Shifa Tameer-e-Millat University

شفا تعمیرِ ملّت یونیورسٹی

Lab Manual for Object-Oriented Programming

Lab 2

Basic Concepts of JAVA Programming

Table of Contents

Lab2: Basic Concepts of JAVA Programming

1. Introduction

You have looked at the basic characteristics of Java and the benefits of using Java over C++. The objective of this lab is to get familiar with the basic concepts of java like implementation of Access Specifiers, Constructors and Destructors, Constructor Overloading.

a) Revise Lecture No. 5 and 6
b) Relevant Lecture Material
c) Text Book: Java: How to Program by Paul J. Deitel, Harvey M. Deitel
    1. Read pages: 11-15, 55-63
    2. Revise the object-oriented concepts.

2. Activity Time boxing

Table 1: Activity Time Boxing

Task No.	Activity Name	Activity time	Total Time
5.1	Evaluation of Design	20mins	20mins
6.2	Setting-up Path for JDK	30mins	30mins
6.2	Walkthrough Tasks	30mins	30mins
7	Practice tasks	15mins for each task	60mins
8	Evaluation Task	60mins for all assigned task	30mins

3. Objective of the experiment

- To get basic understanding of Java programming
- To write simple class and how to run and compile it
- To get an understanding of basic java concepts.
- To understand the command line arguments and their purpose.

4. Concept Map

This section provides you the overview of the concepts that will be discussed and implemented in this lab.

4.1 Access Specifiers

There are four types of Java access modifiers:

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

Encapsulation

Encapsulation is one of the fundamental principles of object-oriented programming (OOP) and it refers to the bundling of data (attributes or properties) and methods (functions or procedures) that operate on the data into a single unit, called a class. Encapsulation hides the internal state of an object from the outside world and only exposes the necessary functionalities through well-defined interfaces.

Getters and Setters

Getters and setters are methods used to access and modify the private attributes of an object in object-oriented programming. Getters retrieve the value of an attribute, providing read-only access, while setters modify the value of an attribute, providing write-only or read-write access. By encapsulating attribute access within methods, getters and setters enforce data encapsulation and enable controlled interaction with object attributes.

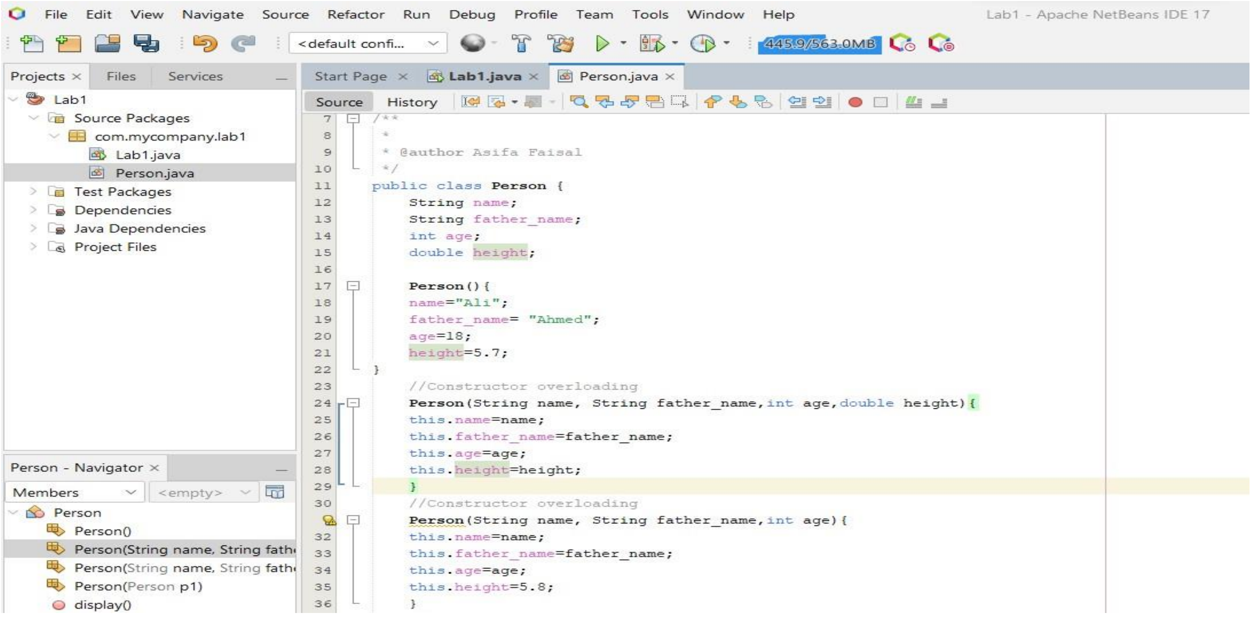
```
public int getAge(){
    return age;
}
public void setAge(int a){
    age = a;
}
public String getName(){
    return name;
}
```

```
public class Human {
    private int age=20;
    private String name="Waqas";
    public Human(){ //Default constructor
        age = 12;
        name = "Waqas";
    }
    public Human(int a, String n){ //parametrized constructor
        age = a;
        name = n;
    }
    public int getAge(){
        return age;
    }
    public void setAge(int a){
        age = a;
    }
    public String getName(){
        return name;
    }

    public void setName(String n) {
        name = n;
    }
}
```

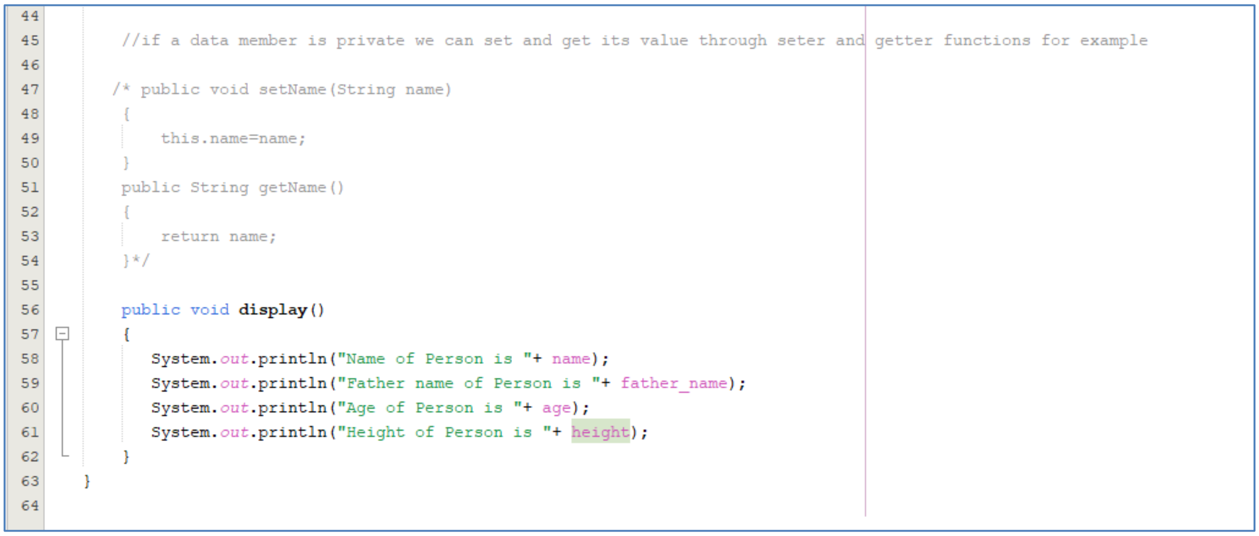
```
public class Main {

    public static void main(String[] args) {
        Human h1 = new Human();
        Human h2 = new Human();
        Human h3 = new Human(27, "Farooq");
        h1.setName("Ubaid:");
        h1.setAge(27);
        System.out.print(h1.getName());
        System.out.println(h1.getAge());
        System.out.println(h2.getName()+ ": \t" +
h2.getAge());
        System.out.print(h3.getName()+ ": \t" +
h3.getAge());


//          h1.name = "Waqas";
//          h1.age = 20;
//          System.out.println(h1.getName());
//          System.out.println(h1.getAge());
```

concepts discussed in concept map.

Person.java

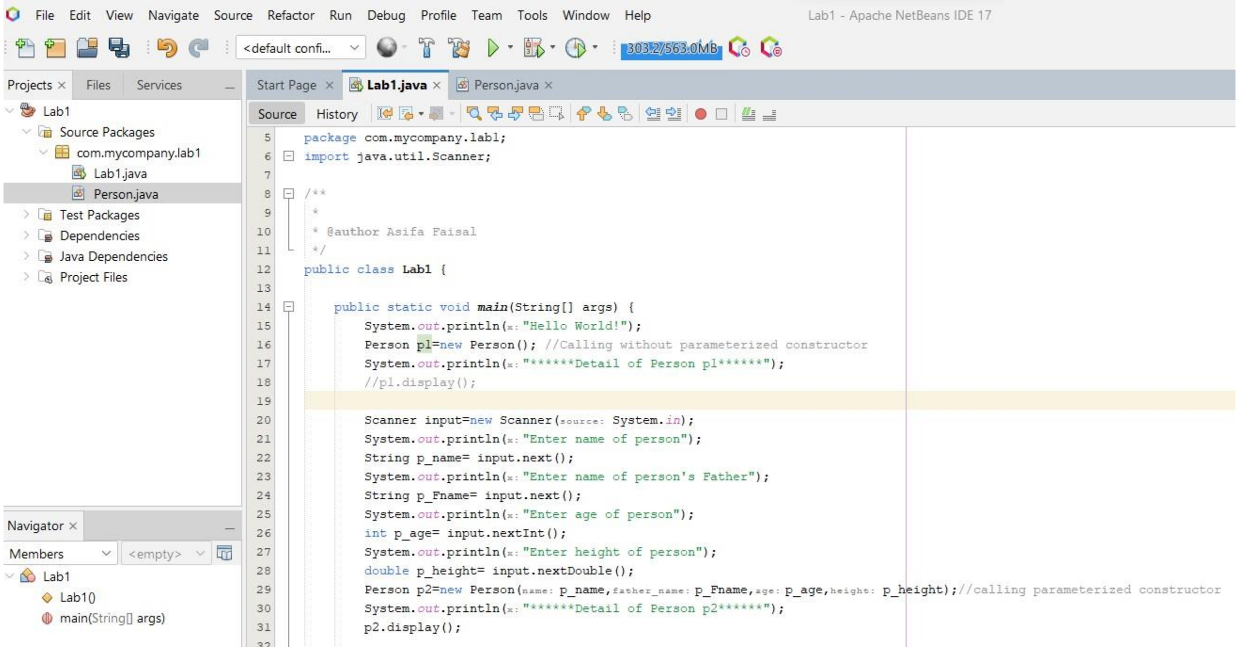


```
/**
 *
 * @author Asifa Faisal
 */
public class Person {
    String name;
    String father_name;
    int age;
    double height;

    Person(){
    name="Ali";
    father_name= "Ahmed";
    age=18;
    height=5.7;
}
    //Constructor overloading
    Person(String name, String father_name,int age,double height){
    this.name=name;
    this.father_name=father_name;
    this.age=age;
    this.height=height;
    }
    //Constructor overloading
    Person(String name, String father_name,int age){
    this.name=name;
    this.father_name=father_name;
    this.age=age;
    this.height=5.8;
    }
```



```java

        //if a data member is private we can set and get its value through seter and getter functions for example

       /* public void setName(String name)
        {
            this.name=name;
        }
        public String getName()
        {
            return name;
        }*/

        public void display()
        {
            System.out.println("Name of Person is "+ name);
            System.out.println("Father name of Person is "+ father_name);
            System.out.println("Age of Person is "+ age);
            System.out.println("Height of Person is "+ height);
        }
    }

```

Lab1.java (main class)



```java
package com.mycompany.lab1;
import java.util.Scanner;

/**
 *
 * @author Asifa Faisal
 */
public class Lab1 {

    public static void main(String[] args) {
        System.out.println(x: "Hello World!");
        Person p1=new Person(); //Calling without parameterized constructor
        System.out.println(x: "******Detail of Person p1******");
        //p1.display();

        Scanner input=new Scanner(source: System.in);
        System.out.println(x: "Enter name of person");
        String p_name= input.next();
        System.out.println(x: "Enter name of person's Father");
        String p_Fname= input.next();
        System.out.println(x: "Enter age of person");
        int p_age= input.nextInt();
        System.out.println(x: "Enter height of person");
        double p_height= input.nextDouble();
        Person p2=new Person(name: p_name,father_name: p_Fname,age: p_age,height: p_height);//calling parameterized constructor
        System.out.println(x: "******Detail of Person p2******");
        p2.display();
```

8. Practice Tasks

This section will provide more practice exercises which you need to finish during the lab. You need to finish the tasks in the required time. When you finish them, submit your task to your instructor.

8.1 Practice Task 1 [Expected time = 15mins]

Create a class Student. The data members of the class are regNo (string), name (String), age (int), CGPA (double). Then Implement member functions to;

1. Make CGPA private member.
2. Create a parameterized constructor that will set only name and regNo through parameters and other two will be taking from user in constructor too.
3. Make setter getter for private data members.
4. Display the values of all data members.

Create another class TestStudent.java to test Student class. Take input via scanner class.

8.2 Practice Task 2

Write a class called Programs. Initialize a variable name **Input Number,** initialize it using **constructors** by taking Input from user (Come from Main Class). In this class Create the following function:

1. Create getter and setter for variable.
2. Make the Function which can check and display the number is even or odd.
3. Make the function which can compute the factorial of this number
4. Make the function which can check and display the number is prime or not.

In Main Class, Create the object of programs class. Take the input from user using joptionpane class. The user input pass via constructor. After that make a Menu like given below

************************MENU*************************************

**********Press 1 for to check whether the Number is even or odd***********

**************Press 2 for to find the factorial of a Number ****************

**********Press 3 for to check whether the Number is Prime or Not***********

Take the choice from user using Scanner class and passes to Switch and calls the functions in respective cases. The Switch Concept of Java and C++ is same.

8.3 Out comes

After completing this lab, student will be able to setup JDK. He/ She will also be able to compile and run basic Java programs.

8.4 Testing

This section provides you the test cases to test the working of your program. If you get the desired mentioned outputs for the given set of inputs then your program is right.

Test Cases for Practice Task-1

Sample Inputs	Sample Outputs

Set the following values
Name = Ali
RegNo = BC113100
Age = 23
CGPA = 3.00
	Name = Ali
	RegNo = BC113100
	Age = 23
	CGPA = 3.00

Test Cases for Practice Task-2

Sample Inputs	Sample Outputs

Length: 4
width: 10
	Length = 5
	Width = 11
	Area = 55

9. Evaluation Task (Unseen) [Expected time = 30mins for tasks]

The lab instructor will give you unseen task depending upon the progress of the class.

10. Evaluation criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

Table 3: Evaluation of the Lab

Sr. No.	Task No	Description	Marks
1	4	Problem Modeling	20
2	6	Procedures and Tools	10
3	7	Practice tasks and Testing	35
4	8	Evaluation Tasks (Unseen)	20
5		Comments	5
6		Good Programming Practices	10

11. Further Reading

This section provides the references to further polish your skills.

11.1 Books

Text Book:

- *Java: How to Program by Paul J. Deitel, Harvey M. Deitel. Eighth Edition*
- *Java Beginners Guide: http://www.oracle.com/events/global/en/java-outreach/resources/java-a-beginners-guide-1720064.pdf*