mexico",
"Belgium", "Japan", "Spain", "Russia", "Croatia", "Denmark", "Sweden", "Switzerland",
"Colombia", "England"};
```

2. Ask user for her favourite team. If the provided team does not exist in the 16 names, you exit with a proper message (e.g., ``Your team is not in the Round of 16''). Ignore case, and any additional white-spaces entered by the user.

3. Generate the output roadmap as follows. Choose teams in pairs for each match (e.g., take the first two teams from your array for Match 1, and the next two teams for Match 2, and so on). The winner of each match is selected as follows: in the given 90-minute period, each team scores between 0 to 4 goals (generated randomly using the Java Random class), the team with more goals wins. If the play is a draw after 90 minutes

(e.g., 2-2), there is a 30-minute *sudden death* period; assume that one team will score a goal during this period (chosen randomly), and that team will win (the play ends with the sudden death goal). In either case, you show the final score line, and the winner moves to the next phase. The final match winner is the FIFA World Cup Winner of 2018. Iterate the tournament until your favourite team wins (due to random goal selection, the score-lines will change each time), or you reach the maximum tournament count.

Sample output from the program (the user input is shown in grey):

```
Enter your favourite team: Japan

ROUND OF 16[Uruguay 1:2 Portugal] [France 0:1 Argentina] [Brazil 1:2 Mexico] [Belgium 1:3
Japan] [Spain 3:4 Russia] [Croatia 1:4 Denmark] [Sweden 4:5 Switzerland] [Colombia 1:3
England]
QUARTER-FINALS[Portugal 4:1 Argentina] [Mexico 4:1 Japan] [Russia 0:1 Denmark]
[Switzerland 4:1 England]
SEMI-FINALS[Portugal 3:4 Mexico] [Denmark 1:3 Switzerland]
FINAL[Mexico 3:1 Switzerland]
Tournament: 0 The WINNER is: Mexico

ROUND OF 16[Uruguay 0:1 Portugal] [France 2:1 Argentina] [Brazil 1:4 Mexico] [Belgium 2:4
Japan] [Spain 3:0 Russia] [Croatia 1:2 Denmark] [Sweden 1:4 Switzerland] [Colombia 3:1
England]
QUARTER-FINALS[Portugal 2:1 France] [Mexico 0:2 Japan] [Spain 4:0 Denmark] [Switzerland
0:2 Colombia]
SEMI-FINALS[Portugal 4:1 Japan] [Spain 3:1 Colombia]
FINAL[Portugal 3:0 Spain]
Tournament: 1 The WINNER is: Portugal

ROUND OF 16[Uruguay 3:2 Portugal] [France 4:3 Argentina] [Brazil 4:0 Mexico] [Belgium 4:3
Japan] [Spain 4:3 Russia] [Croatia 4:5 Denmark] [Sweden 0:4 Switzerland] [Colombia 0:1
England]
QUARTER-FINALS[Uruguay 2:3 France] [Brazil 4:1 Belgium] [Spain 0:2 Denmark] [Switzerland
1:0 England]
SEMI-FINALS[France 1:2 Brazil] [Denmark 2:0 Switzerland]
FINAL[Brazil 0:1 Denmark]
Tournament: 2 The WINNER is: Denmark

ROUND OF 16[Uruguay 2:3 Portugal] [France 0:3 Argentina] [Brazil 3:4 Mexico] [Belgium 1:4
Japan] [Spain 5:4 Russia] [Croatia 1:2 Denmark] [Sweden 3:1 Switzerland] [Colombia 0:3
England]
QUARTER-FINALS[Portugal 3:1 Argentina] [Mexico 2:4 Japan] [Spain 3:4 Denmark] [Sweden 4:1
England]
SEMI-FINALS[Portugal 3:4 Japan] [Denmark 4:5 Sweden]
FINAL[Japan 1:0 Sweden]
Tournament: 3 The WINNER is: Japan

It took 4 tournament(s) of the game for Japan to win!!!
```

4. You also need to keep counts of the total goals scored in each play, and at the end of the game (either when your team won, or the max tournament count has reached), you show the following: total goals from each match in a tournament and the average goal for each tournament; the average goal for the entire game (all matches in all tournaments); the number of matches with scores greater than the overall average. After each match, you need to store the total goals in that match for calculating the above statistics; a 2D array will be needed. Note that there are 15 matches in a tournament. A single digit is kept after the decimal point for the real numbers (averages). Sample output (following the score lines in the part 3 example):

```
GOAL STATS

[Tournament 0] Total goals: [3, 1, 3, 4, 7, 5, 9, 4, 5, 5, 1, 5, 7, 4, 4] [Average: 4.5]
[Tournament 1] Total goals: [1, 3, 5, 6, 3, 3, 5, 4, 3, 2, 4, 2, 5, 4, 3] [Average: 3.5]
[Tournament 2] Total goals: [5, 7, 4, 7, 7, 9, 4, 1, 5, 5, 2, 1, 3, 2, 1] [Average: 4.2]
[Tournament 3] Total goals: [5, 3, 7, 5, 9, 3, 4, 3, 4, 6, 7, 5, 7, 9, 1] [Average: 5.2]

Average goals for 4 tournament(s): 4.3
Total matches in all tournaments over the average goal value: 27
```

5. If your team does not win (i.e., the game ended when you reach the max tournament count), show the following message after the output in part 3 above (instead of showing ``It took 4 tournament(s) of the game for Japan to win!!!'').

```
Sorry, Japan didn't win in 20 tournaments!
```

You still need to show the goal statistics as in part 4.

Question 2 (Simple Class Exercise)

a) Define a class named House that store information about a house. It should comprise the following:
   1. Private instance variables to store age of the House, its type (Detached, Semi-Attached, Attached) and its cost.
   2. 4 constructors: No argument (sets age to 50, type to Attached and cost to 100000), one argument constructor (sets cost to a value, age - 50 and type - Attached), two argument constructors (sets age to a value, cost to a value , and type to Attached), three argument constructors (sets age to a value, cost to a value, and type to Attached, semi-detached, or detached)
   3. 3 Accessor methods: - methods to return age, type and cost respectively
   4. 5 Mutator methods: - 3 methods for setting the three values independently, a method to set all three values and a method to set age and cost of the house.
   5. A public method called estimatePrice() that returns cost of a house based on type and age. An attached costs $100,000, appreciates 1% every year in first five years and 2% every year afterwards. A Semidetached costs $150,000, appreciates 2% every year in first five years and 3% every year afterwards. A detached costs $200,000, appreciates 2% every year in first five years and 2% every year afterwards.
   6. A toString() method that returns type of the house and its age and its cost.
   7. An equals() method to test for equality of two objects of class House based on type and age.
   8. isLessThan() and isGreaterThan() method to compare between the prices of two objects of class House.

b) Write a test code
   1. Which declares 4 house objects using 4 different constructors and output description of the 4 houses.
   2. Test your accessor methods.
   3. Calculate the estimated price of houses given type and age (include 1 attached and 1 detached).

4. Test out all 5 mutator methods to modify the attributes of different House objects.
5. Test methods toString(), equals() , isLessThan() and isGreaterThan() for different House objects.

Here is an example of the output to illustrate the expected behavior of your program.

```
House H1: This House is type attached. Its age is 1950 and costs $100000.0
House H2: This House is type attached. Its age is 1950 and costs $100000.0
House H3: This House is type attached. Its age is 4 and costs $120000.0
House H4: This House is type detached. Its age is 2 and costs $220000.0

Accessor Method: The housetype for house H4 is detached, its age is 2, and it costs $220000.0

The estimated price of house H3 is $104800.0
The estimated price of car H4 is $208800.0

Mutator Method: The new age for house H3 is 5
Mutator Method: The new housetype for house H3 is semi-detached
Mutator Method: The new cost for house H3 is 240000.0
Mutator Method: The new house H3 age is 6 and its new cost is 245000.0
Mutator Method: The new housetype for house H3 is semi-detached, its new age is 14and its
cost is 260000.0

toString: This House is type semi-detached. Its age is 14 and costs $260000.0

Houses H1 and H2 are equal is true
Houses H1 and H4 are equal is false

House H4 is less than H3 is true

House H1 is greater than H3 is false
```

Submitting Assignment 3

- Zip the source code (the .java file only please) of this assignment.
- Naming convention for zip file: Create one zip file, containing the source files for your assignment. The zip file should be called *a#_studentID*, where # is the number of the assignment and *studentID* is your student ID number.
  For example, for the fourth assignment, student 123456 would submit a zip file named a4_123456.zip
- Refer to your section's Moodle page for instructions on where to submit your assignment.

Evaluation Criteria for Assignment 3 (25 points)

Source Code	
Comments for all 2 questions (3 pts.)	
Description of the program (authors, date, purpose)	1 pts.
Description of variables and constants	1 pt.
Description of the algorithm	1 pts.
Programming Style for all 2 questions (2 pts.)	
Use of significant names for identifiers	1 pt.
Indentation and readability	0.5 pt.
Welcome Banner/Closing message	0.5 pt.

Question 1 (10 pts.)		
Prompting user, input validation	1	pt.
Producing random match results each time	3	pt.
Display formatted match results	2	pt.
Display scores for each match with average goals	2	pt.
Display overall match average	1	pt.
Display total number of matches with greater than average goals	1	pt.
Question 3 (10 pts.)		
Constructor-Accessor-Mutator	3	pt.
Comparison (=,<,>) , toString	2	pts.
Other Methods	1	pts.
Test Code	4	
TOTAL	**25**	**pts.**