Homework 05
Coffee Hour

Objective:
Create a program that has a class to represent a cup of coffee that combines the properties: name and caffeine content. The class should also have a method that calculates the number of cups that would be maximally risky to consume in a short period of time. The program should create two instances of the class coffee where the user enters their properties and should output the number of coffees that consumed would be risky.

Requirements:

- Functionality. (80pts)
  - No Syntax Errors. (80pts*)
    - *Code that cannot be compiled due to syntax errors is nonfunctional code and will receive no points for this entire section.
  - Create a class name **Coffee** with the following. (70pts)
    - Instance Variables. (2pts)
      - Name: A non-null String that represents the name of the Coffee. Its default value must be "none"
      - Caffeine content: The amount of caffeine in a single cup of coffee. Its values must be between 50 to 300mg inclusively and its default value should be 50.
      - Every scope must be private.
      - All above must apply for full credit.
    - Default Constructor. (4pts)
      - Each instance variable must be assigned a valid default value.
      - The default values of each instance variable can be found in the section "Instance Variables"
    - Parameterized Constructor. (4pts)
      - Must include a parameter for each instance variable.
      - Parameters must be checked for valid values before they are assigned. If the parameter value is not valid, then an exception must be thrown. Valid values can be found in the section "Instance Variables".
    - Accessors for each instance variable. (15pts)
      - Must follow the structure and naming conventions demonstrated in lecture.
      - All above must apply for full credit.
    - Mutators for each instance variable. (15pts)
      - Each mutator must check for valid values before assigning. If the parameter value is not valid, then an exception must be thrown.

- Default and valid values can be found in the section "Instance Variables".
- Must follow the structure and naming conventions demonstrated in lecture.
- All above must apply for full credit.

- toString method. (10pts)
  - Must return a String formatted as,
    Coffee Name: <<coffee's name>> Caffeine Amount: <<coffee's caffeine amount>>
  - Where coffee's name and coffee's caffeine amount are the instance variable values.
  - All above must apply for full credit.
- equals method. (10pts)
  - Must have a parameter for another Coffee's instance.
  - The method must return true or false based on if this coffee's instance variables match the other coffee's instance variables.
  - All above must apply for full credit.
- riskyAmount method. (10pts)
  - Must require no parameters and must return a decimal value representing the number of cups that would be dangerous to consume within an hour.
  - This value is based on the coffee's caffeine content and can be calculated by,

    Risky Cup Amount = 180.0 / ((Caffeine Content / 100.0)*6.0)

  - All above must apply for full credit.

- Create a class called **CoffeeTester** with the following. (10pts)
  - Includes the main method.
  - Creates 2 instances of Coffee.
  - The user must be asked to enter the name and caffeine content of both coffees, and those values must be assigned as long as they are valid.
  - The program must then print out the properties of each coffee (name and caffeine content) and print out the number of cups that if consumed within an hour would be a health risk.
  - The program must then determine if the two coffee's have the same properties.
  - Finally, the user must be prompted to either stop the program or restart it.
  - All must apply for full credit.

- Coding Style. (10pts)

- Code functionality organized within multiple methods other than the main method, and methods organized within multiple classes where appropriate. (5pts)
- Readable Code. (5pts)
  - Meaningful identifiers for data and methods.
  - Proper indentation that clearly identifies statements within the body of a class, a method, a branching statement, a loop statement, etc.
  - All the above must apply for full credit.

- Comments. (10pts)
  - Your name in every file. (5pts)
  - At least 5 meaningful comments in addition to your name. These must describe the function of the code it is near. (5pts)

Example Dialog:
*The following Example Dialog demonstrates the interactions between a user and ONE possible implementation of the required software's front-end / user interface. The software's front-end / user interface may be implemented in MANY different ways and will receive full credit as long as it meets the most minimal of the above requirements. While you may use the example dialog as a guide, it is strongly encouraged to create the front-end / user interface in your own way. *

Key	
Unhighlighted Text	**Program's Output**
Highlighted Text	**User's Input**

```
Let's Coffee!!!1!11!!ONE!!!1!

What's the name of the first coffee?
Double Triple Loca Mocha Latte Venti Grande
What's the caffeine content?
150

What's the name of the second coffee?
Waffle House Coffee
What's the caffeine content?
100
It would take 20.0 Double Triple Loca Mocha
Latte Venti Grande coffees before it's
dangerous to drink more.
```

```
It would take 30.0 Waffle House Coffee
coffees to before it's dangerous to drink
more.
```

Finally:
Upload the all Java source files (.JAVA extension) to the CSCE Dropbox