National University of Computer & Emerging Sciences,
Karachi
Computer Science Department
Spring 2022, Lab Manual – 03

Course Code: CL-217	Course : Object Oriented Programming Lab
Instructor(s) :	Abeer Gauher, Hajra Ahmed, Syed Zain ul Hassan

LAB - 3

Classes & Objects in Java

CONTENTS:

Class:

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical. A class in Java can contain:

- Fields
- Methods
- Constructors
- Blocks
- Nested class and interface

The syntax to declare the class is:

```
class <class_name>{
    //class body
}
```

```
class lab3 {

}
```

Class declaration is enclosed within code blocks. In other words, the body of the class is enclosed between the area between the curly braces. In the class body, you can declare data members (also called as fields or instance variable), member functions (also called as behaviors or instance methods) and constructors or destructors.

```
package com.mycompany.lab3;

public class Student {
    /*data members */
    int StudentId;
    String StudentName;
    float Marks;

    /* Constructor */
    void Student(int id,String name,float marks){
        StudentId=2;        //initializing a data member with static value
        StudentId=id;       // initializing a data member with dynamic value
        StudentName=name;
        Marks=marks;
    }

    /* member function */
    void display(){
        System.out.println("Student ID = "+StudentId);
        System.out.println("Student Name = "+StudentName);
        System.out.println("Marks = "+Marks);
    }
}
```

Figure 1: structure of a class

Methods or member functions in a class can be of any type given below:

<return_type> function_name (<argument 1, argument2, ……., argumentN>) {}

<return_type> function_name (void) {}

void function_name (<argument 1, argument2, ……., argumentN>) {}

void function_name (void) {}

function_names must not be a Java keyword.

A class can have different methods and constructors. Constructors are specialized methods which are called only when an object is created. Constructors do not have a return type and they are of multiple types like default constructors (that do not accept any arguments) and parameterized constructors (that accepts arguments). We will discuss them further in details in lab 4. If a class does not have a constructor then Java invokes a builtin default constructor for object creation.

```
/* Constructor */
void Students(){
    StudentId=1;    //initializing a data member with static value
    StudentName="Harry Potter";
    Marks=12.6f;
}
void Student(int id,String name,float marks){
    StudentId=id;      // initializing a data member with dynamic value
    StudentName=name;
    Marks=marks;
}
```

Figure 2: Constructors

Object:

An entity is a real-world entity that has state and behavior e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical (tangible and intangible). The example of an intangible object is the banking system. An object has three characteristics:

- State: represents the data (value) of an object.
- Behavior: represents the behavior (functionality) of an object such as deposit, withdraw, etc.
- Identity: An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

An object is created using the new operator. On encountering the new operator, JVM allocates memory for the object and returns a reference or memory address of the allocated object. The reference or memory address is then stored in a variable. This variable is also called as reference variable. The syntax for creating an object is as follows:

<class_name> <object_name> = new <classname>();

Where,

new: Is an operator that allocates the memory for an object at runtime.
object_name(or reference variable): Is the variable that stores the reference of the object

Creation of an object involves:

1) Declaration of reference variable
2) Creation of object and assigning its reference value to reference variable

```
/*object creation and refernce allocation @same time */
Student s = new Student();

/*object creation and refernce allocation in different steps */
Student s2;  // here s2= NULL initialy
s2=new Student(); // memry alocation and refrence assigned to s2
```

Figure 3: object creation

Example 1:

```
import java.util.Scanner;
class Student {
    /*data members */
        int StudentId;
        String StudentName;
        float Marks;
    /* Constructor */
    public Student() {
        StudentId=1;    //initializing a data member with static value
        StudentName="Harry Potter";
        Marks=12.6f;
    }
    public Student(int id, String n, float m){
            StudentId=id;      // initializing a data member with dynamic value
            StudentName=n;
            Marks=m;
    }
    /* member function */
    void display(){
            System.out.println("Student ID = "+StudentId);
            System.out.println("Student Name = "+StudentName);
            System.out.println("Marks = "+Marks);
    }
}
class MAIN{
    public static void main(String[] args) {
        Scanner scan=new Scanner(System.in);
        /* take user inputs */
        System.out.print("Enter ID\t");
        int id = scan.nextInt();
        System.out.print("Enter Name\t");
        String n=scan.next();
        System.out.print("Enter Marks\t");
        float m=scan.nextFloat();
        /*object creation*/
        Student s1=new Student(id,n,m); // parameterized constructor
        Student s2=new Student(); // default constructor
        /*calling a member function via object name */
        s2.display();
        s1.display();
    }}
```

Figure 4: student class

Access Modifiers in Java:

There are four types of Java access modifiers:

Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

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.

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.

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.

Access Modifier	Within class	Within package	Outside class	Outside package
Public	Yes	Yes	Yes	Yes
Private	Yes	No	No	No
Protected	Yes	Yes	Yes	No
default	Yes	Yes	No	No

Example 2:

```
import java.util.Scanner;

public class Access_Modifiers {
    public int a=2;
   // private int b=23;
    protected int c=234;
    int d=2345;
}
class MAIN{
    public static void main(String[] args) {
        Scanner scan=new Scanner(System.in);
        Access_Modifiers a=new Access_Modifiers();
        System.out.println(a.a);
     // System.out.println(a.b); // Private member will not be accessed outside class so it will give error
        System.out.println(a.c); //protected members can be accessed with in the package only
        System.out.println(a.d);
    }
}
```

```
"C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-ja
2
234
2345

Process finished with exit code 0
```

Figure 5: access Modifiers

Accessors & Mutators in Java:

An **Accessor** method is commonly known as a get method or simply a getter. A property of the object is returned by the accessor method. They are declared as public. A naming scheme is followed by accessors, in other words they add a word to get in the start of the method name.

They are used to return the value of a private field. The same data type is returned by these methods depending on their private field.

<return_datatype> <function_name>() {}

A **Mutator** method is commonly known as a set method or simply a setter. A Mutator method mutates things, in other words change things. It shows us the principle of encapsulation. They are also known as modifiers. They are easily spotted because they started with the word set. They are declared as public. Mutator methods do not have any return type and they also accept a parameter of the same data type depending on their private field. After that it is used to set the value of the private field.

Void <function_name>(arguments){}

lab3.java

```
import java.util.Scanner;
class Student {
    /*data members */
        int StudentId;
        String StudentName;
        float Marks;
    /* accessors */
    int getStudentId(){
        return StudentId;
    }

    String getStudentName(){
        return StudentName;
    }

    float getMarks(){
        return Marks;
    }
    /* Mutators */
    void setStudentId(int id){
        StudentId=id;
    }
    void setStudentName(String Name){
        StudentName=Name;
    }
    void setMarks(float m){
        Marks=m;
    }
```

Figure 6: accessors & mutators in Java

```
initialized values are
id = 23
Name = Hajra
Marks = 23.6
After updating the name data is :
initialized values are
id = 23
Name = Ahmed
Marks = 23.6
Class transformation time: 0.0071585s for 204 classes or 3.
```

Figure 7: output

TASK – 01:

Write a class named Car that has the following data members:

- yearModel – an int field that hold the car's year model.
- make – a String field that holds the make of the car.
- speed – an int field that holds the car's current speed.

The class also should have the following constructor and other methods:

- constructor – that accepts the car's year model and make as arguments. These values should be assigned to the object's yearModel and make fields. The constructor also should assign 0 to the speed field.
- Accessors. Appropriate accessor methods should get the values stored in an object's yearModel, make and speed fields.
- accelerate. The accelerate method should add 5 to the speed field each time it is called.
- brake. The brake method should subtract 5 from the speed field each time it is called.

Demonstrate the class in a program that creates a Car object, and then calls the accelerate method five times. After each call to the accelerate method, get the current speed of the car and display it. Then call the brake method five times. After each call to the brake method, get the current speed of the car and display it.

TASK – 02:

Write a Java class Book with following features:

- Instance variables:
    - **title** for the title of book of type String.
    - **author** for the author's name of type String.
    - **price** for the book price of type double.
- Constructor:
    - **public Book (String title, Author name, double price)**: A constructor with parameters, it creates the Author object by setting the the fields to the passed values.
- Instance methods:

- **public void setTitle(String title)**: Used to set the title of book.
- **public void setAuthor(String author)**: Used to set the name of author of book.
- **public void setPrice(double price)**: Used to set the price of book.
- **public String getTitle()**: This method returns the title of book.
- **public String getAuthor()**: This method returns the author's name of book.
- **public String toString()**: This method printed out book's details to the screen

Write a separate class **BookDemo** with a main () method creates a Book titled "Great Expectations" with author Charles Dickens and price 79.75.

TASK – 03:

Create a class called **Account** that includes three pieces of information as instance variables

- Account title (type String)
- Balance (Int/Float)
- Account number (Int)

Ask the user to input the initial balance. If the initial balance is less than 0, display an error message.

Write a test application named Account that demonstrates the class capabilities. Create two functions withdraw and deposit.

If the balance is 0, then user will not be able to withdraw any amount from the account. Display the updated balance after the user withdraws or deposits money.

TASK – 04:

- Write a program that describes a class Student. It should have instance variables to record name, age, StudentID, Course Name and Course teacher for OOP.
- Create a Student object. Set and display its instance variables.

TASK – 05:

Write a program that creates a class circle with instance variables for radius. Initialize the variable using a constructor dynamically. Create a function calculate area and display the area for a circle.