CE00527-5 Further Object Oriented Programming
Week 2 Tutorial

In this tutorial you will review association, and learn to use Enums. You will use the classes from tutorial 1 but create a new Project in NetBeans for this tutorial.

Part 1 – Association – Basic Exercise

1. This part of the tutorial builds on the first weeks tutorial. It is recommended that you create a new project and copy in the classes from tutorial 1.

   Design, build and test a class Customer which has String attributes to represent a Customer's name and contact details (e.g. a telephone number or email address).
   The class should have
   - one or more appropriate constructor(s)
   - methods to return the value of each attribute
   - a toString() method that returns relevant information about the Customer as a String.

   Test your Customer class by creating a TestCustomer class with a main method that creates a Customer object, calls each of its methods, and prints appropriate output.

2. Modify the Vehicle class you developed in Tutorial 1 so that the String attribute representing the customer name is replaced by an object reference to a Customer.

   Re-run the tests you created last week to test the Vehicle and Showroom classes. If you have changed any of the functionality or the interface (method signatures) of Vehicle and/or Showroom, you will need to update your tests.

Part 2 – Advanced exercise

3. Further modify the Vehicle class so that the date of manufacture of the vehicle and the date it was sold are stored as references to Date objects, rather than as Strings.

   Do not change the existing public interface (method signatures) of the Vehicle class, except for the getDate methods when you can return a Date instead of a String. Instead, you will need to add code to the constructor and other method bodies where necessary to convert between the String and Date forms of the date (hint – Date d1= new Date("20-mar-2013"); creates a Date object with the date 20 March 2013. This is a deprecated method but we can still use it, later we will see how we can use the Calendar and DateFormat classes to manipulate dates. Date d2=new Date(); creates todays date.)

If you like, you can also overload the existing constructor and any other relevant methods so one version takes the date parameter as a String, and the other as a Date object.

Rerun the tests of your Vehicle class, and test any methods you added.

Add a getAgeOfVehicle() method which calculates and returns the whole number of **weeks** elapsed since the manufacture of the Vehicle. Test this method, and use it in the toString() method of Vehicle.

Add a getVehiclesSoldRecently() method to the Showroom class which returns an ArrayList containing all the vehicles sold in the last 2 weeks (14 days).

Test this method (in the ShowroomDriver class) by first adding and buying some more vehicles, setting the dates sold of some to be within the last two weeks, and others to be more than two weeks.

Part 3 – Using Enums – Basic Exercise

4. Implement and test the Card class described in Week 2, lecture 1.

5. Modify the Card class so that it uses enumerated types to describe the card suit and rank.
   (Advice: Declare each enumerated type outside the Card class rather than inside the class)
   Each rank has an associated value, or point score, as follows:
   Ace: 1
   Two – Ten: Face value of card
   Jack, Queen, King: 10

Test your new version of the Card class.

Part 4 – The Hand class– Basic Exercise

6. Implement and test a class to manage a Hand of five cards (you can use the example given in Week 2, lecture 1 as a starting point). The Hand class should provide methods to
   - add a Card
   - get the total value of all the cards in the hand
   - return information about all the cards in the hand as a String

   Test all the methods of the Hand class.

Part 5 – The Deck class-– Advanced Exercise

7. Implement and test a Deck class to manage a collection of cards. The class should have a constructor that builds a complete pack of cards; that is 1-13 of each of the 4 suits. Use an array to hold the cards. For

now, the Deck class needs only to provide a toString() method which returns a list of all the cards in the deck. Test this method.

Note that all enums inherit a static values() method that returns an array containing all the constants of the enum. You can use this to iterate over all the enum constants. See the notes and Java documentation for more details.

You should put in your portfolio:

- **the code listing for the new Customer class, and the new versions of the Vehicle and Showroom class**
- **code listing for the test classes and test output for Customer, Vehicle and Showroom**
- **the design (class diagram, including associations) of the Card, Hand and Deck classes, and associated enums**
- **the code listing for each of these classes and enums**
- **code listing for the test classes and test output for Card, Hand and Deck**
- **Screen Shots are required for the testing of all the functionality.**

You may choose to include in your portfolio only the final versions of your Showroom/Vehicle/Customer and Card/Hand/Deck systems, provided that you are able to highlight the aspects asked for in each tutorial during the portfolio test. You must also show in your portfolio how you tested all the functionality of your classes.