CMSC 215 Intermediate Programming
Programming Project 1

The first programming project involves writing a program to find the tallest basketball player whose age is less than or equal to the average of all the players. The program should contain three classes. The first class `Height` should contain two integer instance variables for the feet and inches. The class should be immutable, so it should have no setter methods. At a minimum it should contain the following methods:

- A constructor that accepts feet and inches constructs a `Height` object
- A method `toInches` that returns the height in total inches
- A method `toString` that returns the string representation of the height with a single quote following the feet and a double quote following the inches

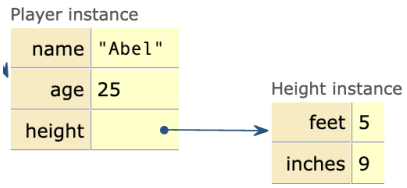
Regardless of what value for inches was supplied to the constructor the `toString` method should display the height normalized so the inches are less than 12. HINT: It is more efficient to normalize once within the constructor rather than each time toString is called.

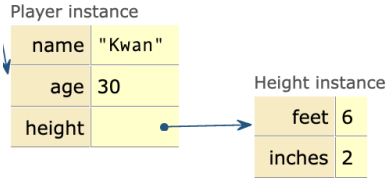
The second class `Player` should contain three instance variables that include the player's name, the player's height, which should be stored as the type `Height`, and the player's age.
The class should be immutable, so it should have no setter methods. It should contain the following methods:

- A constructor that accepts a player's name, height and age constructs a `Player` object
- Getter methods for each of the instance variables
- A method `toString` that returns the string representation of a player with each field appropriately labeled

The `toString` method of `Player` class should call the `toString` method of the `Height` class.

The relationship between Player and Height is aggregation (has-a), not inheritance (is-a). *A player has a height.* A sample object diagram of two players and associated heights is shown below.





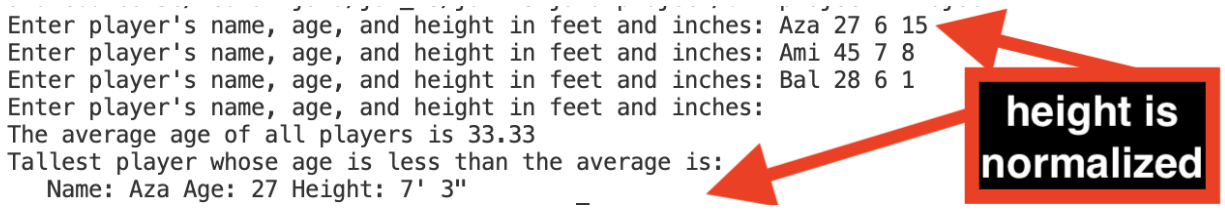
The third class `Project1` should repeatedly prompt the user for the information for each of the players until an empty string is input. It should create a `Player` object for each player and add

the player to an `ArrayList`. As the players are read in, the total age of all players should be computed to enable the average to be calculated. Once all player information is entered, the program should calculate and display the average age of all players. It should then search through the list to find the tallest player whose age is less than or equal to the average. The average should only be calculated if at least one player has been entered. If the ArrayList is empty, print "No player data entered."

Here is a sample run of the program:

```
Enter player's name, age, and height in feet and inches: Pavel 25 7 2
Enter player's name, age, and height in feet and inches: Lou 47 6 8
Enter player's name, age, and height in feet and inches: Kwan 37 6 2
Enter player's name, age, and height in feet and inches: Amir 60 6 0
Enter player's name, age, and height in feet and inches: Dai 20 7 4
Enter player's name, age, and height in feet and inches:
The average age of all players is 37.80
Tallest player whose age is less than the average is:
   Name: Dai Age: 20 Height: 7' 4"
```

Here is another sample run of the program that demonstrates normalization of the height. The user enters 6 15, while the output is 7' 3":



Here is a sample run where no player data is entered.

```
Enter player's name, age, and height in feet and inches:
No player data entered.
```

Documentation Requirements:
Make sure your Java program is using the recommended style such as:

- Javadoc comment with your name as author, date, and brief purpose of the program
- Comments for variables and blocks of code to describe major functionality
- Meaningful variable names and prompts
- Class names are written in upper CamelCase
- Variable and method names are written in lower CamelCase
- Constants are written in All Capitals
- Use proper spacing and empty lines to make your source code human readable

You are to submit two files.

1. The first is a `.zip` file that contains all the source code for the project.

a. The `.zip` file should contain only source code and nothing else, which means only the `.java` files.
b. If you elect to use a package the `.java` files should be in a folder whose name is the package name.
c. Every outer class should be in a separate `.java` file with the same name as the class name.
d. Each file should include a comment block at the top containing your name, the project name, the date, and a short description of the class contained in that file.

2. The second is a Word document (PDF or RTF is also acceptable) that contains the documentation for the project, which should include the following:
    a. A discussion of approach, which should outline the project design and explain the process you followed to implement the project. If the development was done in stages or over multiple versions, outline the focus and changes introduced in each version. Additionally, discuss the class design and relationships between the classes, along with the steps taken to implement and test these classes.
    b. A UML class diagram that includes all classes you wrote. Do not include predefined classes.
        - Use correct symbols to denote any class relationships.
    c. A test plan that includes test cases that you have created indicating what aspects of the program each one is testing. Include the results of your testing with screen captures clearly showing the output for each test case.
        - At least one test case should demonstrate normalization of the height.
        - Include a test case where no player data is entered.
    d. A short paragraph on lessons learned from the project.

Resources

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/