The City College of New York, CUNY
Department of Compute Science
North Academic Center, Room 8/206
160 Convent Avenue, New York, NY 10031

Assignment 1 – Fall 2020

Due Date: *by Wednesday September 16, 2020 11:59PM*
How to submit: *upload JAVA files to Blackboard*

How many steps in a mile?

A steps-to-miles calculator determines the distance a person should walk in order to attain a specific number of steps. The formula is specific to every person and should be based on the length of their stride. A person's stride is calculated based on their gender, height, and whether they are walking or running.

For this assignment we will implement a very simple walking steps to miles calculator using the formulas provided next.

Disclaimer: ***this assignment is only meant as a programming assignment and is not meant for any other purposes.***

Note:

- ✓ ***this is an individual assignment; please do your own work, sharing and/or copying code and/or solution ideas with/from others will result in a grade of 0 and disciplinary actions for all involved parties. If you run into problems and have done your best to solve them, please contact me before/after class or by e-mail.***
- ✓ ***A 20% grade deduction for every day the assignment is late.***

Assignment's Requirements:

1. Program should compile and run in order to be graded
2. You must use $String\ formatters\ (i.e. printf(\ldots))$ for ALL output statements. Your solution should <u>NOT</u> contain the methods $print(\quad)$ OR $println(\quad)$.
3. You should not need any 3rd party libraries (i.e. libraries besides Java's API). If you think some libraries maybe useful, please check with me first.
4. Comment your code:
   a. Javadoc comments for the class
   b. Javadoc comments for each of the class' members (variables and methods)
   c. Comments for major steps in your code
5. Submit two separate classes each in its own JAVA file. Please use Blackboard's upload feature and don't submit your compiled code or LINKS to online editors.
   a. $StepsToMiles$ – a container and operations class for the calculator. Follow the names shown in the UML diagram in Figure 1 exactly. I use a test script to examine your code which fails if you do not follow the naming convention.
   b. $TestStepsToMiles$ – This class contains the main method to perform unit testing on the previous class.

Class $StepsToMiles$:

a. Four variables: $name$ (String), $feet$ (double), and $inches$ (double).
b. Default $constructor$ to initialize the variables to their default values – null for strings and -1 for numeric.
c. $Non-default$ constructor to initialize the variables using the constructor's parameters.
d. $Accessors$ (setters) and $mutators$ (getters) methods for all four variables – 6 methods in total.
e. $height\_inches(\quad)$ returns the height in inches. *Conversion:* $1ft = 12\ inches$
f. $strideLength\_inches(\quad)$ returns the length of a person's stride. Remember to invoke the previous method:
   ⇨ $stride = \ll height\_inches \gg \times 0.413$

g. $miles(steps)$ returns the number of miles to walk given the desired number of steps. Remember to invoke the previous methods:

⇨ $miles = \frac{\ll strideLength\_inches \gg \times steps}{\ll inches\ per\ mile \gg} = \frac{\ll strideLength\_inches \gg \times steps}{12 \times 5280}$

h. $currDate$ returns today's date as a String. One way is to use Java's $GregorianCalendar$ to first extract the $Month$, Day, $Year$ values, then build a string representing today's date. https://docs.oracle.com/javase/8/docs/api/java/util/GregorianCalendar.html)

i. $formatAsString$: receives one input for the numbers of steps and formats and returns the class' members as a multi-line string. Use the String format method to format and return the values as shown in Figure 2.

- ✓ All leading labels should have column widths of 13 characters
- ✓ All floating-point numbers have precision 2 and the thousands comma (i.e. decimal separator)

Class $TestStepsToMiles$:

- ✓ Remember to show a message string before each prompt and use $printf(\quad)$ ONLY

a. Create an instance of class $StepsToMiles$ using the $default$ constructor. **Use the mutator methods to assign values to the class' private members.** See Figure 2 for a sample test.

b. Prompt the user to enter a $name$ and $height$ in $feet$ and $inches$. See Figure 2 for a sample test. **These values will be used in the next step.**

c. Create a second instance of $StepsToMiles$ using the non-default constructor. **Use the values entered in the previous step.**

d. Using $printf(\quad)$:

1. Using the 1st instance, print today's as shown in Figure 2
2. Using the 1st instance, print the results of calling the function $formatAsString$, pass the value 12345 for the steps parameter.
3. Using the 2nd instance, print the results of calling the function $formatAsString$, pass the value 1000 for the steps parameter.

Grading:

Item	Points
Comments (Javadoc and major steps)	10
$PersonWeight$ class (Compiles and runs)	
3 variables	3
Accessor and mutator methods	6
2 Constructors	10
$heigth\_Inches(\quad)$	5
$strideLength\_Inches(\quad)$	5
$miles(\quad)$	5
$TestPersonWeight$ class (Compiles and runs)	
Prompts	10
1st instance using default constructor	5
2nd instance using default constructor	10
Today's date	5
$printf$	16
Correct output	10
	100

Figures:

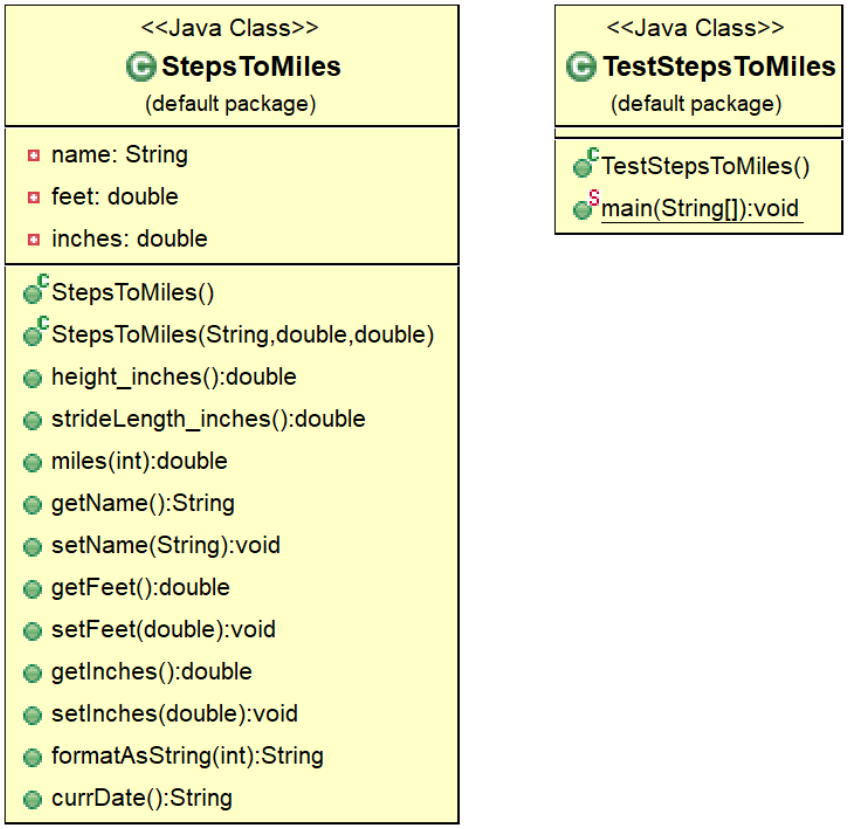

Figure 1: Class Diagram *– UML legend shown below*

```
                Enter Name: John Doe
Enter Height (ft and in): 6 1.1

Totay's Date: 9/6/2020

        Name: Jane Doe
      Height: 5.00' 4.50"
      Stride: 26.64
       Steps: 12,345
        Walk: 5.19 miles

        Name: John Doe
      Height: 6.00' 1.10"
      Stride: 30.19
       Steps: 1,000
        Walk: 0.48 miles
```

← *Prompts for the 2nd instance. Prompts are right aligned*

← *Today's date. Label is right aligned*

← *Formatted output of the 1st instance which used the default constructor and hard-coded values. Labels are right aligned*

← *Formatted output of the 2nd instance which used the non-default constructor and the prompted values above. Labels are right aligned*

Figure 2: Sample Run

The City College of New York, CUNY
Department of Compute Science
North Academic Center, Room 8/206
160 Convent Avenue, New York, NY 10031

UML Diagram Legend

Symbol	Description
	A private member (i.e. variable or method)
	A private final member (i.e. variable)
	A public field (i.e. variable or method)
	A public abstract member (i.e. variable or method)
	A public constructor
	A static public member
	An interface
	A public class
	A public abstract class
	A hollowed arrow indicates inheritance
	An open-ended arrow indicates composition
	A dotted line and hollowed arrow indicate class implementation