OBject oriented programming with java

Lab manual

Aymen Khan

Aymkhan498@gmail.com

**Course Title:** Object Oriented Programming

**Course Code:**

**Course Outcomes:**

At the end of the course the student should be able to:

1. Apply Object Oriented Programming concepts to solve a given problem.
2. Apply design patterns to design a solution for a given problem.
3. Apply inheritance, polymorphism and exception handling mechanism to implement reusable, robust java programs.
4. Implement user interface java programs for a given scenario.

**List of Practical**

|  |  |  |
| --- | --- | --- |
| **Sr.#** | **Week** | **Topics** |
| 1 | One | Introduction to IDE-NetBeans, Getting Started with Java |
| 2 | Two | Java Basics (Input/output, variable declaration and initialization, strings and arrays) |
| 3 | Three | Java Basics (Selection Structure and Iterative Structure, functions) |
| 4 | Four | OOP (creating classes, objects, constructors) |
| 5 | Five | access modifiers, inheritance |
| 6 | Six | multiple/multilevel inheritance |
| 7 | Seven | Function overriding |
| 8 | Eight | Polymorphism |
| 9 | Nine | abstract classes and interfaces |
| 10 | Ten | exception handling and Java file handling |

**OOP (creating classes, objects, constructors)**

**LAB-4**

Java is an object-oriented programming language. Everything in Java is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has attributes, such as weight and color, and methods, such as drive and brake. A Class is like an object constructor, or a "blueprint" for creating objects.

**Create a Class**

To create a class, use the keyword class:

**MyClass.java**

Create a class named "MyClass" with a variable x:

public class MyClass {

int x = 5;

}

**Create an Object**

In Java, an object is created from a class. We have already created the class named MyClass, so now we can use this to create objects.

To create an object of MyClass, specify the class name, followed by the object name, and use the keyword new:

**Example**

Create an object called "myObj" and print the value of x:

public class MyClass {

int x = 5;

public static void main(String[] args) {

MyClass **myObj** = new MyClass();

System.out.println(myObj.x);

}

}

Create an object called "myObj" and print the value of x:

public class MyClass {

int x = 5;

public static void main(String[] args) {

MyClass **myObj** = new MyClass();

System.out.println(myObj.x);

}

}

**Using Multiple Classes**

You can also create an object of a class and access it in another class. This is often used for better organization of classes (one class has all the attributes and methods, while the other class holds the main() method (code to be executed)). Remember that the name of the java file should match the class name. In this example, we have created two files in the same directory/folder:

* MyClass.java
* OtherClass.java

#### **MyClass.java**

public class MyClass {

int x = 5;

}

#### **OtherClass.java**

class OtherClass {

public static void main(String[] args) {

MyClass **myObj** = new MyClass();

System.out.println(myObj.x);

}

}

**Constructor**

A constructor in Java is a **special method** that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes:

#### **Example**

Create a constructor:

// Create a MyClass class

public class MyClass {

int x; // Create a class attribute

// Create a **class constructor** for the MyClass class

public MyClass() {

x = 5; // Set the initial value for the class attribute x

}

public static void main(String[] args) {

MyClass myObj = new MyClass(); // Create an object of class MyClass (This will **call the constructor**)

System.out.println(myObj.x); // Print the value of x

}

}

## **Constructor Parameters**

Constructors can also take parameters, which is used to initialize attributes. The following example adds an int y parameter to the constructor. Inside the constructor we set x to y (x=y). When we call the constructor, we pass a parameter to the constructor (5), which will set the value of x to 5:

## **Example**

public class MyClass {

int x;

public MyClass(int y) {

x = y;

}

public static void main(String[] args) {

MyClass myObj = new MyClass(5);

System.out.println(myObj.x);

}

}

You can have as many parameters as you want:

## **Example**

public class Car {

int modelYear;

String modelName;

public Car(int year, String name) {

modelYear = year;

modelName = name;

}

public static void main(String[] args) {

Car myCar = new Car(1969, "Mustang");

System.out.println(myCar.modelYear + " " + myCar.modelName);

}

}

**Task-1**

Write a program that have a class (PRODUCT) that contain three data members and three member function first is constructor that set the default suitable values and second is in() that can be set and receive values from the user and multiply three values in one variable and third function that show the product of three values. Test this program in main using minimum five object of PRODUCT.

**Task-2**

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 data members a part (type string), a part description (type string), a quantity of the item being purchased (type int) and a price per item (type int). Your class should have a constructor that initializes the four data members. Provide a set and a get function for each data member. In addition, provide a member function named (getInvoiceAmount) that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as an int 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. Write a test program that demonstrates class Invoice's capabilities.

**Assignment Question**

Design and implement a class DATE that implements the day of the week in a program. The class DATE should store the day, such as Sun for Sunday. The program should be able to perform the following operations on an object of type DATE:

a. Set the day.

b. Print the day.

c. Return the day.

d. Return the next day.

e. Return the previous day.

f. Calculate and return the day by adding certain days to the current day.

For example, if the current day is Monday and we add 4 days, the day tobe returned is Friday. Similarly, if today is Tuesday and we add 13 days,the day to be returned is Monday.

g. Add the appropriate constructors.