# PROG 1400 - Assignment 1

# JAVA AND OOP BASICS

Assignment Value: *12*% of overall course mark.

Due Date: **Feb. 17,2020.**

Late submissions will receive the standard late submission penalty as stated in the course outline.

#### Assignment Instructions:

Use IntelliJ to create a Java console application.

#### Submissions:

#### Submit your complete folder project or text files for all classes from your project on Bright space (Assignment 1 Dropbox)

#### Evaluation:

To insure the greatest chance of success on this assignment, be sure to check the marking rubric contained at the end of this document or in Brightspace. The rubric contains the criteria your instructor will be assessing when marking your assignment.

## Program – Fantasy Hockey

You have been asked to write a Java console application to track and gather statistics for a fantasy hockey league. The application requirements are as follows:

* The league has 3 teams with 4 players per team.
* The user will be presented with a set of data entry options for 3 teams of 4 players (see console output below)
  + For teams, gather the team name
  + For players, gather the player name, number of goals and number of assists
  + **Validation**: Names must be at least 3 characters. Numbers must be zero or greater.
* After gathering all the data, you will generate 2 reports (See example below):
  + A team stats report that displays the following:
    - Total goals and assists for the team.
    - Team Budget and Rating based on number of assists and goals.
  + A team will be given a random budget when created. The budget must be between zero and $100000. The report must output the budget dollar amount to 2 decimal places.
  + The team rating is calculated using the following formula:
    - 3 stars (\*\*\*) – the total goals and assists is greater than 20
    - 2 stars (\*\*) – the total goals and assists is greater than or equal to 10
    - 1 star (\*) – the total goals and assists is greater than zero
    - 0 stars – the total goals and assists is zero
  + The Stats per player report will report the player team name, their name, the player number of goals, assists and the total (see below).
* Program Organization: Make 3 Java classes: A Team class, A Player class and a Main class that launches the application, gathers input and displays output. Team and Player should have appropriate properties for their types. These classes should also have appropriate methods like outputPlayerDetails() etc.

## Examples and Testing

In the section below, you will be presented with console output of a successful execution of a sample solution to the program, which should help demonstrate how your input/output on the program should work. In addition to the sample values used in the screenshot(s), there should be validation on all inputs. You can expect your instructor to grade your assignment by trying different values than below. **In other words, you should thoroughly test your code before submitting!**

**Sample Output** - Make sure your program can output data *exactly* as shown below. Bold Text is for user entry.

FANTASY HOCKEY APPLICATION

TEAM ENTRY

================================

Enter name for team # 1:

**Ron's Top Picks**

Enter name for team # 2:

**Best Team Around**

Enter name for team # 3:

**My Random Picks**

PLAYER ENTRY

================================

Enter players for Ron's Top Picks:

Enter name for player # 1:

**Jane Smith**

Enter number of goals for Jane Smith:

**5**

Enter number of assists for Jane Smith:

**2**

Enter name for player # 2:

…

REPORT: Stats per Team

================================

Ron's Top Picks: G - 66 A - 63 Total - 129 Budget - $69596.13

Rating: \*\*\* stars

Best Team Around:G - 2 A - 3 Total - 5 Budget - $26706.63

Rating: \*\* stars

My Random Picks: G - 13 A - 13 Total - 26 Budget - $50519.41

Rating: \*\*\* stars

REPORT: Stats per Player

================================

Ron's Top Picks

Jane Smith: G - 10 A - 5 Total - 15

Ron's Top Picks

Jack Jones: G - 30 A - 10 Total - 40

Ron's Top Picks

Tim Hockey: G - 16 A - 18 Total - 34

Ron's Top Picks

Mike Sears: G - 10 A - 30 Total - 40

Best Team Around

Jacob Jackson: G - 1 A – 0 Total - 1

Best Team Around

Mahalia Roberts: G – 0 A - 2 Total - 2

Best Team Around

Sally Sampson: G - 1 A – 1 Total - 2

Best Team Around

Seigfried Smith: G - 0 A - 0 Total - 0

My Random Picks

Joe Murphy: G - 5 A - 3 Total - 8

My Random Picks

Ivor Callaly: G – 3 A - 7 Total - 10

My Random Picks

Rory Kirby: G - 1 A - 1 Total - 2

My Random Picks

Sheila Moriarty: G - 4 A - 2 Total - 6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Fantasy Hockey** | | |  |  |  |  |  |
| **Criteria** | | **Insufficient (0 pts)** | **Needs Development**  **(1-2 pts)** | **Sufficient (3-4 pts)** | **Excellent (5 pts)** | **Marks** | **X** |
| **Input / Output** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | * All team and player inputs can be successfully entered, and use descriptive prompts |  |  |
| * The report output lines are well-formatted and contain all expected information * All output values are formatted, where appropriate, using proper currency formatting (e.g. preceded by a $ symbol, two decimal places) |
| **OOP** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Solution displays strong understanding of OOP fundamentals. Classes are created, with all required properties and methods as expected. Constructors are included for each class (except Main) and used appropriately. Object instantiation done correctly and in the appropriate class (Main). |  | 2 |
| **Data Validation** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Data input is validated properly: integers must be equal to zero or above. Strings must be at least 3 characters. |  |  |
| **Random Budget** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | A random budget is generated for each team and is in the correct range. |  |  |
| **Team Rating** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Each team is assigned the correct rating, according to program requirements. Output is given with the right number of stars. |  |  |
| **Team Report** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | The team report is present and displays expected report data. Output is well-formatted, clearly labeled and has an easily readable layout. |  |  |
| **Player Report** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | The player report is present and displays expected report data. Output is well-formatted, clearly labeled and has an easily readable layout. |  |  |
| **Array & Object Usage** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Arrays are used to store objects as expected. Proper interaction with object arrays is demonstrated. |  |  |
| **Comments & Best Coding Practices**  (At least 60% of the functional requirements must be complete) | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Organizational or explanatory comments are used extensively, most are meaningful and easily understood. A consistent naming convention was used for most of the program and deviated very little. Source code was clean, consistently well-formatted and easy to read. |  |  |
|  |  | |  |  | **Total:** |  | **/50** |