**Abstract Classes:**

1. Create an abstract class, `RentableDwelling`, containing:

   - a (private) double instance variable, `rent`
   - a public getter and setter methods for `rent`
   - a protected constructor accepting a double used to initialize `rent`
   - a public abstract void method `increaseRent` that accepts a double as a parameter.

write a (non-abstract) subclass, `Apartment`, containing:

   - A private static int `numberOfApts` set initially to 0
   - A constructor accepting a double which it passes to the superclass’s constructor. It also increments `numberOfApts` by 1.
   - A static method `getNumberOfApts` which returns the number of apartments
   - A void method `increaseRent` which accepts a percentage as a parameter and increases the rent by that percentage. Example, if the percentage is .10 and the rent is 1000, the new rent would be 1100.

Write a `TestRent` class which will do the following in main():

   - Create 3 new apartments, called `apt1`, `apt2`, and `apt3`. The first has a rent of 1000, the second 1500 and the third 1800.25.
   - Print the number of apartments using the `getNumberOfApts` method.
   - Modify `apt2`’s rent by 10 percent. Print out the new rent for `apt2`.

Output:

```
Number of Apartments: 3
Apartment2's new rent is: 1650.0
```

2. Update `SimpleGeometricObject` (used last week) so that it’s an **abstract** class.
   a. In addition to the variables and methods already there, modify the constructors so they are protected, add two public abstract methods `getArea` and `getPerimeter`. Both return doubles and take no arguments.

   ```java
   abstract double getArea();
   abstract double getPerimeter();
   ```

   b. Create a class called `Square` which inherits from `SimpleGeometricObject`. It has a private instance variable, `side`, which is the length of one of the sides of the square. Have a constructor which sets the parameter to the instance variable. Add getter and setter methods. Implement the `getArea` and `getPerimeter` methods. The area of a square is the length of one of the sides squared and the perimeter is 4 X the length of a side.
c. Create a test class, TestSquare, which will create a Square object with a side of length 5. Call a static method, printInfo(). Create the method printInfo, which accept a SimpleGeometricObject as a parameter and will print out the perimeter and area by calling the appropriate methods.

Output:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>25.0</td>
</tr>
<tr>
<td>Perimeter</td>
<td>20.0</td>
</tr>
</tbody>
</table>

Interfaces:

3. Create an interface, Account, with the following methods:

d. deposit() - accepts an integer parameter and returns a double.
e. withdraw() - accepts an integer parameter and returns a double.

f. Define a class, BankAccount, that implements the above interface and has the following:

i. A private instance variable named balance.

ii. A constructor that accepts an integer that is used to initialize the instance variable.

iii. An implementation of the deposit() method that adds its parameter to balance. The new balance is returned.

iv. An implementation of the withdraw method that checks whether its parameter is less than or equal to the balance and if so, decreases the balance by the value of the parameter. This method returns the available balance.

g. Define a class, TestBankAccount, which does the following:

i. Have the user create an instance of BankAccount with their balance, of $200.50.

ii. Then deposit another $2,000.00 into the bank account.

iii. Then withdraw $400.00. Print what’s left of your balance.

My output:

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>I have</td>
</tr>
<tr>
<td>1800.5</td>
</tr>
<tr>
<td>remaining</td>
</tr>
</tbody>
</table>

4. Design an interface Colorable. Colorable should contain an abstract void method named howToColor().

Modify the Square class we just created in #2 above to have the Square class implement colorable (in addition to extending SimpleGeometricObject). Now the Square class must
contain the method howToColor(). howToColor should print the message “Color all four sides.” Modify TestSquare that after main calls printInfo() it should call howToColor().

My output:

```
Area 25.0
Perimeter 20.0
Color all four sides.
```