Java Programming

2-2: Java Class Design - Interfaces

Practice Activities

Lesson Objectives:

- Model business problems using Java classes
- Make classes immutable
- User Interfaces

Vocabulary:

Identify the vocabulary word for each definition below.

	A specialized method that creates an instance of a class.
	A keyword that qualifies a variable as a constant and prevents a method from being overridden in a subclass.
	A class that it can't be overridden by a subclass, in fact it can't be subclassed.
	Defines constants and methods without implementation

Try It/Solve It:

1. Update the JavaBank.java application to implement a new light blue company color that is to be used across all of the graphical user interfaces in the Java application. The values to be used are Red: 173, Green 216 and Blue 230.

This exercise uses the bike project from Java Programming 2-1: Working with Pre-Written Code Practice Activity. If you have not completed that section, please go and do so before continuing with this exercise.

2. Create the following interface in the bike project that sets the name of the bike company as an unchangeable value. It also defines the methods that must be implemented by any class that uses the interface.

```
package bikeproject;
public interface BikeParts {
   //constant declaration
   public final String MAKE = "Oracle Bikes";

   //required methods after implementation
   public String getHandleBars();
   public void setHandleBars(String newValue);
   public String getTyres();
   public void setTyres(String newValue);
   public String getSeatType();
   public void setSeatType(String newValue);

}//end interface BikeParts
```

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

3. Create an interface named MountainParts that has a constant named TERRAIN that will store the String value “off_road”. The interface will define two methods that accept a String argument name newValue and two that will return the current value of an instance field. The methods are to be named: getSuspension, setSuspension, getType , setType.

4. Create a RoadParts interface that has a constant named terrain that will store the String value “track_racing”. The interface will define two methods that accept a String argument name newValue and two that will return the current value of an instance field. The methods are to be named: getTyreWidth, setTyreWidth, getPostHeight, setPostHeight.

5. Use the BikeParts interface with the Bike class adding any unimplemented methods required. Add the required internal code for each of the added methods.

6. Use the MountainParts interface with the MountainBike class adding any unimplemented methods required. Add the required internal code for each of the added methods.

7. Use the RoadParts interface with the RoadBike class adding any unimplemented methods required. Add the required internal code for each of the added methods.

8. Run and test your program, it should do exactly as it did before.

9. At the bottom of the driver class update the height of the post for bike1 to 20 instead of 22.

10. Display the values of bike1 to screen to confirm the change.

11. Run and test your program.

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.