Module: Core Java

Session 5: Class Design and Encapsulation Practice

* This is a practice session; you will work on Classes and Encapsulation assignments.
* You can discuss your doubts with the trainer.

**Assignments:**

**Assignment 1 - Methods and documentation comments assignment**

## Objectives

* Be able to implement a void method
* Be able to implement a value-returing method

## Introduction

Methods are commonly used to break a problem down into small manageable pieces. A large task can be broken down into smaller tasks (methods) that contain the details of how to complete that small task. The larger problem is then solved by implementing the smaller tasks (calling the methods) in the correct order. This also allows for efficiencies, since the method can be called as many times as needed without rewriting the code each time.

In this assignment you will start with an incomplete file and implement several methods to solve specific tasks.

Finally, we will use documentation comments for each method, and generate HTML documents similar to the Java APIs that we have seen.

## Task 1: void Methods

1. Create a new project in your IDE
2. Create a file caleed “Geometry.java” in your project. Code is here:

**Geometry.java**

import java.util.Scanner;

/\*\*

This program demonstrates static methods

\*/

public class Geometry

{

public static void main (String [] args)

{

int choice; //the user's choice

double value = 0; //the value returned from the method

char letter; //the Y or N from the user's decision to exit

double radius; //the radius of the circle

double length; //the length of the rectangle

double width; //the width of the rectangle

double height; //the height of the triangle

double base; //the base of the triangle

double side1; //the first side of the triangle

double side2; //the second side of the triangle

double side3; //the third side of the triangle

//create a scanner object to read from the keyboard

Scanner keyboard = new Scanner (System.in);

//do loop was chose to allow the menu to be displayed first

do

{

//call the printMenu method

choice = keyboard.nextInt();

switch (choice)

{

case 1:

System.out.print("Enter the radius of the circle: ");

radius = keyboard.nextDouble();

//call the circleArea method and store the result in the //value variable

System.out.println("The area of the circle is " + value);

break;

case 2:

System.out.print("Enter the length of the rectangle: ");

length = keyboard.nextDouble();

System.out.print("Enter the width of the rectangle: ");

width = keyboard.nextDouble();

//call the rectangleArea method and store the result in the value // variable

System.out.println("The area of the rectangle is " + value);

break;

case 3:

System.out.print("Enter the height of the triangle: ");

height = keyboard.nextDouble();

System.out.print("Enter the base of the triangle: ");

base = keyboard.nextDouble();

//call the triangleArea method and store the result

//in the value variable

System.out.println("The area of the triangle is " + value);

break;

case 4:

System.out.print("Enter the radius of the circle: ");

radius = keyboard.nextDouble();

//call the circumference method and store

// the result in the value variable

System.out.println("The circumference of the circle is " + value);

break;

case 5:

System.out.print("Enter the length of the rectangle: ");

length = keyboard.nextDouble();

System.out.print("Enter the width of the rectangle: ");

width = keyboard.nextDouble();

//call the perimeter method and store the

// result in the value variable

System.out.println("The perimeter of the rectangle is " + value);

break;

case 6:

System.out.print("Enter the length of side 1 of the triangle: ");

side1 = keyboard.nextDouble();

System.out.print("Enter the length of side 2 of the triangle: ");

side2 = keyboard.nextDouble();

System.out.print("Enter the length of side 3 of the triangle: ");

side3 = keyboard.nextDouble();

//call the perimeter method and store the result in the value variable

System.out.println("The perimeter of the triangle is " + value);

break;

default:

System.out.println("You did not enter a valid choice.");

}

keyboard.nextLine(); //consumes the new line character after the //number

System.out.println("Do you want to exit the program (Y/N)?: ");

String answer = keyboard.nextLine();

letter = answer.charAt(0);

}while (letter != 'Y' && letter != 'y');

}

}

This program will compile, but when you run it, it doesn’t appear to do anything except wait. That is because it is waiting for user input, but the user doesn’t have the menu to choose from yet. We will need to create this.

1. Above the main method, but in the Geometry class, create a static method called *printMenu()* that has no parameter list and does not return a value. It will simply print out instructions for the user with a menu of options for the user to choose from. The menu should appear to the user as:

This is a geometry calculator

Choose what you would like to calculate

1. Find the area of a circle

2. Find the area of a rectangle

3. Find the area of a triangle

4. Find the circumference of a circle

5. Find the perimeter of a rectangle

6. Find the perimeter of a triangle

Enter the number of your choice:

1. Add a line in the main method that calls the printMenu method as indicated by the comments.
2. Compile, debug, and run. You should be able to choose any option, but you will always get 0 for the answer. you will fix this in the next task.

## Task 2: Value-Returning Methods

1. Write a static method called *circleArea()* that takes in the radius of the circle and returns the area using the formula

***A*=π*r*2**

1. Write a static method called *rectangleArea()* that takes in the length and width of the rectangle and returns the area using the formula

***A*=*lw***

1. Write a static method called *triangleArea()* that takes in the base and height of the triangle and returns the area using the formula

***A*=1/2*bh***

1. Write a static method called *circleCircumference()* that takes in the radius of the circle and returns the circumference using the formula

***C*=2*pr***

1. Write a static method called *rectanglePerimeter()* that takes in the length and the width of the rectangle and returns the perimeter of the rectangle using the formula

***P*=2*l*+2*w***

1. Write a static method called *trianglePerimeter()* that takes in the lengths of the three sides of the triangle and returns the perimeter of the triangle which is calculated by adding up the three sides.

## Task 3: Calling Methods

1. Add lines in the main method in the GeometryDemo class which will call these methods. The comments indicate where to place the method calls.
2. Below, write some sample data and hand calculated results for you to test all 6 menu items.
3. Compile, debug, and run. Test out the program using your sample data.

## Task 4: Java Documentation

1. Write javadoc comments for each of the 7 static methods that you just wrote. They should include
   1. A one line summary of what the method does.
   2. A description of what the program requires to operate and what the result of that operation is.
      1. @param listing and describing each of the parameters in the parameter list (if any).
      2. @return describing the information that is returned to the calling statement (if any).
2. Generate the documentation. You can use the following command to create a subdirectory named *docs* in your project directory, and generate the files within it.

% javadoc -d docs Geometry.java

Or you can use Eclipse IDE...export javadoc from file menu to generate documentation.

1. To view the resulting documentation, simple load the generated *index.html* located inside the docs directory.

**Assignment 2 - Static Concepts- Variable, method and block Assignment**

## Objectives

1. How to create static variables
2. How to create static blocks
3. How to create static methods
4. How to create objects for classes having private constructors, outside the class

## Introduction

In this small assignment you will implement a Loan class with static methods, variables, private constructors, etc. You will also create LoanTest class to test this application.

## Task 1: Create a Loan class for the class diagram shown

|  |
| --- |
| Loan |
| - loanNo : int  - accountNo : int  - customerNo: int  - loanAmount: float  - loanDuration: int  - interest : float |
| +Loan()  +Loan(int accountNo , int customerNo,int  loanDuration ,float loanAmount, float interest)  +calculateInstallments() : float  +getAccountNo() : int  +getCustomerno() : int  +getLoanAmount() : float  +getInterest() : float  +getInterest() : float  +setAccountNo(int accountNo) : void  +setCustomerNo(int customerNo) : void  +setLoanDuration(int loanDuration) : void  +setLoanAmount(float loanAmount) : void  +setInterest(float interest) : void |

Add a static variable loanCounter to the Loan class.

//loanCounter counts the total number of objects

static int loanCounter;

In both the constructors of the loan class add the following statement

//To count the total number of instances

loanCounter++;

In the main method of the loan class create some instances of the class loan using both default and overloaded constructors( may be five or six instances).

In the main method of the loan class after creating the instances add the following statement.

//To print the count of instances created in the main method

System.out.println("Number of instances :"+loanCounter);

Compile and execute the program. You have just experienced that how to create static variables

## Task 2: Create static method and private constructor

Modify the above class and make the both the constructors as private. Also remove the main method of the loan class. Create one more class TestLoan.java like this

public class TestLoan {

public static void main(String[] args) {

//To-Do: create the instances of the Loan class

System.out.println("Total number of objects :" +Loan.loanCounter);

}

}

Compile and execute the above class. Above code will not compile as the object of a class cannot be created outside the class if the constructors are private. Static methods can be used to create objects for the classes having private constructors

Add a static method to the Loan class as shown. Replace the To-Do comments with the code

public static Loan getLoanInstance(){

//To-Do: create the object of loan class and return it

}

Modify the TestLoan.java class so that the instances of Loan class are created using the getLoanInstance() static method .

public class TestLoan {

public static void main(String[] args) {

//To-Do: create the instances of the Loan class //using getInstance() method of Loan class

System.out.println("Total number of objects :" +Loan.loanCounter);

}

}

Modify the Loan.java class to make the loanCounter variable as private. Save the Loan class and again compile the TestLoan.java class. Create a public method getNumberOfObjects() in the Loan class that will return the number of objects created of Loan class.

public int getNumberOfObjects(){

return loanCounter;

}

Modify the TesLoan.java such that , instead of Loan.loanCounter use getNumberOfObejcts() method in the SOP statement .