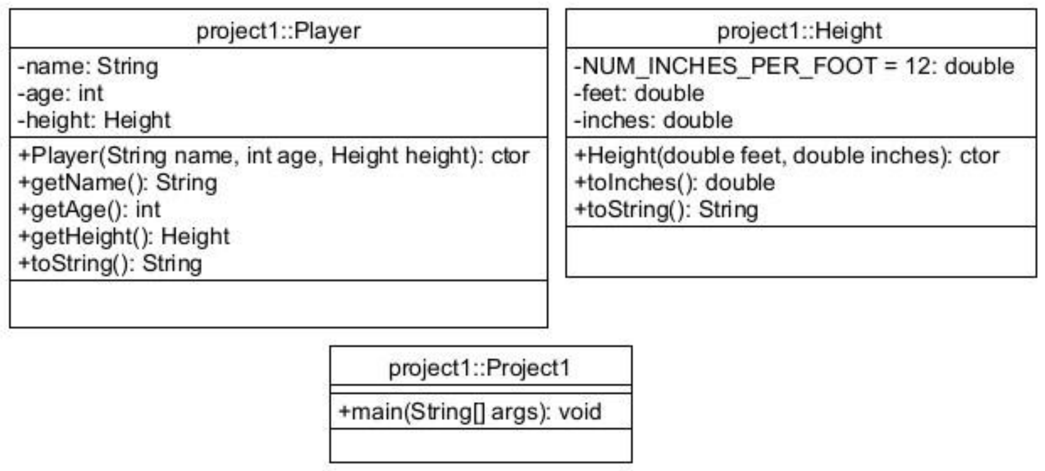
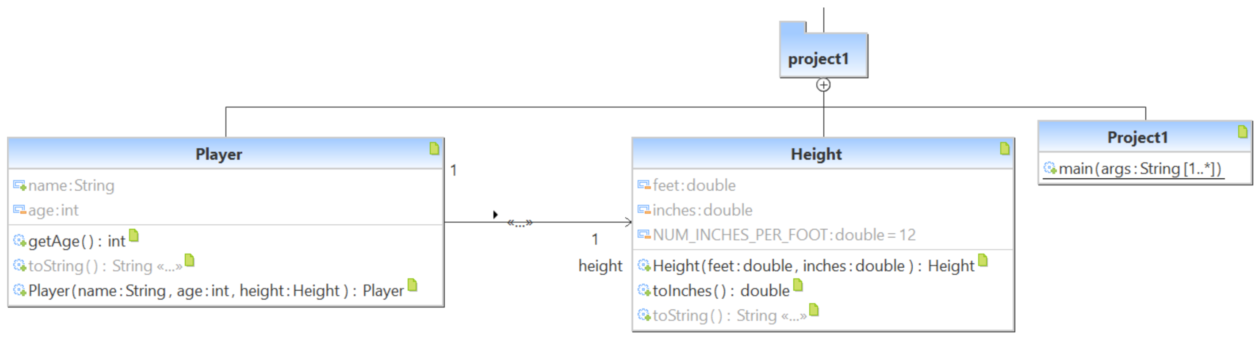
Victoria Lee
CMSC 215
5/27/2024

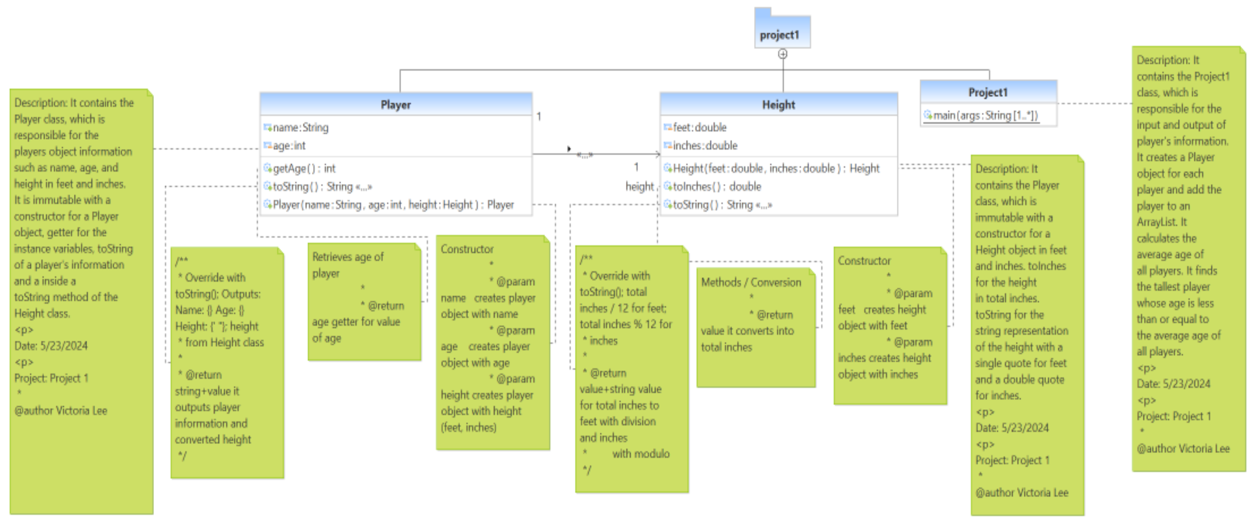
Project 1 Documentation

UML Class Diagrams and Package:




project1
Player
name:String
age:int
getAge() : int
toString() : String «...»
Player(name:String, age:int, height:Height) : Player
1
«...»
1
height
Height
feet:double
inches:double
NUM_INCHES_PER_FOOT:double = 12
Height(feet:double, inches:double) : Height
toInches() : double
toString() : String «...»
Project1
main(args:String[1..*])

UML Class Diagram and Package with descriptions:


project1
Player
name:String
age:int
getAge() : int
toString() : String «...»
Player(name:String, age:int, height:Height) : Player
1
«...»
1
height
Height
feet:double
inches:double
Height(feet:double, inches:double) : Height
toInches() : double
toString() : String «...»
Project1
main(args:String[1..*])

Description: It contains the Player class, which is responsible for the players object information such as name, age, and height in feet and inches. It is immutable with a constructor for a Player object, getter for the instance variables, toString of a player's information and a inside a toString method of the Height class.
<p>
Date: 5/23/2024
<p>
Project: Project 1
*
@author Victoria Lee

```
/**
 * Override with toString(): Outputs: Name: {} Age: {} Height: {' "}; height
 * from Height class
 *
 * @return string+value it outputs player information and converted height
 */
```

Retrieves age of player
*
* @return age getter for value of age

Constructor
*
* @param name creates player object with name
* @param age creates player object with age
* @param height creates player object with height (feet, inches)

```
/**
 * Override with toString(); total inches / 12 for feet; total inches % 12 for
 * inches
 *
 * @return value+string value for total inches to feet with division and inches
 *         with modulo
 */
```

Methods / Conversion
*
* @return value it converts into total inches

Constructor
*
* @param feet creates height object with feet
* @param inches creates height object with inches

Description: It contains the Player class, which is immutable with a constructor for a Height object in feet and inches. toInches for the height in total inches. toString for the string representation of the height with a single quote for feet and a double quote for inches.
<p>
Date: 5/23/2024
<p>
Project: Project 1
*
@author Victoria Lee

Description: It contains the Project1 class, which is responsible for the input and output of player's information. It creates a Player object for each player and add the player to an ArrayList. It calculates the average age of all players. It finds the tallest player whose age is less than or equal to the average age of all players.
<p>
Date: 5/23/2024
<p>
Project: Project 1
*
@author Victoria Lee

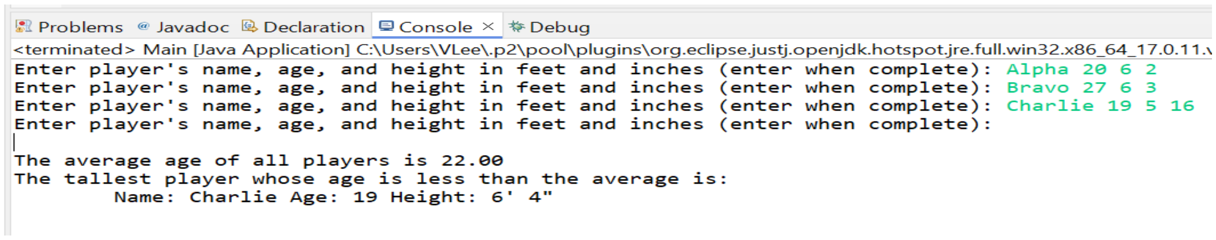
All Test Plans/Cases:

Test 1 (Given):

Input:
Alpha 20 6 2
Bravo 27 6 3
Charlie 19 5 16

Output:
The average age of all players is 22.00
The tallest player whose age is less than the average is:
    Name: Charlie Age: 19 Height: 6’ 4”

Demo test below:



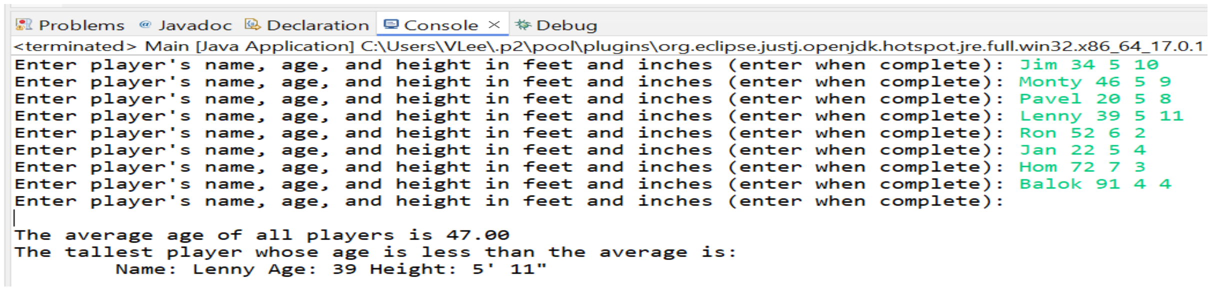
```
<terminated> Main [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.11.
Enter player's name, age, and height in feet and inches (enter when complete): Alpha 20 6 2
Enter player's name, age, and height in feet and inches (enter when complete): Bravo 27 6 3
Enter player's name, age, and height in feet and inches (enter when complete): Charlie 19 5 16
Enter player's name, age, and height in feet and inches (enter when complete):

The average age of all players is 22.00
The tallest player whose age is less than the average is:
        Name: Charlie Age: 19 Height: 6' 4"
```

Test 2 (Given):

Input:
Jim 34 5 10
Monty 46 5 9
Pavel 20 5 8
Lenny 39 5 11
Ron 52 6 2
Jan 22 5 4
Hom 72 7 3
Balock 91 4 4

Output:
The average age of all players is 47.00
The tallest player whose age is less than the average is:
    Name: Lenny Age: 39 Height: 5’ 11”

Demo test below:



```
<terminated> Main [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1
Enter player's name, age, and height in feet and inches (enter when complete): Jim 34 5 10
Enter player's name, age, and height in feet and inches (enter when complete): Monty 46 5 9
Enter player's name, age, and height in feet and inches (enter when complete): Pavel 20 5 8
Enter player's name, age, and height in feet and inches (enter when complete): Lenny 39 5 11
Enter player's name, age, and height in feet and inches (enter when complete): Ron 52 6 2
Enter player's name, age, and height in feet and inches (enter when complete): Jan 22 5 4
Enter player's name, age, and height in feet and inches (enter when complete): Hom 72 7 3
Enter player's name, age, and height in feet and inches (enter when complete): Balok 91 4 4
Enter player's name, age, and height in feet and inches (enter when complete):

The average age of all players is 47.00
The tallest player whose age is less than the average is:
        Name: Lenny Age: 39 Height: 5' 11"
```

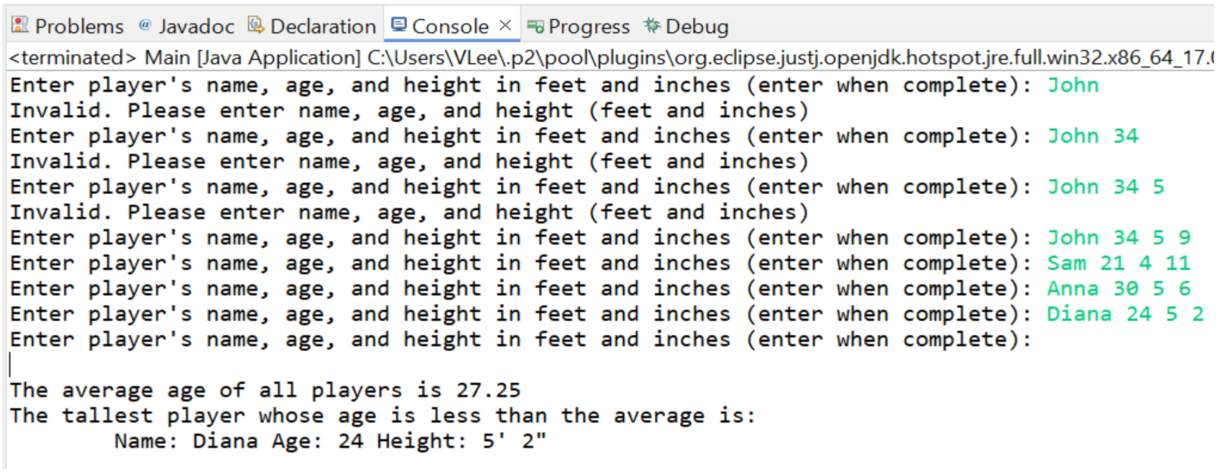
Test 3 (Created):

Note: I purposely used John as an example to make sure each user inputs will have four inputs on the line and if it is not four inputs it must not include it on the Player ArrayList and ask to re-enter the player again.

Input:
John
John 34
John 34 5
John 34 5 9
Sam 21 4 11
Anna 30 5 6
Diana 24 5 2

Output:
The average age of all players is 27.25
The tallest player whose age is less than the average is:
    Name: Diana Age: 24 Height: 5’ 2”

Demo test below:



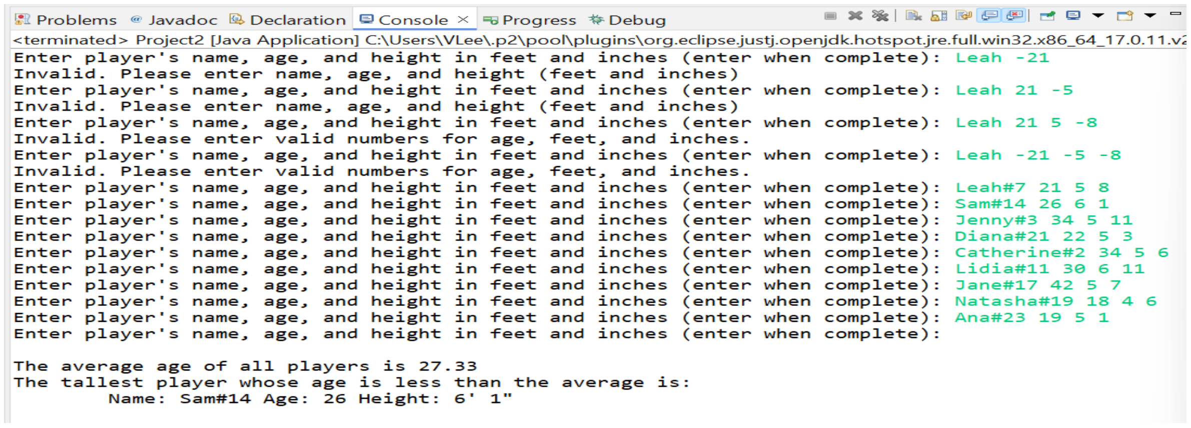
```
<terminated> Main [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.
Enter player's name, age, and height in feet and inches (enter when complete): John
Invalid. Please enter name, age, and height (feet and inches)
Enter player's name, age, and height in feet and inches (enter when complete): John 34
Invalid. Please enter name, age, and height (feet and inches)
Enter player's name, age, and height in feet and inches (enter when complete): John 34 5
Invalid. Please enter name, age, and height (feet and inches)
Enter player's name, age, and height in feet and inches (enter when complete): John 34 5 9
Enter player's name, age, and height in feet and inches (enter when complete): Sam 21 4 11
Enter player's name, age, and height in feet and inches (enter when complete): Anna 30 5 6
Enter player's name, age, and height in feet and inches (enter when complete): Diana 24 5 2
Enter player's name, age, and height in feet and inches (enter when complete):

The average age of all players is 27.25
The tallest player whose age is less than the average is:
        Name: Diana Age: 24 Height: 5' 2"
```

Test 4 (Created):

Note: I purposely used Leah as an example to make sure each input will have four inputs and the number values for age, feet, and inches must be positive only. If it is not four inputs or positive numbers, it must not include it on the Player ArrayList and ask to re-enter the player again. I did not put a coding for the string name because, sometimes some names have numbers in them. For instance, a basketball player's name may have the number next to the name to denote the player's number for the team (number on the basketball player's shirt). And so, I did not include that test case for the coding. This also tests the max age and max height in feet for a player. The tallest person in the world is around 9 feet, and the oldest person in the world to live is 115. I did not inches in case the user wants to input the inches of the player instead of the feet. Max inches demo test is in the last test case 13. For instance, 0 feet, 60 inches = 5 feet. That scenario is tested in a different test case.

Input:
Leah
Leah -21
Leah 21 -5
Leah 21 5 -8
Leah -21 -5 -8
Leah#7 21 5 8
Sam#14 26 6 1
Jenny#3 34 5 11
Diana#21 22 5 3
Catherine#2 34 5 6
Lidia#11 30 6 11
Jane#17 42 5 7
Natasha#19 18 4 6
Ana#23 19 5 1

Output:
The average age of all players is 27.33
The tallest player whose age is less than the average is:
Name: Sam#14 Age: 26 Height: 6' 1"

Demo test below:


```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.11.v2
Enter player's name, age, and height in feet and inches (enter when complete): Leah -21
Invalid. Please enter name, age, and height (feet and inches)
Enter player's name, age, and height in feet and inches (enter when complete): Leah 21 -5
Invalid. Please enter name, age, and height (feet and inches)
Enter player's name, age, and height in feet and inches (enter when complete): Leah 21 5 -8
Invalid. Please enter valid numbers for age, feet, and inches.
Enter player's name, age, and height in feet and inches (enter when complete): Leah -21 -5 -8
Invalid. Please enter valid numbers for age, feet, and inches.
Enter player's name, age, and height in feet and inches (enter when complete): Leah#7 21 5 8
Enter player's name, age, and height in feet and inches (enter when complete): Sam#14 26 6 1
Enter player's name, age, and height in feet and inches (enter when complete): Jenny#3 34 5 11
Enter player's name, age, and height in feet and inches (enter when complete): Diana#21 22 5 3
Enter player's name, age, and height in feet and inches (enter when complete): Catherine#2 34 5 6
Enter player's name, age, and height in feet and inches (enter when complete): Lidia#11 30 6 11
Enter player's name, age, and height in feet and inches (enter when complete): Jane#17 42 5 7
Enter player's name, age, and height in feet and inches (enter when complete): Natasha#19 18 4 6
Enter player's name, age, and height in feet and inches (enter when complete): Ana#23 19 5 1
Enter player's name, age, and height in feet and inches (enter when complete):

The average age of all players is 27.33
The tallest player whose age is less than the average is:
        Name: Sam#14 Age: 26 Height: 6' 1"
```

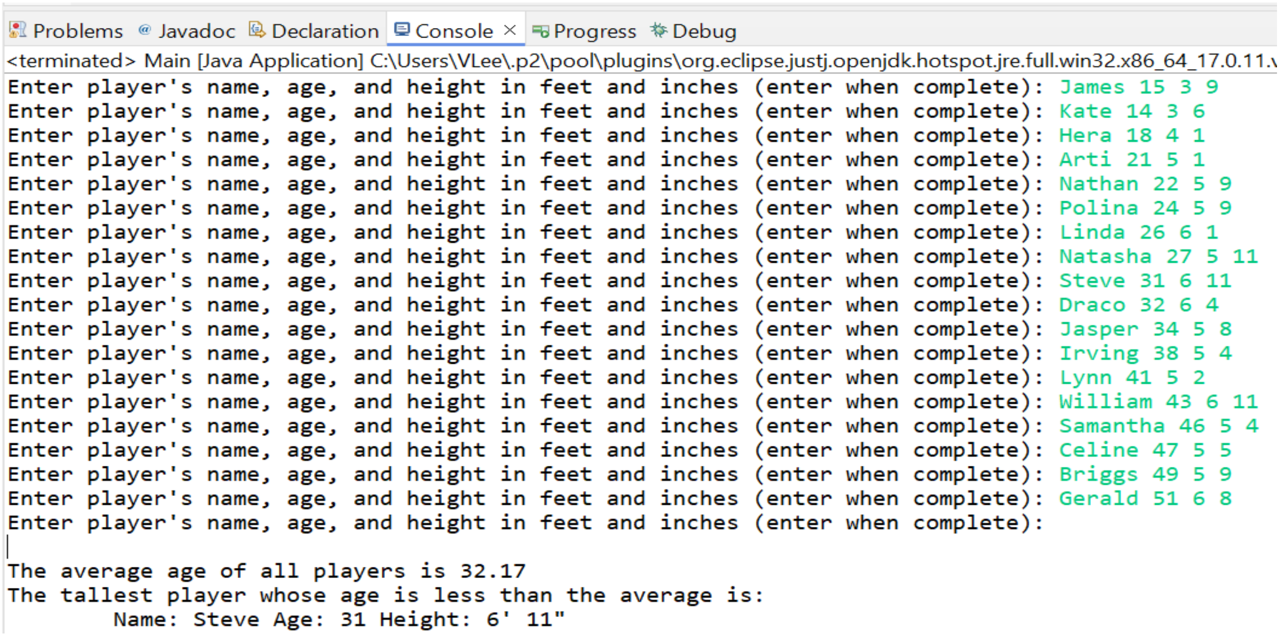
Test 5 (Created):

Note: Testing cases between ages 15-50 with mostly taller numbers this time for height.

Input:
James 15 3 9
Kate 14 3 6
Hera 18 4 1
Arti 21 5 1
Nathan 22 5 9
Polina 24 5 8
Linda 26 6 1
Natasha 27 5 11
Steve 31 6 11
Draco 32 6 4
Jasper 34 5 8
Irving 38 5 4
Lynn 41 5 2
William 43 6 11
Samantha 46 5 4
Celine 47 5 5
Briggs 49 5 9
Gerald 51 6 8

Output:
The average age of all players is 32.17
The tallest player whose age is less than the average is:
Name: Steve Age: 31 Height: 6’ 11”

Demo test below:



```
<terminated> Main [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.11.
Enter player's name, age, and height in feet and inches (enter when complete): James 15 3 9
Enter player's name, age, and height in feet and inches (enter when complete): Kate 14 3 6
Enter player's name, age, and height in feet and inches (enter when complete): Hera 18 4 1
Enter player's name, age, and height in feet and inches (enter when complete): Arti 21 5 1
Enter player's name, age, and height in feet and inches (enter when complete): Nathan 22 5 9
Enter player's name, age, and height in feet and inches (enter when complete): Polina 24 5 9
Enter player's name, age, and height in feet and inches (enter when complete): Linda 26 6 1
Enter player's name, age, and height in feet and inches (enter when complete): Natasha 27 5 11
Enter player's name, age, and height in feet and inches (enter when complete): Steve 31 6 11
Enter player's name, age, and height in feet and inches (enter when complete): Draco 32 6 4
Enter player's name, age, and height in feet and inches (enter when complete): Jasper 34 5 8
Enter player's name, age, and height in feet and inches (enter when complete): Irving 38 5 4
Enter player's name, age, and height in feet and inches (enter when complete): Lynn 41 5 2
Enter player's name, age, and height in feet and inches (enter when complete): William 43 6 11
Enter player's name, age, and height in feet and inches (enter when complete): Samantha 46 5 4
Enter player's name, age, and height in feet and inches (enter when complete): Celine 47 5 5
Enter player's name, age, and height in feet and inches (enter when complete): Briggs 49 5 9
Enter player's name, age, and height in feet and inches (enter when complete): Gerald 51 6 8
Enter player's name, age, and height in feet and inches (enter when complete):

The average age of all players is 32.17
The tallest player whose age is less than the average is:
        Name: Steve Age: 31 Height: 6' 11"
```

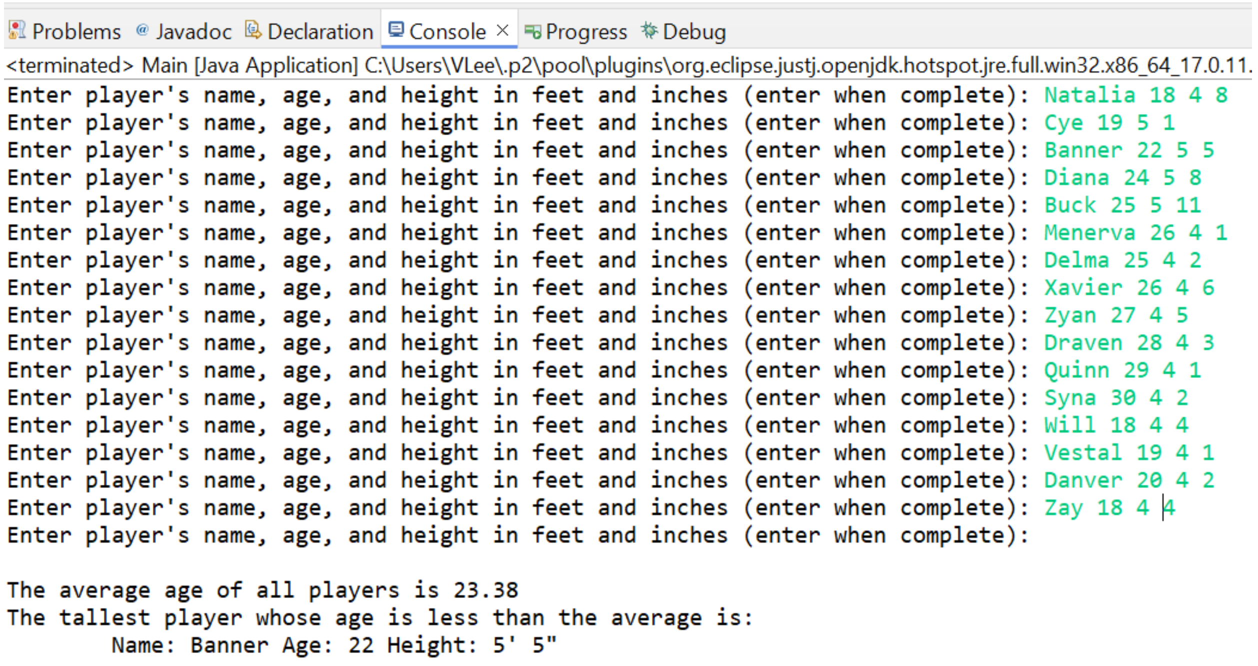
Test 6 (Created):

Note: Testing cases between ages 18-30 with mostly shorter numbers this time for height.

Input:
Natalia 18 4 8
Cye 19 5 1
Banner 22 5 5
Diana 24 5 8
Buck 25 5 11
Menerva 26 4 1
Delma 25 4 2
Xavier 26 4 6
Zyan 27 4 5
Draven 28 4 3
Quinn 29 4 1
Syna 30 4 2
Will 18 4 4
Vestal 19 4 1
Danver 20 4 2
Zay 18 4 4

Output:
The average age of all players is 23.38
The tallest player whose age is less than the average is:
Name: Banner Age: 22 Height: 5’ 5”

Demo test below:



```
<terminated> Main [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.11.
Enter player's name, age, and height in feet and inches (enter when complete): Natalia 18 4 8
Enter player's name, age, and height in feet and inches (enter when complete): Cye 19 5 1
Enter player's name, age, and height in feet and inches (enter when complete): Banner 22 5 5
Enter player's name, age, and height in feet and inches (enter when complete): Diana 24 5 8
Enter player's name, age, and height in feet and inches (enter when complete): Buck 25 5 11
Enter player's name, age, and height in feet and inches (enter when complete): Menerva 26 4 1
Enter player's name, age, and height in feet and inches (enter when complete): Delma 25 4 2
Enter player's name, age, and height in feet and inches (enter when complete): Xavier 26 4 6
Enter player's name, age, and height in feet and inches (enter when complete): Zyan 27 4 5
Enter player's name, age, and height in feet and inches (enter when complete): Draven 28 4 3
Enter player's name, age, and height in feet and inches (enter when complete): Quinn 29 4 1
Enter player's name, age, and height in feet and inches (enter when complete): Syna 30 4 2
Enter player's name, age, and height in feet and inches (enter when complete): Will 18 4 4
Enter player's name, age, and height in feet and inches (enter when complete): Vestal 19 4 1
Enter player's name, age, and height in feet and inches (enter when complete): Danver 20 4 2
Enter player's name, age, and height in feet and inches (enter when complete): Zay 18 4 4
Enter player's name, age, and height in feet and inches (enter when complete):

The average age of all players is 23.38
The tallest player whose age is less than the average is:
        Name: Banner Age: 22 Height: 5' 5"
```

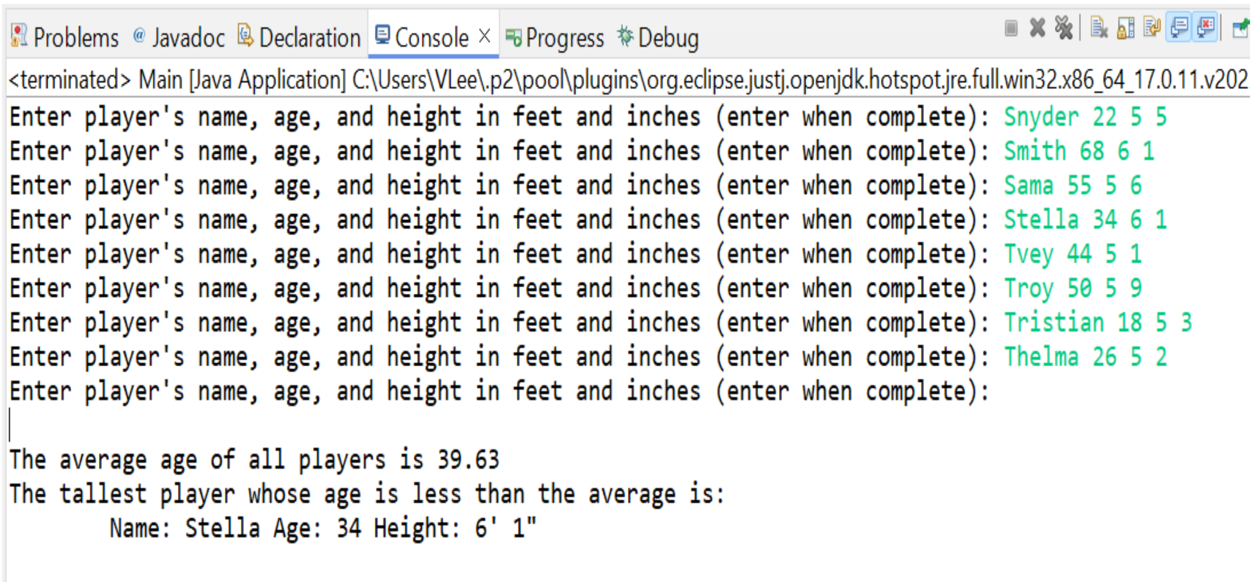
Test 7 (Created):

Note: Testing cases between random ages and height (names within s and t).

Input:
Snyder 22 5 5
Smith 68 6 1
Sama 55 5 6
Stella 34 6 1
Tvey 44 5 1
Troy 50 5 9
Tristian 18 5 3
Thelma 26 5 2

Output:
The average age of all players is 39. 63
The tallest player whose age is less than the average is:
    Name: Stella Age: 34 Height: 6’ 1”

Demo test below:



```
<terminated> Main [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.11.v202
Enter player's name, age, and height in feet and inches (enter when complete): Snyder 22 5 5
Enter player's name, age, and height in feet and inches (enter when complete): Smith 68 6 1
Enter player's name, age, and height in feet and inches (enter when complete): Sama 55 5 6
Enter player's name, age, and height in feet and inches (enter when complete): Stella 34 6 1
Enter player's name, age, and height in feet and inches (enter when complete): Tvey 44 5 1
Enter player's name, age, and height in feet and inches (enter when complete): Troy 50 5 9
Enter player's name, age, and height in feet and inches (enter when complete): Tristian 18 5 3
Enter player's name, age, and height in feet and inches (enter when complete): Thelma 26 5 2
Enter player's name, age, and height in feet and inches (enter when complete):

The average age of all players is 39.63
The tallest player whose age is less than the average is:
        Name: Stella Age: 34 Height: 6' 1"
```

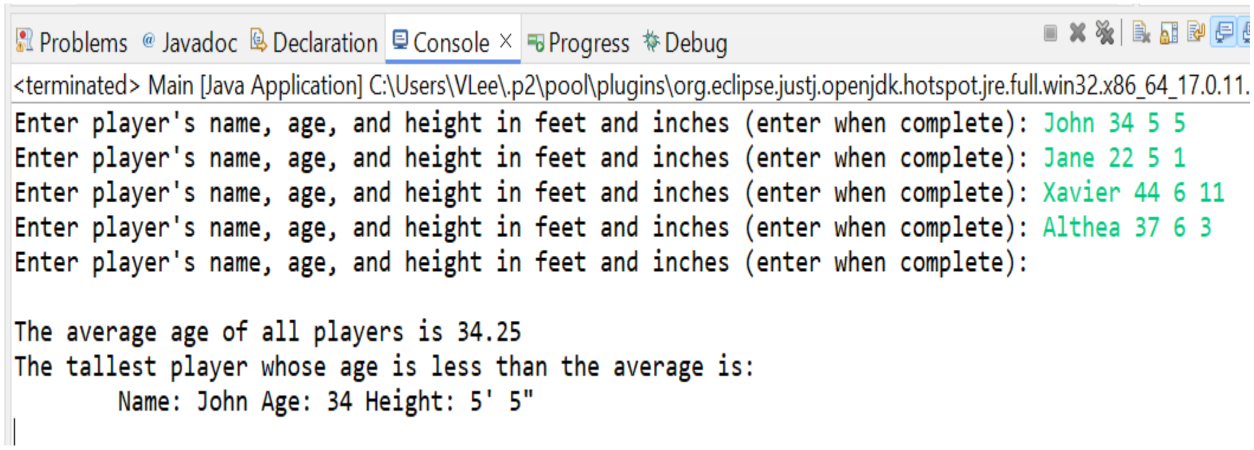
Test 8 (Created):

Note: Testing cases between random ages and height, simplistic four players approach.

Input:
John 34 5 5
Jane 22 5 1
Xavier 44 6 11
Althea 37 6 3

Output:
The average age of all players is 34. 25
The tallest player whose age is less than the average is:
Name: John Age: 34 Height: 5’ 5”

Demo test below:



```
<terminated> Main [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.11.
Enter player's name, age, and height in feet and inches (enter when complete): John 34 5 5
Enter player's name, age, and height in feet and inches (enter when complete): Jane 22 5 1
Enter player's name, age, and height in feet and inches (enter when complete): Xavier 44 6 11
Enter player's name, age, and height in feet and inches (enter when complete): Althea 37 6 3
Enter player's name, age, and height in feet and inches (enter when complete):

The average age of all players is 34.25
The tallest player whose age is less than the average is:
        Name: John Age: 34 Height: 5' 5"
```

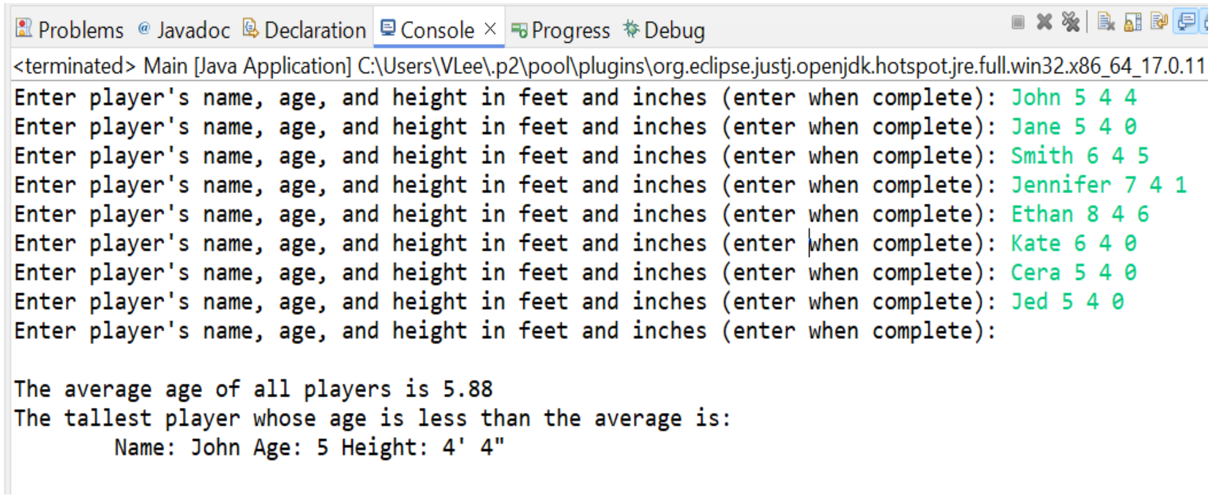
Test 9 (Created):

Note: Testing cases between smaller ages and height. In addition, it is also testing height numbers with 0 in them. For instance, Height is 4 feet 0 inches.

Input:
John 5 4 4
Jane 5 4 0
Smith 6 4 5
Jennifer 7 4 1
Ethan 8 4 6
Kate 6 4 0
Cera 5 4 0
Jed 5 4 0

Output:
The average age of all players is 5.88
The tallest player whose age is less than the average is:
Name: John Age: 5 Height: 4’ 4”

Demo test below:



```
<terminated> Main [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.11
Enter player's name, age, and height in feet and inches (enter when complete): John 5 4 4
Enter player's name, age, and height in feet and inches (enter when complete): Jane 5 4 0
Enter player's name, age, and height in feet and inches (enter when complete): Smith 6 4 5
Enter player's name, age, and height in feet and inches (enter when complete): Jennifer 7 4 1
Enter player's name, age, and height in feet and inches (enter when complete): Ethan 8 4 6
Enter player's name, age, and height in feet and inches (enter when complete): Kate 6 4 0
Enter player's name, age, and height in feet and inches (enter when complete): Cera 5 4 0
Enter player's name, age, and height in feet and inches (enter when complete): Jed 5 4 0
Enter player's name, age, and height in feet and inches (enter when complete):

The average age of all players is 5.88
The tallest player whose age is less than the average is:
        Name: John Age: 5 Height: 4' 4"
```

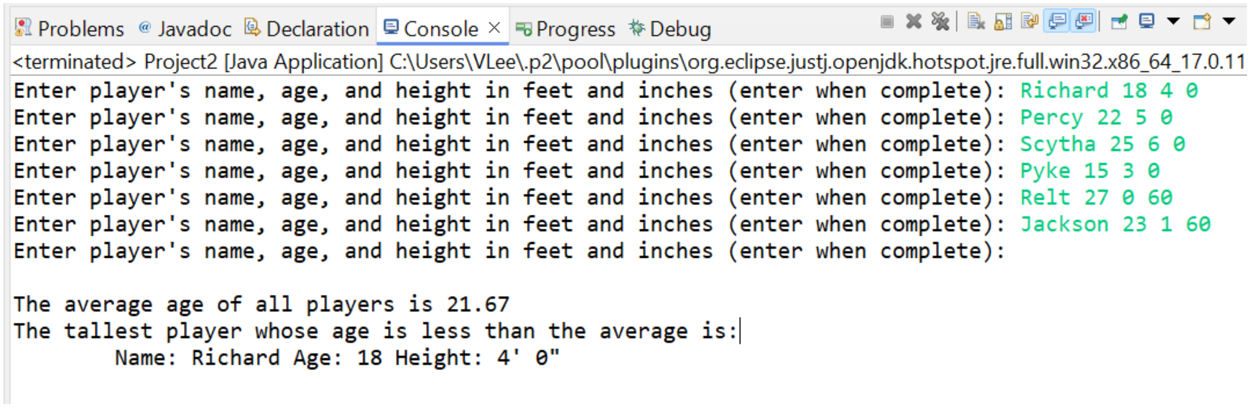
Test 10 (Created):

Note: Testing cases between random ages and height. In addition, it is also testing height numbers with 0 in them. For instance, Height is 0 feet and 60 inches.

Input:
Richard 18 4 0
Percy 22 5 0
Scytha 25 6 0
Pyke 15 3 0
Relt 27 0 60
Jackson 23 0 36

Output:
The average age of all players is 21.67
The tallest player whose age is less than the average is:
    Name: Richard Age: 18 Height: 4’ 0”

Demo test below:



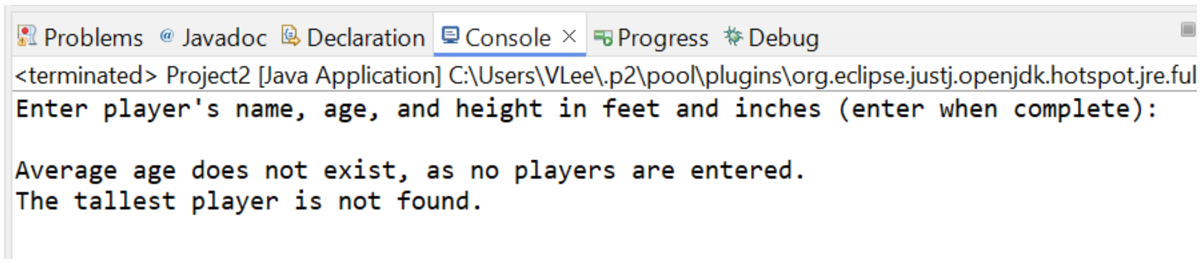
```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.11
Enter player's name, age, and height in feet and inches (enter when complete): Richard 18 4 0
Enter player's name, age, and height in feet and inches (enter when complete): Percy 22 5 0
Enter player's name, age, and height in feet and inches (enter when complete): Scytha 25 6 0
Enter player's name, age, and height in feet and inches (enter when complete): Pyke 15 3 0
Enter player's name, age, and height in feet and inches (enter when complete): Relt 27 0 60
Enter player's name, age, and height in feet and inches (enter when complete): Jackson 23 1 60
Enter player's name, age, and height in feet and inches (enter when complete):

The average age of all players is 21.67
The tallest player whose age is less than the average is:
        Name: Richard Age: 18 Height: 4' 0"
```

Test 11 (Created):

Note: Testing if the user does not input anything and completes it. If it does not input anything and the user ends the program, the code makes sure to output a different message for when the average outputs NaN. It should say it does not exist to computer calculations.

Input:
Enter key to complete (no input)
Output:
Average age does not exist, as no players are entered.
The tallest player is not found.

Demo test below:



```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.ful
Enter player's name, age, and height in feet and inches (enter when complete):

Average age does not exist, as no players are entered.
The tallest player is not found.
```

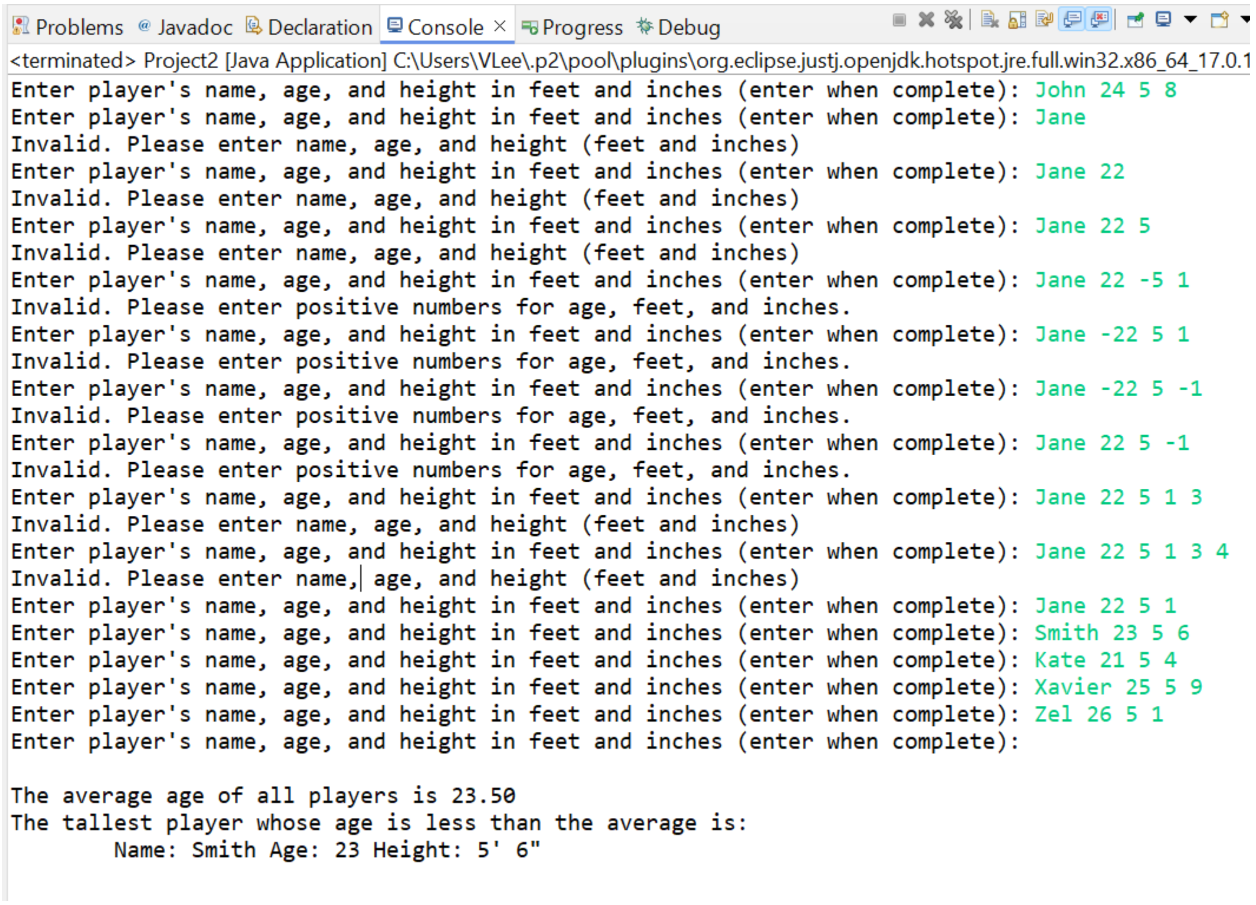
Test 12 (Created):

Note: Testing if the user inputs an invalid input after a valid input. It is also testing if the user entered only four requirements of string, number, number and number. If it does not enter correct number and not four of them it will pop up invalid. It will also notify the user if the entered data is greater than four inputs. For instance, the requirements is: Jane 22 5 1. If the user enters Jane 22 5 1 3 or Jane 22 5 1 3 4 that is in total of five or six inputs, which is an invalid input. Invalid inputs will ask the user to resubmit the correct player again with correct numbers.

Input:
John 24 5 8
Jane
Jane 22
Jane 22 5
Jane 22 -5 1
Jane -22 5 1
Jane -22 5 -1
Jane 22 5 -1
Jane 22 5 1 3
Jane 22 5 1 3 4
Jane 22 5 1
Smith 23 5 6
Kate 21 5 4
Xavier 25 5 9
Zel 26 5 1

Output:
The average age of all players is 23.50
The tallest player whose age is less than the average is:
    Name: Smith Age: 23 Height: 5’ 6”

Demo test below:


Problems Javadoc Declaration Console Progress Debug

```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1
Enter player's name, age, and height in feet and inches (enter when complete): John 24 5 8
Enter player's name, age, and height in feet and inches (enter when complete): Jane
Invalid. Please enter name, age, and height (feet and inches)
Enter player's name, age, and height in feet and inches (enter when complete): Jane 22
Invalid. Please enter name, age, and height (feet and inches)
Enter player's name, age, and height in feet and inches (enter when complete): Jane 22 5
Invalid. Please enter name, age, and height (feet and inches)
Enter player's name, age, and height in feet and inches (enter when complete): Jane 22 -5 1
Invalid. Please enter positive numbers for age, feet, and inches.
Enter player's name, age, and height in feet and inches (enter when complete): Jane -22 5 1
Invalid. Please enter positive numbers for age, feet, and inches.
Enter player's name, age, and height in feet and inches (enter when complete): Jane -22 5 -1
Invalid. Please enter positive numbers for age, feet, and inches.
Enter player's name, age, and height in feet and inches (enter when complete): Jane 22 5 -1
Invalid. Please enter positive numbers for age, feet, and inches.
Enter player's name, age, and height in feet and inches (enter when complete): Jane 22 5 1 3
Invalid. Please enter name, age, and height (feet and inches)
Enter player's name, age, and height in feet and inches (enter when complete): Jane 22 5 1 3 4
Invalid. Please enter name, age, and height (feet and inches)
Enter player's name, age, and height in feet and inches (enter when complete): Jane 22 5 1
Enter player's name, age, and height in feet and inches (enter when complete): Smith 23 5 6
Enter player's name, age, and height in feet and inches (enter when complete): Kate 21 5 4
Enter player's name, age, and height in feet and inches (enter when complete): Xavier 25 5 9
Enter player's name, age, and height in feet and inches (enter when complete): Zel 26 5 1
Enter player's name, age, and height in feet and inches (enter when complete):

The average age of all players is 23.50
The tallest player whose age is less than the average is:
        Name: Smith Age: 23 Height: 5' 6"
```

Test 13 (Created):

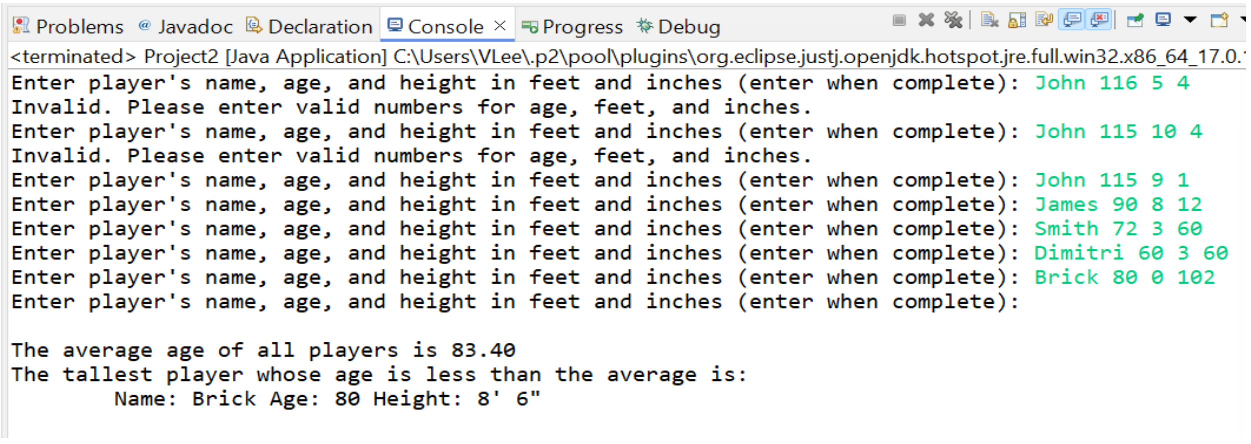
Note: This is a follow up to test case four. I am testing the max age and max height in feet for a player. The tallest person in the world is around 9 feet, and the oldest person in the world to live is 115. I did not include max inches in case the user wants to input the inches of the player instead of the feet. For instance, 0 feet, 60 inches = 5 feet. That scenario is tested in a different test case.

Input:

John 116 5 4
John 115 10 4
John 115 9 1
James 90 8 12
Smith 72 3 60
Dimitri 60 3 60
Brick 80 0 102

Output:
The average age of all players is 18.00
The tallest player whose age is less than the average is:
        Name: Brick Age: 80 Height: 8’ 6”

Demo test below:



```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.
Enter player's name, age, and height in feet and inches (enter when complete): John 116 5 4
Invalid. Please enter valid numbers for age, feet, and inches.
Enter player's name, age, and height in feet and inches (enter when complete): John 115 10 4
Invalid. Please enter valid numbers for age, feet, and inches.
Enter player's name, age, and height in feet and inches (enter when complete): John 115 9 1
Enter player's name, age, and height in feet and inches (enter when complete): James 90 8 12
Enter player's name, age, and height in feet and inches (enter when complete): Smith 72 3 60
Enter player's name, age, and height in feet and inches (enter when complete): Dimitri 60 3 60
Enter player's name, age, and height in feet and inches (enter when complete): Brick 80 0 102
Enter player's name, age, and height in feet and inches (enter when complete):

The average age of all players is 83.40
The tallest player whose age is less than the average is:
        Name: Brick Age: 80 Height: 8' 6"
```

Test 14 (Created):

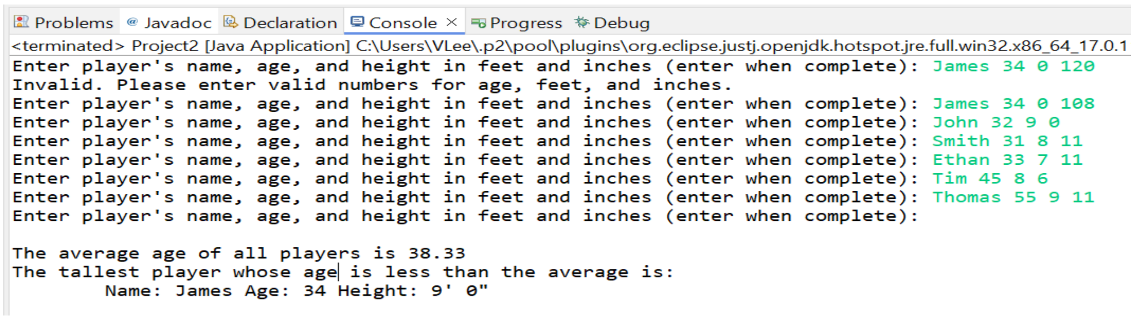
Note: This is a follow up to test case four. I am testing the height in inches for a player. The tallest person in the world is around 9 feet, the oldest person in the world to live is 115, and so the feet and inches maximum must be around 9 feet tall for basketball players. For instance, 0 feet, 60 inches = 5 feet. Or 0 feet, 108 inches = 9 feet. If the inches entered is greater than 108, it will ask the user to reinput the data.

Input:

James 34 0 120
James 34 0 108
John 32 9 0
Smith 31 8 11
Ethan 31 8 11
Tim 45 8 6
Thomas 55 9 11

Output:
The average age of all players is 38.33
The tallest player whose age is less than the average is:
Name: James Age: 34 Height: 9’ 0”

Demo test below:



```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1
Enter player's name, age, and height in feet and inches (enter when complete): James 34 0 120
Invalid. Please enter valid numbers for age, feet, and inches.
Enter player's name, age, and height in feet and inches (enter when complete): James 34 0 108
Enter player's name, age, and height in feet and inches (enter when complete): John 32 9 0
Enter player's name, age, and height in feet and inches (enter when complete): Smith 31 8 11
Enter player's name, age, and height in feet and inches (enter when complete): Ethan 33 7 11
Enter player's name, age, and height in feet and inches (enter when complete): Tim 45 8 6
Enter player's name, age, and height in feet and inches (enter when complete): Thomas 55 9 11
Enter player's name, age, and height in feet and inches (enter when complete):

The average age of all players is 38.33
The tallest player whose age is less than the average is:
        Name: James Age: 34 Height: 9' 0"
```

Lessons Learned (brief paragraphs):

I learned how to create a project, classes, methods, and functions to achieve my project goals. I learned to create objects for a solution design in Project 1 with some methods to perform on the attributes and classes which is a description of an object. An object is an instance of a class. A class defines all the attributes which an object can have and methods, which defines the functionality of the object. I learned more about encapsulation of important data such as information of an object and restricting access of the data and methods. I learned more about inheritance of classes in hierarchical manner. I also learned polymorphism which provides a mechanism where methods performing similar tasks but vary in arguments, can be assigned same name. For Project1 class utilizes both the Player and Height class to create a player and analyze the information given. I utilize the scanner for user input of the player's information for each object created and added to the Player Array List. I created a constructor and getters for all classes. I created methods and an override string for converting the height into total inches and then converting back into the new height. Then, I calculate the average age of all players and find the tallest player whose age is less than or equal to the average age of all players. In addition, I learned to utilize the lessons learned from the chapters learned for try/catch/exceptions/throws, classes, subclasses, packages, importing libraries, constructors, getters/setters, toString(), returning, private vs public, static vs non-static, Array List, Objects, Arrays, while loops, for loops, and getting values from the class using the ".". For instance, player.getHeight().toInches(). Overall, I learned how to utilize most of these lessons from the book and apply it to project 1.

My design approach was to create the Height and Player classes first before creating the Project1 class. I started with a Bottom-Up Design when building the code, but then debugged the code through a Top-Down Design. I followed the instructions on what is asked for both the Height and Player class. I utilized the lessons from the chapters that were in the past few weeks so that I can apply it to the Height and Player classes. Once it was finished, I went back into Project1 class to create the Player players Array List, find the sum and average the ages, and then find the tallest players according to the instructions. To debug the Project1, I ended up looking at the Excel test cases and then modified the Height and Player classes according to the formulas given. I also adjust the classes so that the toString() would override each other for the height values. Then, I went back to the Project1 class and checked if the output was correct. After I got the basic outputs, then I went back to modify in the Project1 class with the main method so that I can adjust for invalid inputs and create a code for them. Once I created the code, I went back to check if the outputs were correct.

(If necessary, my whole Javadoc to word is below)

Project1 Javadoc Combined Document

Table of Contents

JavaScript is disabled on your browser.

Skip navigation links

- Package

All Classes and Interfaces

Classes

Class

Description

Height

Description: It contains the Player class, which is immutable with a constructor for a Height object in feet and inches.

Player

Description: It contains the Player class, which is responsible for the players object information such as name, age, and height in feet and inches.

Project1

Description: It contains the Project1 class, which is responsible for the input and output of player's information.

All Packages

Package Summary

Package

Description

project1

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

Constant Field Values

Contents

project1.*

- project1.Height

Modifier and Type	Constant Field	Value
`private final double`	`NUM_INCHES_PER_FOOT`	`12.0`

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help
- Help:
- Navigation |
- Pages

JavaDoc Help

Navigation

Starting from the Overview page, you can browse the documentation using the links in each page, and in the navigation bar at the top of each page. The Index and Search box allow you to navigate to specific declarations and summary pages, including: All Packages, All Classes and Interfaces

Search

You can search for definitions of modules, packages, types, fields, methods, system properties and other terms defined in the API, using some or all of the name, optionally using "camelCase" abbreviations. For example:

- `j.l.obj` will match "java.lang.Object"
- `InpStr` will match "java.io.InputStream"
- `HM.cK` will match "java.util.HashMap.containsKey(Object)"

Refer to the Javadoc Search Specification for a full description of search features.

Kinds of Pages

The following sections describe the different kinds of pages in this collection.

Package

Each package has a page that contains a list of its classes and interfaces, with a summary for each. These pages may contain the following categories:

- Interfaces
- Classes
- Enum Classes
- Exceptions
- Errors
- Annotation Interfaces

Class or Interface

Each class, interface, nested class and nested interface has its own separate page. Each of these pages has three sections consisting of a declaration and description, member summary tables, and detailed member descriptions. Entries in each of these sections are omitted if they are empty or not applicable.

- Class Inheritance Diagram
- Direct Subclasses
- All Known Subinterfaces
- All Known Implementing Classes
- Class or Interface Declaration
- Class or Interface Description

- Nested Class Summary
- Enum Constant Summary
- Field Summary
- Property Summary
- Constructor Summary
- Method Summary
- Required Element Summary
- Optional Element Summary
- Enum Constant Details
- Field Details
- Property Details
- Constructor Details
- Method Details
- Element Details

Note: Annotation interfaces have required and optional elements, but not methods. Only enum classes have enum constants. The components of a record class are displayed as part of the declaration of the record class. Properties are a feature of JavaFX.

The summary entries are alphabetical, while the detailed descriptions are in the order they appear in the source code. This preserves the logical groupings established by the programmer.

Other Files

Packages and modules may contain pages with additional information related to the declarations nearby.

Use

Each documented package, class and interface has its own Use page. This page describes what packages, classes, methods, constructors and fields use any part of the given class or package. Given a class or interface A, its Use page includes subclasses of A, fields declared as A, methods that return A, and methods and constructors with parameters of type A. You can access this page by first going to the package, class or interface, then clicking on the USE link in the navigation bar.

Tree (Class Hierarchy)

There is a Class Hierarchy page for all packages, plus a hierarchy for each package. Each hierarchy page contains a list of classes and a list of interfaces. Classes are organized by inheritance structure starting with `java.lang.Object`. Interfaces do not inherit from `java.lang.Object`.

- When viewing the Overview page, clicking on TREE displays the hierarchy for all packages.
- When viewing a particular package, class or interface page, clicking on TREE displays the hierarchy for only that package.

Constant Field Values

The Constant Field Values page lists the static final fields and their values.

All Packages

The All Packages page contains an alphabetic index of all packages contained in the documentation.

All Classes and Interfaces

The All Classes and Interfaces page contains an alphabetic index of all classes and interfaces contained in the documentation, including annotation interfaces, enum classes, and record classes.

Index

The Index contains an alphabetic index of all classes, interfaces, constructors, methods, and fields in the documentation, as well as summary pages such as All Packages, All Classes and Interfaces.

This help file applies to API documentation generated by the standard doclet.

project1/package-summary.html

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

Hierarchy For All Packages

Package Hierarchies:

- project1

Class Hierarchy

- java.lang.Object
  - project1.Height
  - project1.Player
  - project1.Project1

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

Index

Index

Index

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

Index

Index

I

inches - Variable in class project1.Height
Double inches for Height

Index

M

main(String[]) - Static method in class project1.Project1
Project1 uses Height, Player classes; User inputs, create Player in ArrayList, compute averages and find tallest; has try/catch, else for invalid inputs

Index

N

name - Variable in class project1.Player
String name for Player

NUM_INCHES_PER_FOOT - Variable in class project1.Height
Constant for inches conversion to feet

Index

A F G H I M N P T
All Classes and Interfaces|All Packages|Constant Field Values

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

Index

Uses of Class project1.Height

- ## Uses of Height in project1

  Fields in project1 declared as Height

  Modifier and Type

  Field

  Description

  `private final Height`

  Player.height

  Height height for Player

  Methods in project1 that return Height

  Modifier and Type

  Method

  Description

  `Height`

  Player.getHeight()

  Retrieves height of player

  Constructors in project1 with parameters of type Height

  Modifier

  Constructor

  Description

  `Player(String name, int age, Height height)`

  Constructor

- Package
- Class
- Use
- Tree
- Index
- Help

Uses of Class project1.Player

No usage of project1.Player

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

Uses of Class project1.Project1

No usage of project1.Project1

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help
- Summary:
- Nested |
- Field |
- Constr |
- Method

- Detail:
- Field |
- Constr |
- Method

Package project1

Class Height

java.lang.Object

project1.Height

---

public class Height extends Object

Description: It contains the Player class, which is immutable with a constructor for a Height object in feet and inches. toInches for the height in total inches. toString for the string representation of the height with a single quote for feet and a double quote for inches.

Date: 5/23/2024

Project: Project 1

Version:
JRE17

Author:
Victoria Lee

- ### Field Summary

  Fields

Modifier and Type	Field	Description
`private final double`	`feet`	Double feet for Height
`private final double`	`inches`	Double inches for Height
`private final double`	`NUM_INCHES_PER_FOOT`	

Constant for inches conversion to feet

- ## Constructor Summary

  Constructors

  Constructor

  Description

  `Height(double feet, double inches)`

  Constructor

- ## Method Summary

  All Methods

  Instance Methods

  Concrete Methods

  Modifier and Type

  Method

  Description

  `double`

  `toInches()`

  Methods / Conversion

  `String`

  `toString()`

  Override with toString(); total inches / 12 for feet; total inches % 12 for inches

  ### Methods inherited from class java.lang.Object

  `clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

- ## Field Details

  - ### NUM_INCHES_PER_FOOT

    private final double NUM_INCHES_PER_FOOT

    Constant for inches conversion to feet

    See Also:

    - Constant Field Values

  - ### feet

    private final double feet

    Double feet for Height

  - ### inches

    private final double inches

Double inches for Height

- ## Constructor Details

  - ### Height

    public Height(double feet, double inches)

    Constructor

    Parameters:

    `feet` - creates height object with feet

    `inches` - creates height object with inches

- ## Method Details

  - ### toInches

    public double toInches()

    Methods / Conversion

    Returns:

    value it converts into total inches

  - ### toString

    public String toString()

    Override with toString(); total inches / 12 for feet; total inches % 12 for inches

    Overrides:

    `toString` in class `Object`

    Returns:

    value+string value for total inches to feet with division and inches with modulo

Package project1

---

package project1

- Classes
  Class
  Description
  Height
  Description: It contains the Player class, which is immutable with a constructor for a Height object in feet and inches.
  Player
  Description: It contains the Player class, which is responsible for the players object information such as name, age, and height in feet and inches.
  Project1
  Description: It contains the Project1 class, which is responsible for the input and output of player's information.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

Hierarchy For Package project1

Class Hierarchy

- java.lang.Object
  - project1.Height
  - project1.Player
  - project1.Project1

Skip navigation links

- Package
- Class
- Use
- Tree

- Index
- Help

Uses of Package project1

- Classes in project1 used by project1
  Class
  Description
  Height
  Description: It contains the Player class, which is immutable with a constructor for a Height object in feet and inches.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help
- Summary:
- Nested |
- Field |
- Constr |
- Method
- Detail:
- Field |
- Constr |
- Method

Package project1

Class Player

java.lang.Object

project1.Player

public class Player extends Object

Description: It contains the Player class, which is responsible for the players object information such as name, age, and height in feet and inches. It is immutable with a constructor for a Player object, getter for the instance variables, toString of a player's information and a inside a toString method of the Height class.

Date: 5/23/2024

Project: Project 1

Version:
JRE17

Author:
Victoria Lee

- ## Field Summary

  Fields

Modifier and Type	Field	Description
`private final int`	`age`	Integer age for Player
`private final Height`	`height`	Height height for Player
`private final String`	`name`	String name for Player

- ## Constructor Summary

  Constructors

Constructor	Description
`Player(String name, int age, Height height)`	Constructor

- ## Method Summary

  All Methods

Instance Methods
Concrete Methods
Modifier and Type
Method
Description
`int`
`getAge()`
Retrieves age of player
`Height`
`getHeight()`
Retrieves height of player
`String`
`getName()`
Retrieves name of player
`String`
`toString()`
Override with toString(); Outputs: Name: {} Age: {} Height: {' "}; height from Height class

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

- ## Field Details
    - ### name
      private final String name
      String name for Player
    - ### age
      private final int age
      Integer age for Player
    - ### height
      private final Height height
      Height height for Player
- ## Constructor Details
    - ### Player
      public Player(String name, int age, Height height)
      Constructor

Parameters:

`name` - creates player object with name

age - creates player object with age

`height` - creates player object with height (feet, inches)

- ## Method Details

  - ### getName

    public String getName()

    Retrieves name of player

    Returns:

    name getter for value of name

  - ### getAge

    public int getAge()

    Retrieves age of player

    Returns:

    age getter for value of age

  - ### getHeight

    public Height getHeight()

    Retrieves height of player

    Returns:

    height getter for value of height from Height object/class

  - ### toString

    public String toString()

    Override with toString(); Outputs: Name: {} Age: {} Height: {' "}; height from Height class

    Overrides:

    `toString` in class `Object`

    Returns:

    string+value it outputs player information and converted height

- Summary:
- Nested |
- Field |
- Constr |
- Method
- Detail:
- Field |
- Constr |
- Method

Package project1

Class Project1

java.lang.Object

project1.Project1

---

public class Project1 extends Object

Description: It contains the Project1 class, which is responsible for the input and output of player's information. It creates a Player object for each player and add the player to an ArrayList. It calculates the average age of all players. It finds the tallest player whose age is less than or equal to the average age of all players.

Date: 5/23/2024

Project: Project 1

Version:
JRE17

Author:
Victoria Lee

- ### Constructor Summary

  Constructors
  Constructor
  Description
  `Project1()`

- ## Method Summary

  All Methods
  Static Methods
  Concrete Methods
  Modifier and Type
  Method
  Description
  `static void`
  `main(String[] args)`
  Project1 uses Height, Player classes; User inputs, create Player in ArrayList, compute averages and find tallest; has try/catch, else for invalid inputs

  ### Methods inherited from class java.lang.Object

  ```
  clone, equals, finalize, getClass, hashCode, notify, notifyAll,
  toString, wait, wait, wait
  ```

- ## Constructor Details

  - ### Project1

    public Project1()

- ## Method Details

  - ### main

    public static void main(String[] args)
    Project1 uses Height, Player classes; User inputs, create Player in ArrayList, compute averages and find tallest; has try/catch, else for invalid inputs

    Parameters:

    `args` - the command line arguments for main method

    Throws:

    `NumberFormatException` - if the user input is incorrect with not 4 items on one line