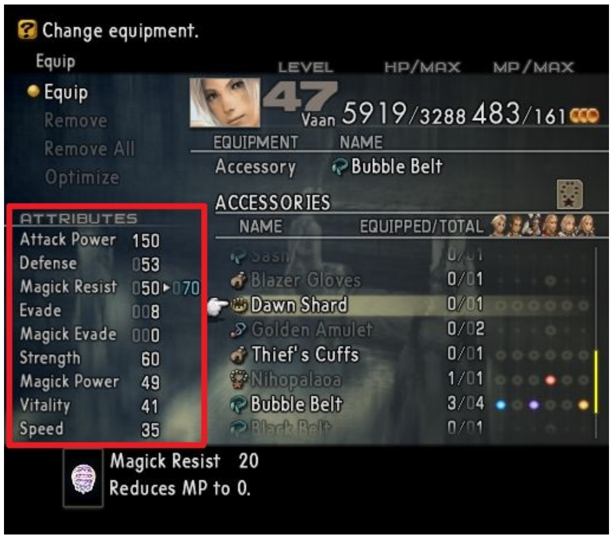
CS161: Introduction to Computer Science
Homework Assignment 8
Due: Friday 4/12 by 11:59pm

Team versus Team

I'm currently playing Final Fantasy XII (a video game). In the game, there are various characters each with their own stats. For example, one of the main characters in the game is called Vaan. Here's a snapshot of Vaan's stats (shown in the red box):



ATTRIBUTES	
Attack Power	150
Defense	053
Magick Resist	050▸070
Evade	008
Magick Evade	000
Strength	60
Magick Power	49
Vitality	41
Speed	35

At this point in the game, Vaan is at level 47 and he has attack power 150, defense 53, magick resist 50, evade 8, magick evade 0, and so on. As Vaan continues to level up, his stats will increase. Other than Vaan, there are 5 other main characters in the game [1] – each with their own level and stats [2].

One action that I find myself doing again and again is splitting the characters up into teams because, although there are 6 main characters, only 3 can be active at any one time. In fact, the need to split characters into teams transcends Final Fantasy. I can think of other video games, and other superhero comics/movies, where characters are constantly being split into teams (to fight against one another or to fight with one another).

Your assignment this week is to write a program that reads in various character names and stats and then randomly splits the characters into two teams. The user should be able to add characters, display all of the characters added so far, and generate a random team. In fact, this should be in a loop so that the user can generate as many random teams as they want. In addition, when you print the teams, you should also print the average of the team's stats so we can compare the two teams. Below is an example of what my program prints.

[1] The other 5 characters are: Penelo, Balthier, Fran, Basche, and Ashe. I don't know if there will be more characters added to the party. If you do know, don't spoil the game for me!

[2] This notion of characters having stats is not unique to Final Fantasy. Lots of video games, comic books, etc. follow the same template.

```
====== Menu ======
1. Add a player
2. See all players
3. Generate team
4. Quit
Choose an option: 1

Enter player’s name: A
Enter player’s attack power: 1
Enter player’s magick power: 1

====== Menu ======
1. Add a player
2. See all players
3. Generate team
4. Quit
Choose an option: 1

Enter player’s name: B
Enter player’s attack power: 2
Enter player’s magick power: 2

[...OMITTING...]

====== Menu ======
1. Add a player
2. See all players
3. Generate team
4. Quit
Choose an option: 2

[A, attack=1, magick=1]
[B, attack=2, magick=2]
[C, attack=3, magick=3]
[D, attack=4, magick=4]

====== Menu ======
1. Add a player
2. See all players
3. Generate team
4. Quit
Choose an option: 3

Team A:
[C, attack=3, magick=3]
[D, attack=4, magick=4]
Average attack: 3.5
Average magick: 3.5

Team B:
[A, attack=1, magick=1]
[B, attack=2, magick=2]
Average attack: 1.5
Average magick: 1.5
```

```
====== Menu ======
1. Add a player
2. See all players
3. Generate team
4. Quit
Choose an option: 3

Team A:
[B, attack=2, magick=2]
[C, attack=3, magick=3]
Average attack: 2.5
Average magick: 2.5

Team B:
[A, attack=1, magick=1]
[D, attack=4, magick=4]
Average attack: 2.5
Average magick: 2.5

====== Menu ======
1. Add a player
2. See all players
3. Generate team
4. Quit
Choose an option: 4
Goodbye!
```

Notice I kept my program simple and only kept track of two stats: attack power and magic power. Feel free to come up with your own stats. So, to summarize:

- Your menu should include options for adding a player, displaying all players, generating a team, and quitting
- If there are fewer than 2 players, please print an error message when the user tries to generate a team
- The number of players on each team should be as even as possible. (If there are an odd number of players, then one team will have one more player than the other)

The structure of this program is up to you! You can use as many (or as few) classes as you need. You can name your class(es) whatever is appropriate. Most likely, you will need to use `ArrayList`, `Scanner`, and `Random` as well as loops and conditionals. Your code should still adhere to the style guide (see below).

Style Guide

Before you submit your assignment, go through the checklist below and make sure your code conforms to the style guide.

Checklist

- ☐ All unused variables are deleted
- ☐ All instance variables are used in more than one method (if not, make them local)
- ☐ All instance variables are declared private
- ☐ All instance variables are initialized in the constructor
- ☐ Javadoc comment for all classes
- ☐ All methods have Javadoc comments (except for the `main` method)
- ☐ Good use of private methods

☐ Proper capitalization of variables (final and non-final variables), methods, and classes
☐ All numbers have been replaced with constants (i.e. a final variable)
☐ Use white space to separate different sections of your code
☐ Code is correctly indented to improve readability

See the “Style Guide” (under “Resources” on the course website) for more detailed information.

Submitting your assignment

You should submit your `hw8` folder with your all of your code via Moodle.