Lab 07
Bicycle Builder

Objective:
Practice object-oriented principles by building a bicycle out of multiple objects and test it with the tester.

Lab Solution

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.
  - Set-Up the Project (5pts)
    - Include the tester code in your project.
    - Do not alter the provided code.
  - Write a class called **Wheel** with the following: (25pts)
    - Instance Variables
      - Diameter: This represents the diameter of the wheel and must be between 16in to 55in inclusively. Its default value is 16.
      - Width: This represents the width of the wheel and must be between 1in to 2.5in inclusively. Its default value is 1.
    - Constructors
      - Default: Must set all properties to their default values mentioned in the "Instance Variables" section.
      - Parameterized: Must take in a parameter for each instance variable in the order named above. This means the first instance variable is the first parameter, the second instance variable is the second parameter, and so on. This must set the instance variable values only if the given values are valid, but otherwise it must set the instance variables to their default values.
    - Methods
      - Accessors and Mutators for the instance variables
        - Make sure in the mutators check for valid values named in the "Instance Variables" Section.
        - If the value that is being set is not valid, then set the instance variable to its default value.
      - Equals: This method takes in another instance of Wheel and only returns true if all of the instance variables match.

- ToString: This method returns a String with all of the instance variable values concatenated together with the format:
  [Wheel] Diameter <<Wheel's Diameter>> Width: <<Wheel's Width>>
  Where values in "<<>>" correspond to the instance variable values.
- All above must apply for full credit.

- Write a class called **Frame** with the following: (25pts)
  - Instance Variables
    - Size: This represents the frame's size and must be between 18.5in to 60 in inclusively. Its default value is 18.5.
    - Type: This non-null String value represents the type of frame and can only be "Diamond", "Step-Through", "Truss", or "Penny-Farthing". Its default value is "Diamond"
  - Constructors
    - Default: Must set all properties to their default values mentioned in the "Instance Variables" section.
    - Parameterized: Must take in a parameter for each instance variable in the order named above. This means the first instance variable is the first parameter, the second instance variable is the second parameter, and so on. This must set the instance variable values only if the given values are valid, but otherwise it must set the instance variables to their default values.
  - Methods
    - Accessors and Mutators for the instance variables
      - Make sure in the mutators check for valid values named in the "Instance Variables" Section.
      - If the value that is being set is not valid, then set the instance variable to its default value.
    - Equals: This method takes in another instance of Frame and only returns true if all of the instance variables match.
    - ToString: This method returns a string with all of the instance variable values concatenated together with the format:
      [Frame] Size: <<Frame's Size>> Type: <<Frame's Type>>
      Where values in "<<>>" correspond to the instance variable values.
  - All above must apply for full credit.
- Write a class called **Bicycle** with the following: (25pts)
  - Instance Variables
    - Make: This non-null String value represents the maker of the bicycles. Its default value is "none".

- FrontWheel: This is an instance of type Wheel and represents the front wheel of the bicycle. Its default value must be the default Wheel.
- BackWheel: This is another instance of type Wheel and represents the back wheel of the bicycle. Its default value must be the default Wheel.
- Frame: This is an instance of type Frame and represents bicycle's frame. Its default value is the default Frame.

- Methods
  - Accessors and Mutators for the instance variables
    - Make sure in the mutators check for valid values named in the "Instance Variables" Section.
    - If the value that is being set is not valid, then set the instance variable to its default value.
  - Equals: This method takes in another instance of Peanut Butter and only returns true if all of the instance variables match. For name case should be ignored.
  - ToString: This method returns a string with all of the instance variable values concatenated together with the format.

    [Bicycle] Make: <<Bicycle's Make>> Front Wheel <<Bicycle's Front Wheel>> Back Wheel <<Bicycle's Back Wheel>> Frame: <<Bicycle's Frame>>

    Where values in "<<>>" correspond to the instance variable values.
  - All above 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:

```
---------------------------------
Welcome to the Bicycle Builder Tester!
```

```
---------------------------------
First we will create a "Default" Bicycle
Printing the Bicycle's data
Bicycle's make: none
Bicycle's front wheel: Diameter 16.0 Width:
1.0
Bicycle's back wheel: Diameter 16.0 Width:
1.0
Bicycle's frame: Size: 18.5 Type: diamond

Testing the toString method
[Bicycle] Make: none Front Wheel: [Wheel]
Diameter: 16.0 Width: 1.0 Back Wheel:
[Wheel] Diameter: 16.0 Width: 1.0 Frame:
[Frame] Size: 18.5 Type: diamond

Setting invalid values for the default
bike's wheels and frame
[Bicycle] Make: none Front Wheel: [Wheel]
Diameter: 16.0 Width: 1.0 Back Wheel:
[Wheel] Diameter: 16.0 Width: 1.0 Frame:
[Frame] Size: 18.5 Type: diamond

Creating another bike using the
parameterized constructor
[Bicycle] Make: Big Wheel Front Wheel:
[Wheel] Diameter: 55.0 Width: 2.5 Back
Wheel: [Wheel] Diameter: 18.0 Width: 2.0
Frame: [Frame] Size: 60.0 Type: Penny-
Farthing

Creating another bike using the
parameterized constructor with invalid
values
```

```
[Bicycle] Make: none Front Wheel: [Wheel]
Diameter: 16.0 Width: 1.0 Back Wheel:
[Wheel] Diameter: 16.0 Width: 1.0 Frame:
[Frame] Size: 18.5 Type: diamond

Checking the "equals" method
Does the first and third bicycles have
different memory addresses? true
Does the first and third bicycles have the
same properties? true

---------------------------------
Tests Complete! Goodbye
---------------------------------
```

Solution Tests:

1. Is your name written as a comment in all source files?
2. Does the solution compile (no syntax errors)?
3. Does your output match the example dialog?

Lab Report

1. Create a section named "Problem" and describe this lab's problem in your own words. (10pts).
2. Create a section named "Solution Description" and describe how the code solves the problem in your own words. (10pts).
3. Create a section named "Problems Encountered" and describe the various syntax, run-time, and logic errors that were encountered while implementing the solution. (10pts).
4. Explain the purpose of accessors ("getters"). (10pts).
5. Explain the purpose of mutators ("setters"). (10pts).
6. Describe when it is most appropriate to use the "equals" method for comparing objects. (10pts).
7. Describe when it is most appropriate to use the "==" operator for comparing objects. (10pts).

For questions 8, 9, and 10 refer to the code below.

```java
public class Gear
{
    private double diameter;
    private int cogs;
    public Gear()
    {
        this.diameter = 1.0;
        this.cogs = 2;
    }
    public Gear(double aD, int aC) {
        this.diameter = aD;
        this.cogs = aC;
    }
    public double getDiameter()
    {
        return diameter;
    }
    public void setDiameter(double aD)
    {
        if(aD > 0.0)
        {
            this.diameter = aD;
        }
        else
        {
            this.diameter = 1.0;
        }
    }
    public int getCogs()
    {
        return cogs;
    }
    public void setCogs(int aC)
    {
        if(aC >= 2)
        {
            this.cogs = aC;
        }
        else
        {
            this.cogs = 2;
        }
    }
    public String toString()
    {
        return "[Gear] Diameter: "+this.diameter+" Cogs: "+this.cogs;
    }
    public boolean equals(Gear aG)
    {
        return aG != null &&
                this.diameter == aG.getDiameter() &&
                this.cogs == aG.getCogs();
    }

}
```

```
public class Machine
{
    private String name;
    private Gear bigGear;
    private Gear mediumGear;
    private Gear smallGear;

    public Machine()
    {
        this.name = "none";
        this.bigGear = new Gear(16.0,32);//Default big gear
        this.mediumGear = new Gear(8.0,16);//Default medium gear
        this.smallGear = new Gear(4.0,8);//Default small gear
    }
    public Machine(String aN, Gear bG, Gear mG, Gear sG)
    {
        if(aN== null || bG == null || mG == null || sG == null)
        {
            this.name = "none";
            this.bigGear = new Gear(16.0,32);//Default big gear
            this.mediumGear = new Gear(8.0,16);//Default medium gear
            this.smallGear = new Gear(4.0,8);//Default small gear
        }
        else
        {
            this.name = aN;
            this.bigGear = bG;
            this.mediumGear = mG;
            this.smallGear = sG;
        }
    }
    public String getName()
    {
        return this.name;
    }
    public void setName(String aN)
    {
        if(aN != null)
        {
            this.name = aN;
        }
        else
        {
            this.name = "none";
        }
    }
    public String toString()
    {
        return "[Machine] Name: "+this.name+"\n"+bigGear+"\n"+mediumGear+"\n"+smallGear+"\n";
    }
    public static Machine copyAndRenameMachine(String aN, Machine aM)
    {
        if(aM == null)
            return null;
        Machine ret = aM;
        ret.setName(aN);
        return ret;
    }
}
```

8. What will the code snippet below print to the console? (10pts)

```
public static void main(String[] args)
{
    Machine m = new Machine();
    System.out.println(m);
}
```

9. What will the code snippet below print to the console? (10pts).

```
public static void main(String[] args)
{
    Gear g1, g2, g3;
    g1 = g2 = g3 = new Gear();
    g1.setDiameter(32);
    g1.setCogs(64);
    g2.setDiameter(32);
    g2.setCogs(64);
    g3.setDiameter(32);
    g3.setCogs(64);
    Machine m = new Machine("Machine02",g1,g2,g3);
    System.out.println(m);
}
```

10. What will the code snippet below print to the console? (10pts).

```
public static void main(String[] args)
{
    Machine m1 = new Machine();
    m1.setName("Machine01");
    Machine m2 = Machine.copyAndRenameMachine("Machine02", m1);
    System.out.println(m1);
    System.out.println(m2);
}
```

Finally:
Upload the source code (.JAVA File Extension) and written lab report (.DOC, .DOCX, or .PDF file extension) to the CSCE Dropbox.