Sukkur Institute of Business Administration University

Department of Computer Science

Object Oriented Programming using Java

BS – II (CS/AI/SE)

Spring-2024

Lab # 05: Diving into the Ocean of Classes, Objects, and Methods

Instructor: Nimra Mughal

Objectives

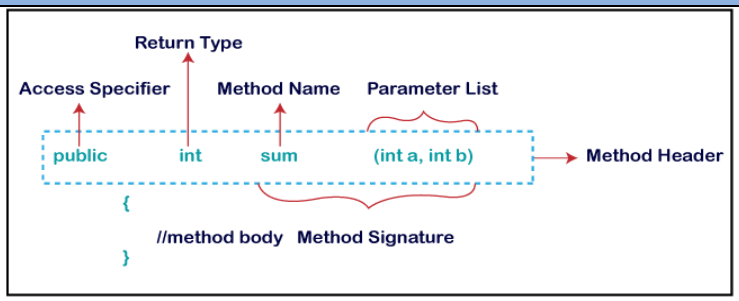
After performing this lab, students will be able to understand:

- Constructor and its types
- Method and its types
- Static methods
- This keyword
- Java Documentation basics

Method in class

***Syntax*:**

```
type name(parameter_list){
  //body of method
}
```



Method Overloading

Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different.

Example:

```
public class OverloadingExample {
    // Method with no parameters
    public void printMessage() {
        System.out.println("Hello!");
    }

    // Method with one parameter of type int
    public void printMessage(int n) {
        System.out.println("Number: " + n);
    }

    // Method with one parameter of type double
    public void printMessage(double d) {
        System.out.println("Double: " + d);
    }

    // Method with two parameters of different types
    public void printMessage(int num, double d) {
        System.out.println("Number: " + num + ", Double: " + d);
    }

    // Method with the same name but different type and number of parameters
    public void printMessage(String str) {
        System.out.println("String: " + str);
    }

    public static void main(String[] args) {
        OverloadingExample example = new OverloadingExample();
```

```
        // Calling methods with different parameter lists
        example.printMessage();
        example.printMessage(5);
        example.printMessage(3.14);
        example.printMessage(10, 3.14);
        example.printMessage("Overloading");
    }
}
```

Three ways to overload a method

In order to overload a method, the parameters of the methods must differ in either of these:

1. **Number of parameters.**
   For example: This is a valid case of overloading

add(int, int) add(int, int, int)

2. **Data type of parameters.**
   For example:

add(int, int) add(int, float)

3. **Sequence of Data type of parameters.**
   For example:

add(int, float) add(float, int)

Invalid case of method overloading:

If two methods have same name, same parameters and have different return type, then this is not a valid method overloading example. This will throw compilation error.

int add(int, int) float add(int, int)

Constructor

A constructor in Java is a special method that initializes an object when it is created.

- **Same name as its class** and syntactically similar to a method
- *Has **no explicit return type***. (not even void)
- It is **called** when an **object of class is created**.

***Syntax*:**

```
class ClassName {
   ClassName() {
   }
}//constructor call
```

Types of Constructors

There are three types of constructors in java

1. Default constructor
2. No-argument constructor(Parameter less).
3. Parameterized constructor
4. Copy constructor

Default constructor.

If you do not implement any constructor in your class, Java compiler inserts a default constructor into your code on your behalf.

If you implement any constructor then you no longer receive a default constructor from Java compiler.

No argument constructor

Constructor with no arguments is known as no-argument constructor. The signature is same as default constructor while creating an object.

Although you may see some people claim that that default and no-argument constructor is same but in fact they are not, even if you write public Demo() { } in your class Demo it cannot

be called default constructor since you have written the code of it.

```
class Demo
{
     public Demo()
     {
         System.out.println("This is a no argument constructor");
     }
     public static void main(String args[]) {
         new Demo();
     }
}
```

Parameterized constructor

Constructor with arguments (or you can say parameters) is known as Parameterized constructor.

```
public class Employee {

   int empId;
   String empName;

   //parameterized constructor with two parameters
   Employee(int id, String name){
       this.empId = id;
       this.empName = name;
   }
   void info(){
        System.out.println("Id: "+empId+" Name: "+empName);
   }

   public static void main(String args[]){
        Employee obj1 = new Employee(10245,"Ali");
        Employee obj2 = new Employee(92232,"saira");
        obj1.info();
        obj2.info();
   }
}
```

Copy Constructor

A copy constructor in a Java class is a constructor that creates an object using another object of the same Java class. This is specially helpful when we want to copy a complex object that has several fields .

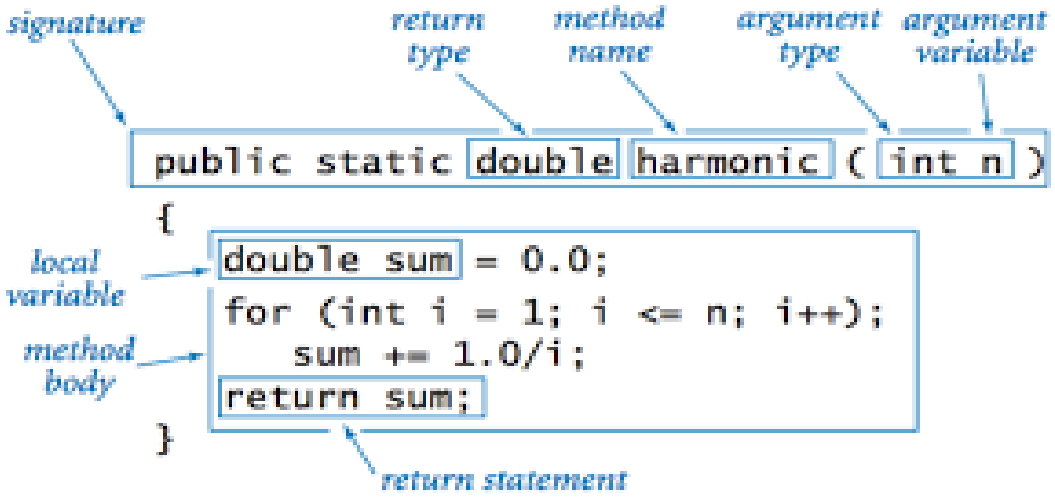
```
import java.util.Scanner;
public class Student {
```

```
    private String name;
    private int age;
    public Student(String name, int age){
        this.name = name;
        this.age = age;
    }

    public Student(Student std){
        this.name = std.name;
        this.age = std.age;
    }

    public void displayData(){
        System.out.println("Name : "+this.name);
        System.out.println("Age : "+this.age);
    }

    public static void main(String[] args) {
        Scanner sc =new Scanner(System.in);
        System.out.println("Enter your name ");
        String name = sc.next();
        System.out.println("Enter your age ");
        int age = sc.nextInt();
        Student std = new Student(name, age);
        System.out.println("Contents of the original object");
        std.displayData();
        System.out.println("Contents of the copied object");
        Student copyOfStd = new Student(std);
        copyOfStd.displayData();
    }
}
```

Static Methods

Methods that can be called without creating an object of class. They are referenced by the class name itself . Static method belongs to the class rather than the object of a class.

The following is its format:




```java
public class MathUtils {
    // Static method to calculate the factorial of a number
    public static int factorial(int n) {
        int result = 1;
        for (int i = 1; i <= n; i++) {
            result *= i;
        }
        return result;
    }

    public static void main(String[] args) {
        int num = 5;
        int result = MathUtils.factorial(num);
        System.out.println("Factorial of " + num + " is: " + result);
    }
}
```

Restrictions

a. The static method cannot use non-static data member or call non-static method directly.
b. this and super cannot be used in static context.

Why is the Java main method static?

Types of static Methods

Methods without Return Type and No Parameter

```
public static void method_name ( )
```

What we will do in this case for similar functionality code, create a function named drawLine() and call that function at repeated code locations.

```
class StaticMethods {
    // Static method with no parameters and no return type
    public static void greet() {
        System.out.println("Hello, World!");
    }

    public static void main(String[] args) {
        // Calling the static method without parameters and return type
        StaticMethods.greet();
    }
}
```

Methods without Return Type and with Parameters

```
public static void method_name ( par1, par2, par3 )
```

drawLine method with parameter to draw line according to the size provided in an argument.

```
class StaticMethods {
    // Static method with parameters but no return type
    public static void printSum(int num1, int num2) {
        int sum = num1 + num2;
        System.out.println("Sum: " + sum);
```

```
    }

    public static void main(String[] args) {
        // Calling the static method with parameters but no return type
        StaticMethods.printSum(5, 3);
    }
}
```

Methods With Return Type but no Parameter

```
public static return_type method_name ( )
```

```
public class StaticMethods {
    // Static method with return type but no parameters
    public static String getGreeting() {
        return "Hello, World!";
    }

    public static void main(String[] args) {
        // Calling the static method with return type but no parameters
        String greeting = StaticMethods.getGreeting();
        System.out.println(greeting);
    }
}
```

Methods With Return Type and Parameters

```
public static return_type method_name (par1, par2, par3)
```

```
public class StaticMethods {
    // Static method with parameters and return type
    public static int multiply(int num1, int num2) {
        return num1 * num2;
    }

    public static void main(String[] args) {
        // Calling the static method with parameters and return type
        int result = StaticMethods.multiply(5, 3);
        System.out.println("Result: " + result);
    }
}
```

There are two types of parameter passing:

1. Pass by Value

```
public static void sum(int a, int b)
```

- Call By value (copy of value) is when primitive data types are passed in the method call.

2. Pass by Reference (value of memory address location)

```
public static void displayCars(Car car)
```

- Objects and object variables are passed by reference or address.

7: The this Keyword

The this keyword refers to the current object in a method or constructor.

The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name (because a class attribute is shadowed by a method or constructor parameter).

this can also be used to:

- Invoke current class constructor
- Invoke current class method
- Return the current class object
- Pass an argument in the method call
- Pass an argument in the constructor call

Syntax:

```
public class Main{
  int x;
  // Constructor with a parameter
  public Main(int x) {
    this.x = x;
  }

  // Call the constructor
  public static void main(String[] args) {
    Main myObj = new Main(5);
    System.out.println("Value of x = " + myObj.x);
  }
}
```

Constructor of the current class can be accessed by using keyword this()

Java Documentation Comments

```
import java.io.*;
/**
* Add Two Numbers!
* The AddNum program implements an application that
* simply adds two given integer numbers and Prints
* the output on the screen.
* <p>
* <b>Note:</b> Giving proper comments in your program makes it more
* user friendly and it is assumed as a high quality code.
*
* @author  Zara Ali
* @version 1.0
* @since   2014-03-31
*/
public class Account {

   /**
   * This is the main method which makes use of addNum method.
   * @param args Unused.
   * @exception IOException On input error.
   * @see IOException
   */
   public static void main(String args[]) throws IOException {
      AddNum obj = new AddNum();
      int sum = obj.addNum(10, 20);

      System.out.println("Sum of 10 and 20 is :" + sum);
   }
}
```

```
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3\doc> javac DocExample.java
PS E:\Collaborative Learning\Spring_24_BS-II_OOP\Codes\lab3\doc> javadoc DocExample.java
Loading source file DocExample.java...
Constructing Javadoc information...
Building index for all the packages and classes...
Standard Doclet version 21.0.2+13-LTS-58
Building tree for all the packages and classes...
Generating .\DocExample.html...
```

The javadoc tool recognizes the following tags:

https://www.tutorialspoint.com/java/java_documentation.htm

https://www.javatpoint.com/java-comments

Exercises

Mega Exercise: Solve Exercises from the HackerRank (OOP track topics). As many as you can. I have listed here two basic challenges)

https://www.hackerrank.com/domains/java

1. Exercise 1(a) Car.java

Implement a class Car, that has the following characteristics:

a) brandName,
b) priceNew, which represents the price of the car when it was new, c) color
d) odometer, which is milo meter shows number of milage travelled by car

The class should have:

A. A method getPriceAfterUse() which should return the price of the car after being used according to the following formula:

$$car\ price\ after\ being\ used = priceNew * (1 - \frac{odometer}{600000})$$

B. A method updateMileage(double traveledDistance) that changes the current state of the car by increasing its milage,

C. A method outputDetails() that will output to the screen all the information of the car, i.e., brand name, price new, price used, color, and odometer.

Exercise 1(b) TestCar.java

Write a test class for the Car class above. You are required to do the followings:

a. Create an object of type Car.
b. Assign any valid values to the instance variables of the object created in 'A'.
c. Use the method getPriceAfterUse on the object created in 'A' then output the result to the screen.
d. Use the method updateMilage on the object created in 'A' by passing a valid value.
e. Do part 'C' again.
f. Use the method outputDetails on the object created in 'A'.

2. Exercise 2(a) Ship.java

MyJava Coffee Outlet runs a catalog business. It sells only one type of coffee beans. The company sells the coffee in 2-lb bags only and the price of a single 2-lb bag is $5.50. When a customer places an order, the company ships the order in boxes. The boxes come in 3 sizes with 3 different costs:

	Large box	Medium box	Small box
capacity	20 bags	10 bags	5 bags
cost	$1.80	$1.00	$0.60

The order is shipped using the least number boxes. For example, the order of 52 bags will be shipped in 2 boxes: 2 large boxes, 1medium and 1 small.

Exercise 2(b) ShipTest.java

Develop an application that computes the total cost of an order. **Sample output:**

```
Number of Bags Ordered: 52
The Cost of Order: $ 286.00

Boxes Used:
        2 Large  - $3.60
        1 Medium - $1.00
        1 Small  - $0.60

Your total cost is: $ 291.20
```

3. Exercise 3 Date.java

Create your version of **`Date`** class that includes three instance variables—a **`month`** (type int), a **`day`** (type int) and a **`year`** (type int).

Provide a constructor that initializes the three instance variables and assumes that the values provided are correct. Write a test app named DateTest that demonstrates class Date's capabilities.

4. Exercise, EmployeeInfor.java

Write a program by creating an 'EmployeeInfo' class having the following methods and print the final salary.

- **'collectInfo()'** which takes the salary, number of hours of work per day of employee as parameter and sets the values of paramters
- **'AddWork()'** which adds $5 to salary of employee if the number of hours of work per day is more than 6 hours.
- **'printFinalSalary()'** That displays the final salary of the employee

Code snippet to call the methods in the main()

```
        EmployeeInfo employee = new EmployeeInfo();

        // Collect employee information
        employee.collectInfo(1000, 8); // Assuming $1000 salary and 8 hours of
work
        // Add work bonus if applicable
        employee.AddWork();

        // Print final salary
        employee.printFinalSalary();
```

Output

```
PS E:\Collab-Learning\Spring24_BS-II_OOP\Codes\lab5> java EmployeeInfo
Final Salary: $1005.0
```

5. Exercise: Static Method: (LastDigit.java):

Write a static method named findLastDigit that returns the last digit of an integer. For example, findLastDigit(3852) should return 2

6. Exercise: ArrayUtils.java

Create a class **ArrayUtils** that contains a static method for finding the maximum element in an array of integers.

Method: findMaxElement

Signature: public static int findMaxElement(int[] arr)

Purpose: Returns the maximum element in the given array of integers.

7. Exercise (FactorialDetector.java)

Write a program that takes input from the user. It tells the factorial number of the input.

Output

```
Enter any input: 24

24 is the factorial of: 4

Enter any input: 6

6 is the factorial of: 3

Enter any input: 10

10 has no any factorial!
```

8. Exercise (Method Overloading)

Write a program by creating class MethodOverloading that contains following methods

- Computations (int length, int width). It takes length and width and displays Area of Rectangle.
- Computations (String name, int age) it takes name, and age. Display name, and age.
- Computations (double mass, double acceleration) It takes mass and acceleration and displays force.

Code snippet example to call/invoke these methods in the main() function

```
MethodOverloading overloading = new MethodOverloading();

// Call the methods with different parameters
overloading.Computations(5, 3); // Area of Rectangle
overloading.Computations("John", 25); // Name and Age
overloading.Computations(10.5, 9.8); // Force
```