Java Programming

2-1: 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. Create an interface named Chassis. Add the following to the interface:
    - A public constant string named chassis with a value of “Chassis”.
    - The definition of a public getChassisType method that returns an instance of Chassis.
    - The definition of a public setChassisType that accepts a string named vehicleChassis and returns a void.

2. Create an interface Engine with the following list of public method definitions that return a void:

```
setEngineCylinders(int engineCylinders);
setEngineManufacturedDate(Date date);
setEngineManufacturer(String manufacturer);
setEngineMake(String engineMake);
setEngineModel(String engineModel);
setDriveTrain(String driveTrain);
setEngineType(String fuel);
```

3. Create a concrete class named VehicleChassis that implements the interface Chassis with the following:
    - Create a String named chassisName instance variable.
    - A public default constructor and an overloaded constructor with the following value:

        A String with a parameter value of chassisName
    - Set the chassisName instance variable in both, use the interface constant as the default String value.
    - A public method named getChassisType that doesnt have a formal parameter and that returns an instance of the interface Chassis (hint that would be a copy of this class).
    - A public method setChassisType that takes a String parameter vehicleChassis and that returns a void. It should set the instance variable chassisName.
    - A public toString method that returns the following:

      ```
      Chassis Name        : Chassis
      ```
    - Write a static main method that creates two objects, one with the default constructor and the other with the constructor with parameters. Give sample data for the parameters.

4. Create a concrete class named VehicleFrame that implements the interface Chassis with the following:
    - Create a String named vehicleFrameType instance variable.
    - A public default constructor and an overloaded constructor with the following value:

        A String with a parameter value of vehicleFrameType
    - Set the vehicleFrameType instance variable in both, use “Unibody” as the default String value.
    - A public method named getChassisType that didn't have a formal parameter and that returns an instance of the interface Chassis (hint that would be a copy of this class).
    - A public method setChassisType that takes a String parameter vehicleFrameType and that returns a void. It should set the instance variable vehicleFrameType.
    - A public toString method that returns the following:

      ```
      Chassis             : Chassis
      Vehicle Frame       : Unibody
      ```
    - Write a static main method that tests two scenarios:
        1. One that prints all fuel grade values, like:

           ```
           Chassis             : Chassis
           Vehicle Frame       : Unibody
           ```
        2. One that prints a value set by a single string value.

           ```
           Vehicle Frame       : Ladder Frame
           ```

5. Create a concrete class named ManufacturedEngine that implements the interface Engine with the following:

    - Create the following private instance variables:

      ```
      String     engineManufacturer;
      Date       engineManufacturedDate;
      String     engineMake;
      String     engineModel;
      int        engineCylinders;
      String     engineType;
      String     driveTrain;
      ```

    - A public default constructor with no formal parameters and initialize all instance variables with generic literal values.
    - A public overloaded constructor with values for all of the variables defined above.
    - A public method implementations for all of the public methods found in the Engine interface.
    - A public toString method that returns the following:

      ```
      Engine Manufacturer : Generic
      Engine Manufactured : Thu Feb 02 00:55:44 MST 2012
      Engine Make         : Generic
      Engine Model        : Generic
      Engine Type         : 85 AKI
      Engine Cylinders    : 0
      Drive Train         : 2WD: Two-Wheel Drives
      ```

    - Write a static main method that tests two scenarios:
    - One that prints a generic set of strings, like:

      ```
      Engine Manufacturer : Generic
      Engine Manufactured : Thu Feb 02 00:55:44 MST 2012
      Engine Make         : Generic
      Engine Model        : Generic
      Engine Type         : 85 AKI
      Engine Cylinders    : 0
      Drive Train         : 2WD: Two-Wheel Drive
      ```

    - One that accepts call parameters and returns the following:

      ```
      Engine Manufacturer : Honda
      Engine Manufactured : Tue Jan 03 07:13:19 MST 2012
      Engine Make         : H-Series
      Engine Model        : H23A1
      Engine Type         : 88 AKI
      Engine Cylinders    : 4
      ```

```
Drive Train         : 2WD: Two-Wheel Drive
```

6. Create an interface Feature with the following method definitions:

```
public String getFeature();
public void setFeature(String feature);
```

7. Create a concrete class named InteriorFeature that implements the interface Feature with the following:
   - Create a String named interiorFeature as an instance variable.
   - A public default constructor without parameters that sets the interiorFeature instance variable to “Generic”.
   - An overloaded constructor with the following value:

     A String with a parameter value of interiorFeature
   - Set the interiorFeature instance variable to the parameter interiorFeature.
   - A public method named getFeature that doesnt have a formal parameter and that returns an instance of String.
   - A public method setFeature that takes a String parameter interiorFeature and that returns a void. It should set the instance variable interiorFeature.
   - A public toString method that returns the following:

     `Interior [Generic]`
   - Write a static main method that tests two scenarios:
   - One that prints all fuel grade values, like:

     `Interior [Generic]`

     Another like

     `Interior [Climate Control]`

8. Create a concrete class named ExteriorFeature that implements the interface Feature with the following:
   - Create a String named exteriorFeature as an instance variable.
   - A public default constructor without parameters that sets the exteriorFeature instance variable.
   - An overloaded constructor with the following value:

     A String with a parameter value of exteriorFeature
   - Set the exteriorFeature instance variable in both, use features as String values.
   - A public method named getFeature that doesnt have a formal parameter and that returns an instance of String.
   - A public method setFeature that takes a String parameter exteriorFeature and that returns a void. It should set the instance variable exteriorFeature.
   - A public toString method that returns the following:

     `Exterior [Generic]`
   - Write a static main method that tests two scenarios:

- One that prints all fuel grade values, like:

  ```
  Exterior [Generic]
  ```

  Another like

  ```
  Exterior [Fog Lamps]
  ```

9. Create a concrete class named Vehicle that implements the Engine and Chassis interfaces with the following:
   - Create the following private instance variables:

     ```
     Date       vehicleManufacturedDate;
     String     vehicleManufacturer;
     String     vehicleMake;
     String     vehicleModel;
     Chassis    vehicleFrame;
     String     vehicleType;
     String     driveTrain;
     Engine     vehicleEngine;
     ```

   - A public default constructor with no formal parameters and initialize all instance variables with generic literal values.
   - A public overloaded constructor with values for all of the variables defined above.
   - Public method implementations for all of the public methods found in the Engine interface.
   - Public method implementations that set all instance variables
   - Public method implementations for the Chassis interface
   - A public toString method that returns the following:

     ```
     Manufacturer Name   : Generic
     Manufactured Date   : Thu Feb 02 01:38:31 MST 2015
     Vehicle Make        : Generic
     Vehicle Model       : Generic
     Vehicle Type        : None
     Engine Manufacturer : Generic
     Engine Manufactured : Thu Feb 02 01:38:31 MST 2015
     Engine Make         : Generic
     Engine Model        : Generic
     Engine Type         : 88 AKI
     Engine Cylinders    : 0
     Drive Train         : 2WD: Two-Wheel Drive
     ```

   - Write a static main method that tests two scenarios:

- One that prints a generic set of strings, like:

```
Manufacturer Name   : Honda
Manufactured Date   : Tue Jan 03 07:13:19 MST 2015
Vehicle Make        : Honda
Vehicle Model       : Prelude
Vehicle Type        : null
Engine Manufacturer : Honda
Engine Manufactured : Thu Feb 02 01:38:31 MST 2015
Engine Make         : H-Series
Engine Model        : H23A1
Engine Type         : 88 AKI
Engine Cylinders    : 4
Drive Train         : 2WD: Two-Wheel Drive
```

- One that accepts call parameters and returns the following:

```
Manufacturer Name   : Honda
Manufactured Date   : Tue Jan 03 07:13:19 MST 2012
Vehicle Make        : Honda
Vehicle Model       : Prelude
Vehicle Type        : null
Engine Manufacturer : Honda
Engine Manufactured : Thu Feb 02 01:38:31 MST 2012
Engine Make         : H-Series
Engine Model        : H23A1
Engine Type         : 88 AKI
Engine Cylinders    : 4
Drive Train         : 2WD: Two-Wheel Drive
```

10. Create a concrete class named Car that extends the Vehicle class with the following:

- Create the following private instance variables:

```
private Feature[] feature = new Feature[10];
private int carAxle;
```

- A public default constructor with no formal parameters and initialize all instance variables with generic literal values by using the super() call.
- A public overloaded constructor with a super() method call and instantiation of values for all of the variables defined above.
- Public methods to return formatted strings of the Internal and External features:

```
String getExteriorFeatures()
```

```
String getInteriorFeatures()
```

- These methods should display the following:

```
Exterior Features   : Exterior [Wood Panels]
                    : Exterior [Moonroof]
Interior Features   : Interior [AM/FM Radio]
                    : Interior [Air Conditioning]
```

- A public toString method that returns the following:

```
Manufacturer Name   : Honda
Manufactured Date   : Tue Jan 03 07:13:19 MST 2012
Vehicle Make        : Honda
Vehicle Model       : Prelude
Vehicle Type        : null
Engine Manufacturer : Honda
Engine Manufactured : Thu Feb 02 02:00:28 MST 2012
Engine Make         : H-Series
Engine Model        : H23A1
Engine Type         : 88 AKI
Engine Cylinders    : 4
Drive Train         : 2WD: Two-Wheel Drive
Features            : Interior [AM/FM Radio]
                    : Exterior [Wood Panels]
                    : Interior [Air Conditioning]
                    : Exterior [Moonroof]
Car Axle            : 2
```

- Write a static main method scenarios for default (no parameter) constructor and a full constructor, like:

```
public Car( String     vehicleManufacturer
, Date       vehicleManufacturedDate
, String     vehicleMake
, String     vehicleModel
, String     vehicleType
, Chassis    vehicleFrame
, Engine     vehicleEngine
, Feature[]  feature
, int        carAxle)
```