INSTITUTE OF ACCOUNTANCY ARUSHA

DEPARTMENT OF INFORMATICS

ITT07322: OBJECT-ORIENTED PROGRAMMING

LAB WORKSHEET

Introduction

This lab is intended to equip students with object-oriented programming style in developing computer software.

Competency Aimed (Enabling Learning Outcome)

- Use object-oriented programming paradigm in developing computer software

Specific Objectives (Sub-Enabling Learning Outcomes)

(i) Employ object-oriented programming concepts in constructing computer software

(ii) Apply object-oriented approach in writing, compiling and running computer programs

(iii) Use error handling techniques in writing fault-tolerant programs

References

- Lecture handouts
- Deitel, H. M. & Deitel, P. J. (2014). Java: How to program. 10th Ed. Upper Saddle River, NJ: Pearson Education
- Horstmann, C. S. (2015). Core Java Volume I Fundamentals. 10th Ed. Upper Saddle River, NJ: Pearson Education.
- Wu, C. (2009). An Introduction to object-oriented programming with Java. New York City, NY: McGraw-Hill Higher Education
- Bloch, J. (2008). Effective java: Programming language guide. Upper Saddle River, NJ: Addison-Wesley

- Mohan, P. (2013). Fundamentals of Object-Oriented Programming in Java. Scotts Valley, CA: CreateSpace Independent Publishing Platform.

Practical Questions

Topics: Introduction to OOP and Java Operators

1. Modify the Java first program so that the program displays your name.
2. Write Java application that displays the sum of two integers.
3. Write Java application that obtains two integers typed by a user at the keyboard, computes the sum of these values and outputs the result.
4. Write an application that inputs three integers from the keyboard and prints the sum, average, product and the biggest number of these numbers.
5. Write an application that asks user to enter the radius of a circle, calculates the area and outputs the result.

Topic: Control Statements

6. Write a Java application that checks whether the number entered by user is an odd.
7. Write a Java application that divides two numbers but checks whether the divisor is not zero.
8. Write a Java application that displays even numbers between 1 and 100.
9. Write a Java application that calculates the sum of the first 200 counting integers.
10. Write a Java application that uses looping to print the following table of values:

```
N       10*N    100*N   1000*N

1       10      100     1000
2       20      200     2000
3       30      300     3000
4       40      400     4000
5       50      500     5000
```

Topic: Arrays

11. Write an application that initializes all array elements to 10.00.
12. Write Java application that obtains data from keyboard and write into an array.
13. Write Java application that displays all elements of an array.
14. Write Java application that calculates and displays the sum and average of all elements of an array.

15. If the float array numbers is initialized as follows: float [] numbers = {50, 10, 20, 60, 120, 90}; Write an application that displays the smallest and largest elements in the float array numbers.

Topic: Classes and objects

16. Create a class called Box that includes three pieces of information as instance variables – a length (type double), a width (type double) and a depth (type double). Your class should have a constructor that initializes the three instance variables. Provide a method called calculateVolume for calculating the volume of the box. Write a test application named BoxTest that demonstrates class Box's capabilities.

17. Create a class called Arithmetic that includes two pieces of information as instance variables – a number1 (type float) and number2 (type float). Your class should have a constructor that initializes the two instance variables. Provide a method called add that calculates the sum of two numbers. Provide another method called divide which calculates the quotient of the two numbers. Write a test application named ArithmeticTest that demonstrates class Arithmetic's capabilities.

18. Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance variables – a part number (type String), a part description (type String), a quantity of the item being purchased (type int) and a price per item (double). Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable. In addition, provide a method named getInvoiceAmount that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test application named InvoiceTest that demonstrates class Invoice's capabilities.

Topic: Inheritance

19. Create a class called Box that includes three pieces of information as instance variables – a length (type float), a width (type float) and a depth (type float). Your class should have a constructor that initializes the three instance variables. Provide a method called calculateVolume for calculating the volume of the box. Create another class called ColouredBox inherits from class Box. The class ColouredBox includes one more piece of information as its instance variable – colour (type String). Write a test application named ColouredBoxTest that demonstrates class ColouredBox's capabilities.

20. Draw an inheritance hierarchy for students at a university. Use Student as the superclass of the hierarchy, and then extend Student with classes UndergraduateStudent and GraduateStudent. Continue to extend the hierarchy as deep (i.e., as many levels) as possible. After drawing the hierarchy, write a test application called StudentTest that creates objects of each class (Student, UndergraduateStudent and GraduateStudent) and tests their member methods.

Topic: Exception Handling

21. Write a Java application that prompts the user for two integers, calculates the sum, and prints the sum. The application uses the exception handling so that if the user makes a mistake, the program catches and deals with an exception that arises – in this case, allowing the user to try to enter the input again.

22. Write a Java application that prompts the user for two floating-point numbers, calculates the quotient, and prints the quotient. The application uses the exception handling so that if the user makes a mistake, the program catches and deals with exceptions that arise – in this case, allowing the user to try to enter the input again.