

# Programming Assignment:

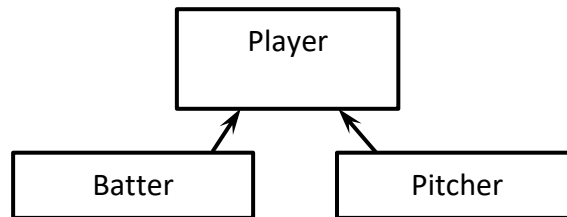
## Player Inheritance

**Copyright Statement:**

© Tanuja Joshi, 2024

*This assignment is protected by copyright. I am the exclusive owner of copyright in the materials I create.*

In this assignment, you will be creating an inheritance hierarchy of classes used to handle (simplified) data for Baseball players in a team. The class hierarchy is shown in the following diagram:



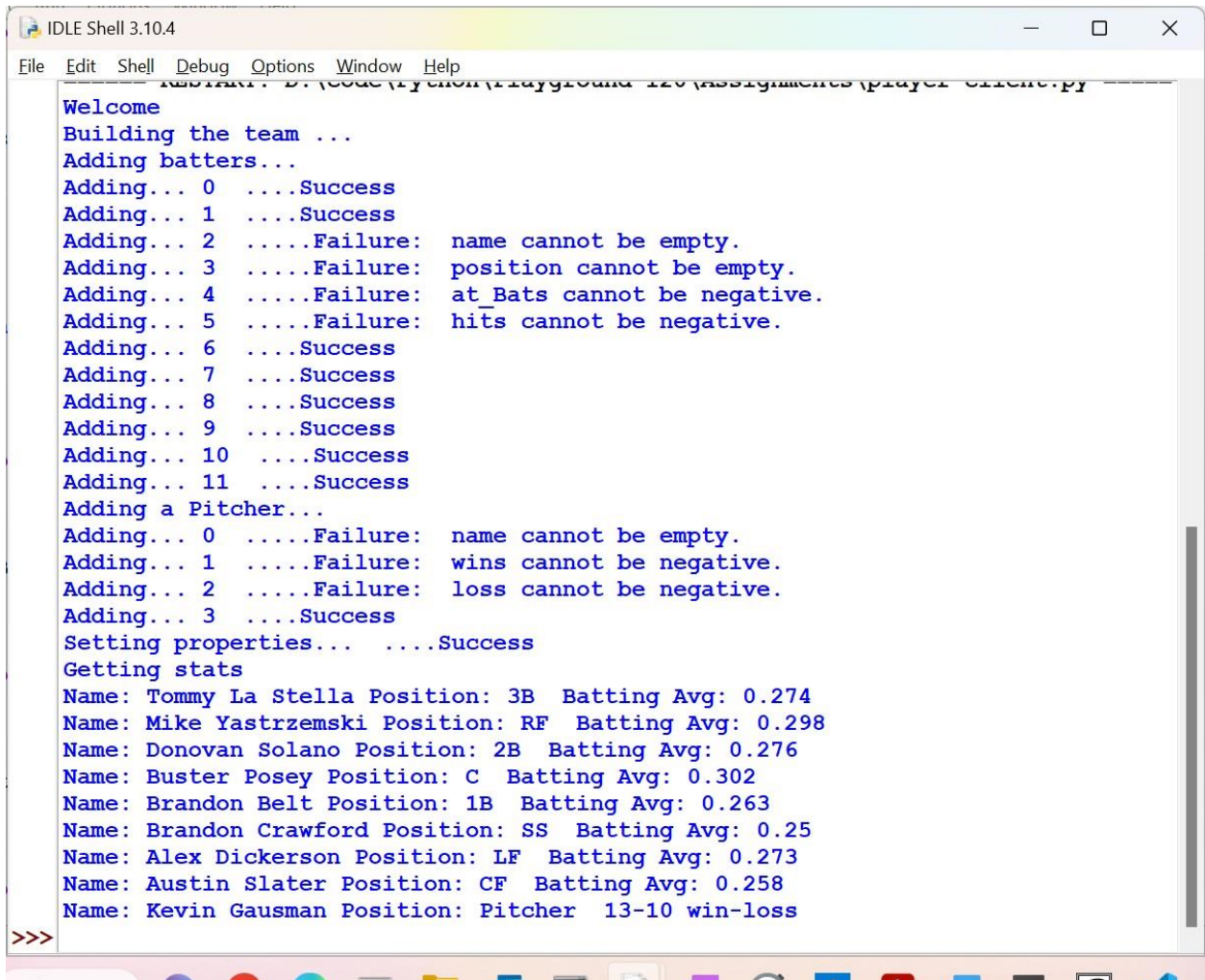
You will write definitions for all the three classes. In doing so, you will make sure that the hierarchy is maintained, and any attributes/ code that is common is placed in the base class so it can be shared. Feel free to add helper methods to reduce code duplication.

Specifically,

1) Add a base class called **Player**:

- a. Add a private attribute called **\_\_name** which represents the name of the player.
- b. Add a private attribute called **\_\_position** which represents the position of the player on the team.
- c. Add a constructor that takes two arguments called name and position. The constructor should set the corresponding private attributes based on these arguments. It should validate and throw a ValueError exception in case either is an empty string.
- d. Add two read-only public properties called **name** and **position** backed by the corresponding private attributes.
- e. Add a **getStats** method that takes no arguments and returns a string of the form "Player Name: *name* Position: *position*".

- f. Add a **Pitcher** class which is derived from the **Player** class. Add two public attributes called **wins** and **loss** that represent the number of game wins and game losses credited to the pitcher.
  - g. Add a constructor that takes three parameters: name, wins and loss. The constructor should set corresponding attributes. It should throw an exception if the name is empty or either of wins or loss arguments is < 0. It should call the base class constructor to set the name and the position attribute (as "Pitcher") in the base class.
  - h. Add a **getStats** method that returns a string that says: "Player Name: *name* Position: Pitcher Stats: *wins-loss* win-loss." where the value of *name*, *wins* and *loss* come from its corresponding attributes. This method should use the getStats method of the base class method. See the sample run below.
- 2) Add a **Batter** class which is also derived from the Player class.
- a. Add two public attributes **at\_bats** (represents the number of times the batter got a turn batting against a pitcher) and **hits** (represents the number of hits for the batter)
  - b. Add a constructor that takes four parameters: name, position, at\_bats and hits. The constructor should throw an exception if either name or position is an empty string. Or if either at\_bats or hits is negative.
  - c. Add a read-only public property called average that returns the batter's batting average. The formula for calculating batting average is:
$$\text{average} = \text{hits} / \text{at\_bats}$$
  - d. Add a **getStats** method that returns a string that says: "Player Name: *name* Position: *position* Stats: Batting Avg: *avg*." where the value of *name*, *position* and *avg* come from the corresponding properties. This method should use the getStats method of the base class method.
- 3) Write all the classes in a single file called player.py. Now run the provided playerClient.py file. It should run without errors and should give the output as shown in the screenshot of the sample run below. **NOTE: you shouldn't modify the client code. It just has to run correctly and give output as shown in the screenshot below.**



```

Welcome
Building the team ...
Adding batters...
Adding... 0 ....Success
Adding... 1 ....Success
Adding... 2 .....Failure: name cannot be empty.
Adding... 3 .....Failure: position cannot be empty.
Adding... 4 .....Failure: at_Bats cannot be negative.
Adding... 5 .....Failure: hits cannot be negative.
Adding... 6 ....Success
Adding... 7 ....Success
Adding... 8 ....Success
Adding... 9 ....Success
Adding... 10 ....Success
Adding... 11 ....Success
Adding a Pitcher...
Adding... 0 .....Failure: name cannot be empty.
Adding... 1 .....Failure: wins cannot be negative.
Adding... 2 .....Failure: loss cannot be negative.
Adding... 3 ....Success
Setting properties... ....Success
Getting stats
Name: Tommy La Stella Position: 3B Batting Avg: 0.274
Name: Mike Yastrzemski Position: RF Batting Avg: 0.298
Name: Donovan Solano Position: 2B Batting Avg: 0.276
Name: Buster Posey Position: C Batting Avg: 0.302
Name: Brandon Belt Position: 1B Batting Avg: 0.263
Name: Brandon Crawford Position: SS Batting Avg: 0.25
Name: Alex Dickerson Position: LF Batting Avg: 0.273
Name: Austin Slater Position: CF Batting Avg: 0.258
Name: Kevin Gausman Position: Pitcher 13-10 win-loss
>>>

```

Submit the program in a file named `first_last_player.py`. Make sure to add a block-comment at the start of the file that lists assignment title, class name, date, your name, and assignment description. Follow naming conventions.