Regis University CC&IS
CS210 Introduction to Programming
Java Programming Assignment 5: Objects and Decisions

This assignment will require you to implement decision making in Java, starting with the code from your last Java program (Java Assn 4), which implemented objects. The Java Assn 4 program ran all of the code sequentially.

Decision statements will now be used to run some of the statements conditionally, by implementing at least one of each of the following conditional programming constructs: **if** statement or **if/else** statement, ***multiple alternative*** **if** statement, ***nested*** **if** statement, and **switch** statement.

> WARNING: Again, this assignment will be ***more challenging*** than the previous assignments, so make sure to ***start early*** and ***allocate enough time*** to complete it, including time you might need to seek any necessary help.

Problem Summary

Suppose you wanted to further modify your **BAC Calculator**, to make it more versatile. Starting with Java programming assignment 4, you could add the following features:

- Let the user specify if they are male or female.
  - Create an object
  - Calculate and display results using this information.
- Let the user adjust the metabolic rate used, based on what type of metabolism they have.
- Adjust the output, depending on the BAC level calculated.

Overview of Program

This program will still contain two classes, in two separate files within your project:

- A modified **Drinker** class to define the data fields and methods for a Drinker object, containing:
  - ***Ten*** data field definitions for a Drinker object
  - ***Two constructor*** methods to create a Drinker object (one constructor without parameters and one constructor with parameters).
  - ***Six setter*** methods to set the values for six of the data fields
  - An instance method to calculate the male volume distribution
  - An instance method to compute the female volume distribution
  - An instance method to compute the current BAC
  - An instance method to display the drinker's info and the current BAC.
  - An instance method to analyze the current BAC.
- The main **BACcalculator** class, modified to define only ***two*** methods, containing:
  - A static method to display a description of what the program will do.
  - A **main** method to display the program description, to read the inputs from the user, to create a Drinker object, and to call the other methods to calculate the current BAC, then display and analyze it.

NOTE: For an example of a Java program containing decisions, see Online Content section 11.16.

Program Requirements

Modify the program you wrote for Java Assignment 4, as follows:

1. ***Within*** the **Drinker** class:
    - Add the following additional **private** data field:
        - ***int*** weight in pounds
        - ***int*** height in inches
        - ***char*** gender (will store 'M' or 'F')
        - ***double*** metabolic rate
        - ***double*** current BAC
    - Add initialization statements to the original constructor, for each of these new fields. Use zero values for the numeric data fields and a single space character (`' '`) for the gender.
    - Modify the weight setter so that it will set the ***both*** the weight data fields at once. This method will still have one parameter: an integer weight in pounds. It will:
        - Set the new weight in pounds data field, using the parameter value
        - Use the original code to convert and set the weight in kilograms data field
    - Modify the height setter so that it will set the ***both*** the height data fields at once. This method will still have one parameter: an integer height in inches. It will:
        - Set the new height in inches data field, using the parameter value
        - Use the original code to convert and set the height in meters data field
    - Add a setter for the metabolic rate data field. This method will:
        - Have one parameter: a metabolic type (char with value 'V', 'S', 'A', or 'F')
        - Have two constants, representing the base metabolic rates for males and females (move these constant values from the assn 4 BACcalculator class to this method)
        - Use decisions to set the initial metabolic rate
            - Begin with a base metabolic rate, based on the drinker's gender
        - Then use a **switch** statement to adjust the metabolic rate, if necessary, using the metabolic type parameter, as follows:

          metabolic type is V (very slow) – reduce the metabolic rate by 0.002
          metabolic type is S (slow) – reduce the metabolic rate by 0.001
          metabolic type is A (average) – do not change the metabolic rate
          metabolic type is F (fast) – increase the metabolic rate by 0.001

          If the parameter contains any other letter, do not adjust the metabolic rate. Just display an error message telling the user that since his/her choice was invalid, no adjustment will be made.

- Modify the setter for the drinker identifier data field.
    - Setter will now only have ***one*** parameter -- the last name (will no longer use the zip code).
    - The drinker identifier field will now be created using the first 7 letters (uppercased) of the last name.
        - If the last name has less than 7 letters, the code should append Xs to the end of the name, so that there will be 7 letters total.
    - The code in the body of the setter method will:
        - Declare an integer constant to hold an id length of 7.
        - Declare a second String constant to hold seven Xs (“XXXXXXX”).
        - Use decisions with these constants and the last name to create the drinker identifier, WITHOUT using brute force.
    - NOTE: You will need to check the length of the last name BEFORE trying to extract any letters, and then use a **decision** statement to decide which letters to extract. Otherwise the program may crash when there are less than 7 letters in the last name.
- Add a ***second*** **constructor** to instantiate a **Drinker** type object.

  The new constructor with have ***seven*** parameters:

    - the drinker’s last name (String)
    - the gender (char)
    - the weight in pounds (int)
    - the height in inches (int)
    - the number of drinks consumed (int)
    - the elapsed time in hours (int)
    - metabolism type (char)

  The **constructor** will:

    - Assign initial values to the gender, drinks consumed, and elapsed time data fields, using the constructor parameters for the values.
    - Initialize the current BAC value to 0.
    - Call the modified setters for the drinker id, weights, and heights, using the constructor parameter as the setter arguments.
    - Call the new setter for the metabolic rate, using the constructor metabolism type parameter as the setter argument.

  NOTE: The original **getters** for this class will no longer be necessary. All data fields will be accessed via other instance methods.

- Modify the method that computes and displays the current BAC, so that it will ONLY compute the BAC and store the computed value into the current BAC data field (no displaying of anything).
    - Remove the parameters and use data fields instead.
    - Use a decision statement to call the correct volume distribution method for the Drinker’s gender. Store the result in a local variable.

    - Modify the formulas to use only data fields and the local variable with the volume distribution result to compute the current BAC.
- Create a new instance method to display the drinker info and current BAC, as described below.
    - Use **decision** statement to display the gender as a String
        - Display “male” when the gender data field is ‘M’
        - Otherwise, display “female”
    - Indent the lines with the drinker data.
    - Display the BAC to 3 decimal places.

      Sample for an average metabolism:

      ```
      Calculations for:
          male drinker, id JONESXX
          who is 68 inches tall and weighs 160 pounds
          after drinking 5 drinks, in 2 hours:
      Current BAC is approximately 0.103
      ```

- Create a new instance method to analyze the current BAC and display results, as described below.
    - Create three constant for BAC limits
        - A possible impairment limit (0.02)
        - A commercial driver legal limit (0.04)
        - A standard legal limit for all drivers (0.08)
    - Use a **multiple alternative if** statement to decide what to tell the user about their impairment
        - When the BAC is below the possible impairment limit, display it is okay to drive
        - When the BAC is at or above the possible impairment limit, but below the commercial driver legal limit, display that the user can legally drive but should be cautious because they could be partially impaired.
        - When the BAC is at or above the commercial driver legal limit, but below standard legal limit, display that the user cannot legally drive any commercial vehicles because they have exceeded the commercial driver legal limit and list the limit value in your output.
        - When the BAC is at or above the standard driver legal limit, display that the user cannot legally drive ANY vehicle, because they have exceeded the standard driver legal limit and list the limit value in your output.

      Sample

      ```
      You may not legally drive ANY vehicle, because you have exceeded
      the legal limit of 0.080.
      ```

    - Then use a **nested if** statement to give the user more information
        - When the BAC is at or above the standard legal limit:
            - Subtract the drinker’s metabolic rate from their current BAC to compute what their BAC level would be in one hour.
            - If the BAC level in one hour is still at or above the standard legal limit:
                - Tell the user to find a ride home

- Otherwise
    - Tell the user that the can legally drive in one hour, but should think hard about doing it.

Sample

```
Even waiting an hour will not be enough.  Find a ride home!
```

- After creating the above two instance methods, be sure to delete the old display methods from the **BACcalculator** class and the **Drinker** class.

2. Within the original class `BACcalculator` class that contains the **main** method:

- Modify the **main** method to:
    - Still display the program description, as in Java Assignment 4.
    - Modify what data is read from the user.
        - Read the drinker's last name, as in Java Assignment 4.
        - ***Remove*** the code that reads the zip code.
        - Read the drinker's height and weight, as in Java Assignment 4.
        - **Add** code to read the drinks consumed and hours elapsed.
        - Read the metabolism type (as shown in the **Sample Inputs** below):
            - Display a menu to the user, with **letters** as choices.
            - Prompt for the user's choice, and read the letter chosen by the user.

Sample Inputs

```
Enter drinker's last name: jones
Enter drinker's gender (M for male, F for female): m
Enter drinker's height (in whole inches): 68
Enter the drinker's weight (in whole pounds): 160
How many standard American drinks have been consumed: 5
How many whole hours have elapsed since drinking began? 2

Metabolism type choices
    V - Very slow metabolism
    S - Slow metabolism
    A - Average metabolism
    F - Fast metabolism
Please select the drinker's metabolism type: a
```

- Display a couple of lines to separate the input from the output.
- ***Remove*** the code that creates a blank object and calls setter for it.
- Use the new constructor to create an object of the **Drinker** class type, passing in all necessary arguments.
- Display a couple of lines to separate the input from the output.
- Use the new constructor to create an object of the **Drinker** class type, passing in all necessary arguments.
- ***Remove*** the code that computes the volume distributions (a call to the correct volume distribution should now be in the method to compute the current BAC).

- Using the **Drinker** class object:
    - Call the method to compute the current BAC.
    - Call the method to display the drinker’s data and current BAC.
    - Call the method to analyze the drinker’s current BAC.

WARNING: As with your previous Java programs, the objects, classes, and methods must be implemented exactly as specified above, or you will lose a significant number of points.

Coding Standards

The program must also follow the **CS210 Coding Standards** from Content section 6.10.

You must ***include*** the following comments:

- Comments at the ***top of the file, above*** the **main** class, describing what the class does
    - Include **tags** with the author’s name (i.e. your full name) and the version of the code (e.g. @version 1.0, Java Assn 3)
- Comments at the ***top of each method***, ***above*** the method header, describing what the method does (***only*** this method – do not refer to actions of any other methods)
    - Include **tags** with names and descriptions of ***each*** parameter and return value.

Delete any default comments supplied by the IDE that you did not use.

Debugging and Testing

Run, test, and debug your Java program, until you confirm that all of the decision control structures work. Be sure to test your program with different inputs to make sure it provides correct results.

Submission

This programming assignment is due by midnight of the date listed in the **Course Assignments by Week**.

Again, you will submit a single **zip** file containing all of the files in your project.

- First export your project from NetBeans:
    - Highlight the project name.
    - Click on **File** from the top menu, and select **Export Project**.
    - Select **To ZIP**
    - Name your export file in the following format:
      `<lastname>Assn<x>.zip`

      For example:
      `SmithAssn5.zip`

  NOTE: Save this zip file to some other directory, not your project directory.

- Then submit your **.zip** file to the **Java Prog Assn 5** assignment submission folder (located under the **Assignments/Dropbox** tab in the online course).

    - Warning: Only NetBeans export files will be accepted.
      Do not use any other kind of archive or zip utility.

Grading

Your program will be graded using the **rubric** that is linked on the same assignment page from which this program requirements file was downloaded.

WARNING:
Programs submitted more than 5 days past the due date will ***not*** *be accepted,*
and will receive a grade of 0.