Composition

Directions:

This is an exercise in **Class Composition**. To complete the exercise, you must create five classes with associated member variables and methods.

The five classes should be created as follows:

1) Create a class and name it **Lamp**. Inside this class should be declared three member variables: **style** of type **String**, **battery** of type **boolean**, and **globRating** of type **int**. All variables should be marked **private**. A constructor needs to be created which accepts the three member variables as parameters. In addition, four methods should also be created: **turnOn()** has no return type and should print a message that the lamp is being turned on; **getStyle()** returns the lamp style; **isBattery** returns the value of **battery**; and **getGlobRating()** returns the **globRating** of the lamp.

2) Create a class and name it **Bed**. Five **private** member variables should be declared: **style** of type **String**, and **pillows, height, sheets, quilt** of type **int**. A constructor should be coded which accepts these five member variables as parameters. Also, six methods should be defined: **make()** has no return type and prints a message to the effect that the bed is being made; **getStyle()** which returns the value of **style**; **getPillows()** returns the number of **pillows**; **getHeight()** returns the **height** of the bed; **getSheets()** returns the number of **sheets** on the bed; and **getQuilt()** returns the value of **quilt**.

3) Create a class with the name **Ceiling**. There are two member variables for this one, **height** and **paintedColor**, both of type **int**. There should also be a constructor which accepts both member variables as parameters. There are also two additional methods which should be defined: **getHeight()** shall return the value of **height** and **getPaintedColor()** should return the value of **paintedColor**.

4) Create a class with the name **Wall**. It contains one member variable, **direction**, and is of type **String**. A constructor for **Wall** should accept one parameter for the member variable **direction**. A getter should also be defined for the **direction** field called **getDirection()**.

5) Create a class with the name **Bedroom**. This class contains eight member variables:

name of type **String**; **wall1, wall2, wall3, wall4** of type **Wall**; **ceiling** of type **Ceiling**; **bed** of type **Bed**, and **lamp** of type **Lamp**. The class constructor should accept all eight of the member variables as parameters, and there should also be two additional methods:

getLamp() which returns an object of type **Lamp**, and **makeBed()** which prints a message that the bed is being made and also calls the **make()** method in the **Bed** class.

Input/Output:

Once you have completed coding your classes you should then use the following code in your main class to test your code and for correct output. This way you can be sure that your code works before pasting your five classes into the code evaluator.

1. Wall wall1 = new Wall("West");
2. Wall wall2 = new Wall("East");
3. Wall wall3 = new Wall("South");

4. Wall wall4 = new Wall("North");
5. Ceiling ceiling = new Ceiling(12, 55);
6. Bed bed = new Bed("Modern", 4, 3, 2, 1);
7. Lamp lamp = new Lamp("Classic", false, 75);
8. Bedroom bedRoom = new Bedroom("YOUR NAME HERE", wall1, wall2, wall3, wall4, ceiling,bed, lamp);
9. bedRoom.makeBed();
10. bedRoom.getLamp().turnOn();

Tips:

Remember that after testing you will not put your main method into the code evaluator. You will only paste in your five classes you have created in the exercise.

To be sure that the correct output is generated so your code passes the evaluation, use the following statements in your code:

1) System.out.print("Bedroom -> Making bed | "); should be used in the **makeBed()** method of the **Bedroom** class;

2) System.out.print("Bed -> Making | "); should be used in the **make()** method of the **Bed** class; and

3) System.out.print("Lamp -> Turning on"); should be used in the **turnOn()** method of the **Lamp** class.

Encapsulation

Directions:

In this exercise you will create one class and name it **Printer**. This class will have two member variables of type **int**, **tonerLevel** and **pagesPrinted**, and one of type **boolean** called **duplex**. All three fields will have **private** access. The constructor will accept two of these variables, **tonerLevel** and **duplex**, as parameters following these rules:

- **tonerLevel** must be greater than -1 but less than or equal to 100. If it does not meet these conditions then it should be initialized to -1.
- **duplex** should be initialized to the value of the parameter.

In addition, **pagesPrinted** should be initialized to 0.

Three other methods need to be defined in this way:

1. **addToner** will accept one parameter, **tonerAmount** of type **int**. First off, **tonerAmount** should be greater than 0 and less than or equal to 100. If this condition is met a second test must be included to test whether **tonerLevel** plus **tonerAmount** is greater than 100. If so, the method should return -1. If not then **tonerLevel** should have the incoming **tonerAmount** added to it and the new **tonerLevel** should then be returned by the method. Also, if the initial condition test fails, then the method should return -1.
2. **printPages** will accept one parameter, **pages** of type **int**. A variable called **pagesToPrint** should be created and initialized to the value of the incoming parameter. A conditional check should be performed on whether the **Printer** class field, **duplex**, is either "true" or "false". If "true" then a calculation must be performed to determine the number of pages needed to print the job double sided. The **pagesToPrint** value is then added to the class field **pagesPrinted**, but the value of **pagesToPrint** should be returned by the method.
3. **getPagesPrinted** has no parameters and merely returns the value of the member variable **pagesPrinted**.

Example input:

1. Printer printer = new Printer(50, true);
2. System.out.println(printer.addToner(50));
3. System.out.println("initial page count = " +printer.getPagesPrinted());
4. int pagesPrinted = printer.printPages(4);
5. System.out.println("Pages printed was " + pagesPrinted +" new total print count for printer = " +printer.getPagesPrinted());
6. pagesPrinted = printer.printPages(2);
7. System.out.println("Pages printed was " + pagesPrinted +" new total print count for printer = " +printer.getPagesPrinted());

Example output:

1. 100

2. initial page count = 0
3. Printing in duplex mode
4. Pages printed was 2 new total print count for printer = 2
5. Printing in duplex mode
6. Pages printed was 1 new total print count for printer = 3

Tips:

1. Remember to only paste the code from **Printer** class into the exercise evaluator.
   The **Main** class does not need to be pasted in.
2. You can include a message in the **printPages** method which informs the user that the printer is printing in duplex mode if **duplex** is equal to "true" if you want.
3. When calculating **pagesToPrint** if **duplex** is equal to "true" remember that there are two operators which can help you with this. The division "**/**" operator divides a number and returns only the quotient without any remainder. And the modulo "**%**" operator divides the number and returns only the remainder, whether 0 or otherwise.

Polymorphism

Description:

For this exercise you will create four classes of vehicles. The first class should be named **Car**. This will be the base class for three other classes, **Mitsubishi**, **Holden**, and **Ford**.

The first class contains four member variables, or fields, with these names and conditions:

- **engine** of type **boolean**
- **cylinders** of type **int**
- **name** of type **String**
- **wheels** of type **int**

All four member variables should have **private** access.

The constructor should accept two parameters, **cylinders** and **name**. Also, all of these cars have an engine and four wheels. So the other two fields should be set to default values.

In addition, there are five other methods These methods should be defined as described below:

1. **startEngine** accepts no parameters and returns a message which says that the car's engine is starting.
2. **accelerate** accepts no parameters and returns a message that says the car is accelerating.
3. **brake** accepts no parameters and returns a message stating the car is braking.

The messages these methods return should somewhere contain the name of the class, **Car**.

Two getter methods should also be defined here for the member variables **cylinders** and **name**. All methods have **public** access.

The other three classes mentioned above are sub-classes of **Car**. These classes have no member variables and the constructor for each will call the parent constructor for object instantiation. Each of these classes will override the three parent methods **startEngine**, **accelerate**, and **brake**. The return messages for these methods should somewhere contain the name of the class to which the methods belong.

Example input:

1. Car car = new Car(8, "Base car");
2. System.out.println(car.startEngine());
3. System.out.println(car.accelerate());
4. System.out.println(car.brake());
5. Mitsubishi mitsubishi = new Mitsubishi(6, "Outlander VRX 4WD");
6. System.out.println(mitsubishi.startEngine());
7. System.out.println(mitsubishi.accelerate());
8. System.out.println(mitsubishi.brake());

```
Ford ford = new Ford(6, "Ford Falcon");
System.out.println(ford.startEngine());
System.out.println(ford.accelerate());
System.out.println(ford.brake());
Holden holden = new Holden(6, "Holden Commodore");
System.out.println(holden.startEngine());
System.out.println(holden.accelerate());
System.out.println(holden.brake());
```

Example output:

```
Car -> startEngine()
Car -> accelerate()
Car -> brake()
Mitsubishi -> startEngine()
Mitsubishi -> accelerate()
Mitsubishi -> brake()
Ford -> startEngine()
Ford -> accelerate()
Ford -> brake()
Holden -> startEngine()
Holden -> accelerate()
Holden -> brake()
```

Notes:

- Remember that your Main class is not included in the code evaluation.
- You should paste the code from each of your classes into the respective class file in the code evaluator.