Tutorial # 3 Writing Classes

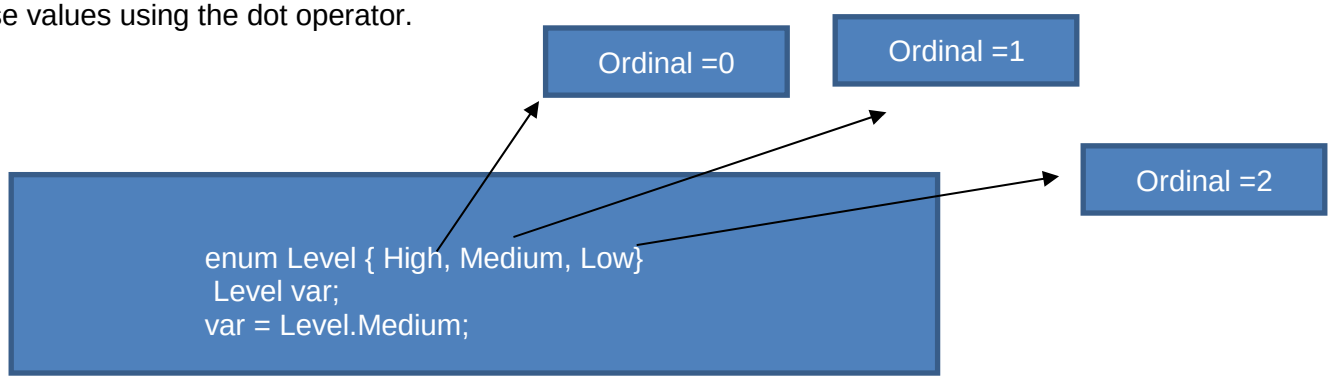
SOFE 2710U Object Oriented Programming and Design

Objective:

In this tutorial you will learn use how to use enumerated types and how to write your own Classes

Enumerated Types

In Java, variables can also be declared using enumerated types. In this context, enumerated types declaration lists a set of values for that variable. Once a variable is defined, you can easily access these values using the dot operator.



Wrapper classes use primitive data types as objects. The following table contains the wrapper class that is equivalent to every primitive data type.

Primitive Data Type	Wrapper Classes
byte	Byte
short	Short
Int	Integer
float	Float
double	Double
long	Long
char	Char
boolean	Boolean

Writing your own Classes

In the previous tutorials we have covered how to use the built in Java Classes. Today we will learn how to write our own classes and use them in programs. Classes, Objects, attributes and methods form the basis of Java Programming. Taking For instance, class vehicle instantiates object bus. A vehicle class has attributes such as Length, color and methods such as steer and brake. It is important to note that any objects have state (gets defined by the values of the declared data) and behavior (gets defined by the functionality of the method declared). Noting that a constructor is used to create a new object and it does not have a return type even void, whereas encapsulation is a mechanism where data and inner workings of objects are hidden from other classes. It can be accessed through setter and getter methods. Moreover, to achieve encapsulation instance variables of classes are declared in private. In the Following table visible modifiers such as public, default, private and protected are described.

Modifiers	Description
Public	This can be accessed by all classes
Default	Can be accessed by all classes within the same package
Private	Only accessed within the same class
Protected	This involves inheritance which will be discussed later

Method declaration points out the code that will be run when the method Is invoked. It is also important to note that when a variable is declared within a method, it is considered local and is only used within this method.

Problems:

1. Write a program to demonstrate an enumerated type called *Directions* {*North, South, West and East}*. Define variables *dir1, dir2, dir3, dir4, for dir3* use *direction type West* and *for dir 2 use direction type South* and Print out value, ordinal, name of *dir2* & *dir3.*

2. Write a class that defines an enumerated type named *Rank* with values `ace, two, three, four, five, six, seven, eight, nine, ten, jack, queen, king`.The main method should do the following:

a) Declare variables *highCard, faceCard, card1,* and *card2* of type Rank.

b) Assign highCard to be an ace, faceCard to be a jack, queen or king (your choice), and card1 and card2 to be two different numbered cards (two through ten - your choice).

c) Print a line, using the highCard and faceCard objects, in the following format:

```
A blackjack hand: ace and ....
```

The faceCard variable should be printed instead of the dots.

d) Declare two variables card1 Val and card2Val of type int and assign them the face value of your card1 and card2 objects. Use your card1 and card2 variables and the ordinal method associated with enumerated types. Remember that the face value of two is 2, three is 3, and so on so you need to make a slight adjustment to the ordinal value of the enumerated type.

e) Print two lines, using the card1 and card2 objects and the name method, as follows:

```
A two card hand:    (print card1 and card2)
Hand value:   (print the sum of the face values of the two cards)
```

3. Write a complete Java program that simulates the rolling of a pair of dice. For each die in the pair, the program should generate a random number between 1 and 6 (inclusive). It should print out the result of the roll for each die and the total roll (the sum of the two dice), all appropriately labeled. You must use the Random class.

4. The file *Distance.java* contains an incomplete program to compute the distance between two points. Recall that the distance between the two points (x1, y1) and (x2, y2) is computed by taking the square root of the quantity $(x1 - x2)^2 + (y1 - y2)^2$. The program already has code to get the two points as input. You need to add an assignment statement to compute the distance and then a print statement to print it out (appropriately labeled). Test your program using the following data: The distance between the points (3, 17) and (8, 10) is 8.6023... (lots more digits printed); the distance between (-33,49) and (-9, -15) is 68.352.…

```
// ********************************************************
//  Distance.java
//
//  Computes the distance between two points
// ********************************************************

import java.util.Scanner;

public class Distance
{
    public static void main (String[] args)
    {
      double x1, y1, x2, y2; // coordinates of two points
      double distance;      // distance between the points

      Scanner scan = new Scanner(System.in);
```

```
        // Read in the two points
        System.out.print ("Enter the coordinates of the first point " +
                  "(put a space between them): ");
        x1 = scan.nextDouble();
        y1 = scan.nextDouble();

        System.out.print ("Enter the coordinates of the second point: ");
        x2 = scan.nextDouble();
        y2 = scan.nextDouble();

        // Compute the distance

        // Print out the answer
    }
}
```

5. File *Deli.java* contains a partial program that computes the cost of buying an item at the deli. Save the program to your directory and do the following:

a) Study the program to understand what it does.

b) Add the import statements to import the DecimalFormat and NumberFormat classes.

c) Add the statement to declare *money* to be a NumberFormat object as specified in the comment.

d) Add the statement to declare *fmt* to be a DecimalFormat object as specified in the comment.

e) Add the statements to print a label in the following format (the numbers in the example output are correct for input of $4.25 per pound and 41 ounces). Use the formatting object money to print the unit price and total price and the formatting object fmt to print the weight to 2 decimal places.

***** CSDeli *****

Unit Price: $4.25 per pound
Weight: 2.56 pounds

TOTAL: $10.89

```
// ********************************************************
// DeliFormat.java
//
//Computes the price of a deli item given the weight
// (in ounces) and price per pound -- prints a label,
// nicely formatted, for the item.
//
//

*********************************************************

* import java.util.Scanner;



public class Deli
{
    //  ------------------------------------------------------------
    //  main reads in the price per pound of a deli item
    //  and number of ounces of a deli item then computes
    //  the total price and prints a "label" for the item
    //  ------------------------------------------------------------
    public static void main (String[] args)
    {
        final double OUNCES_PER_POUND = 16.0;

        double pricePerPound;          // price per pound
        double weightOunces;          // weight in ounces
        double weight;                // weight in pounds
        double totalPrice;     // total price for the
        item

        Scanner scan = new Scanner(System.in);

        // Declare money as a NumberFormat object and use the
        // getCurrencyInstance method to assign it a value

        // Declare fmt as a DecimalFormat object and instantiate
        // it to format numbers with at least one digit to the left of the
        // decimal and the fractional part rounded to two digits.

        // prompt the user and read in each input
        System. out. println ("Welcome to the CS Deli! ! \n ");

        System.out.print ("Enter the price per pound of your item:
        "); pricePerPound = scan.nextDouble();

        System.out.print ("Enter the weight (ounces):
        "); weightOunces = scan.nextDouble();

        // Convert ounces to pounds and compute the total
        price weight = weightOunces / OUNCES_PER_POUND;
        totalPrice = pricePerPound * weight;

        // Print the label using the formatting objects
        // fmt for the weight in pounds and money for the prices



    }
}
```