02A Lesson Proper for Week 8

What are the Building blocks of object-oriented programming -OOP- ?

Java is an object-oriented programming (OOP) language and adheres to few main concepts of OOP. In this module, we will check out all the Object Oriented Concepts in Java explained with code and examples.

What is Object-Oriented Programming and Concepts?

Object-Oriented Programming is a method of programming where programmers define the type of data as well the operations that the data can perform.

The core of any programming language is to have method. In C, a method is called function and in Common Business Oriented Language (COBOL) it is a subroutine. All of them are collection of statements that are grouped together to perform an operation or specific task. When you call the System.out.println() method, for example, the system actually executes several statements in order to display a message on the console.

Creating Method

Considering the following example to explain the syntax of a method −

Syntax

```
public static int methodName(int a, int b) {
    // body
}
Here,
```

- **public static** − modifier
- **int** − return type
- **methodName** − name of the method
- **a, b** − formal parameters
- **int a, int b** − list of parameters

Method definition consists of a method header and a method body. The same is shown in the following syntax −

Syntax

```
modifier returnType nameOfMethod (Parameter List) {
    // method body
```

}

The syntax shown above includes −

- **modifier** − It defines the access type of the method and it is optional to use.
- **returnType** − Method may return a value.
- **nameOfMethod** − This is the method name. The method signature consists of the method name and the parameter list.
- **Parameter List** − The list of parameters, it is the type, order, and number of parameters of a method. These are optional, method may contain zero parameters.
- **method body** − The method body defines what the method does with the statements.

Example

The method called **minFunction()**. This method takes two parameters num1 and num2 and returns the maximum between the two −

```
/** the snippet returns the minimum between two numbers */

public static int minFunction(int n1, int n2) {
    int min;
    if (n1 > n2)
        min = n2;
    else
        min = n1;

    return min;
}
```

Method Calling

For using a method, it should be called. There are two ways in which a method is called i.e., method returns a value or returning nothing (no return value).

The process of method calling is simple. When a program invokes a method, the program control gets transferred to the called method. This called method then returns control to the caller in two conditions, when −

- the return statement is executed.
- it reaches the method ending closing brace.

The methods returning void is considered as call to a statement. Lets consider an example −

```
System.out.println("This is tutorialspoint.com!");
```

The method returning value can be understood by the following example −

```
int result = sum(6, 9);
```

To demonstrate how to define a method and how to call it −

Example

```
public class ExampleMinNumber {

   public static void main(String[] args) {
      int a = 11;
      int b = 6;
      int c = minFunction(a, b);
      System.out.println("Minimum Value = " + c);
   }

   /** returns the minimum of two numbers */
   public static int minFunction(int n1, int n2) {
      int min;
      if (n1 > n2)
         min = n2;
      else
         min = n1;

      return min;
   }
}
```

This will produce the following result − \

Output

```
Minimum value = 6
```

The void Keyword

The void keyword allows us to create methods which do not return a value. Here, in the following example we're considering a void method methodRankPoints. This method is a void method, which does not return any value. Call to a void method must be a statement i.e. methodRankPoints(255.7);. It is a Java statement which ends with a semicolon as shown in the following example.

Example

```
public class ExampleVoid {

   public static void main(String[] args) {
      methodRankPoints(255.7);
   }

   public static void methodRankPoints(double points) {
      if (points >= 202.5) {
         System.out.println("Rank:A1");
      }else if (points >= 122.4) {
         System.out.println("Rank:A2");
      }else {
         System.out.println("Rank:A3");
      }
   }
}
```

This will produce the following result −

Output

```
Rank:A1
```

Passing Parameters by Value

Passing Parameters by Value means calling a method with a parameter. Through this, the argument value is [passed] to the parameter.

Example

The following program shows an example of passing parameter by value. The values of the arguments remains the same even after the method invocation.

```java
public class swappingExample {

  public static void main(String[] args) {
    int a = 30;
    int b = 45;
    System.out.println("Before swapping, a = " + a + " and b = " + b);

    // Invoke the swap method
    swapFunction(a, b);
    System.out.println("\n**Now, Before and After swapping values will be same here**:");
    System.out.println("After swapping, a = " + a + " and b is " + b);
  }

  public static void swapFunction(int a, int b) {
    System.out.println("Before swapping(Inside), a = " + a + " b = " + b);

    // Swap n1 with n2

    int c = a;
    a = b;
    b = c;
    System.out.println("After swapping(Inside), a = " + a + " b = " + b);
}}
```

This will produce the following result −

Output

```
Before swapping, a = 30 and b = 45
Before swapping(Inside), a = 30 b = 45
After swapping(Inside), a = 45 b = 30

**Now, Before and After swapping values will be same here**:
After swapping, a = 30 and b is 45
```

Method Overloading

When a class has two or more methods by the same name but different parameters, it is known as method overloading. It is different from overriding. In overriding, a method has the same method name, type, number of parameters, etc.

Let's consider the example discussed earlier for finding minimum numbers of integer type. If, let's say we want to find the minimum number of double type. Then the concept of overloading will be introduced to create two or more methods with the same name but different parameters.

The following example explains the same −

Example

```
public class ExampleOverloading {

   public static void main(String[] args) {
      int a = 11;
      int b = 6;
      double c = 7.3;
      double d = 9.4;
      int result1 = minFunction(a, b);

      // same function name with different parameters
      double result2 = minFunction(c, d);
      System.out.println("Minimum Value = " + result1);
      System.out.println("Minimum Value = " + result2);
   }

   // for integer
   public static int minFunction(int n1, int n2) {
      int min;
      if (n1 > n2)
```

```
      min = n2;
      else
         min = n1;

      return min;
   }
   // for double
   public static double minFunction(double n1, double n2) {
      double min;
      if (n1 > n2)
         min = n2;
      else
         min = n1;

      return min;
   }}
```

This will produce the following result −

Output

```
Minimum Value = 6
Minimum Value = 7.3
```

Overloading methods makes program readable. Here, two methods are given by the same name but with different parameters. The minimum number from integer and double types is the result.

Classes and Objects.

- **Object** – Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.
- **Class** – A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.

Classes in Java

A class is a blueprint from which individual objects are created.

Following is a sample of a class.

Example

```
public class Dog {
   String breed;
```

```
int age;
String color;

void bark () {
}

void hungry() {
}

void sleep () {
}
```

A class can contain any of the following variable types.

- **Local variables** − Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
- **Instance variables** − Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
- **Class variables** − Class variables are variables declared within a class, outside any method, with the static keyword.

A class can have any number of methods to access the value of various kinds of methods. In the above example, bark(), hungry() and sleep() are methods.

Constructors

When discussing about classes, one of the most important sub topic would be constructors. Every class has a constructor. If we do not explicitly write a constructor for a class, the Java compiler builds a default constructor for that class.

Each time a new object is created, at least one constructor will be invoked. The main rule of constructors is that they should have the same name as the class. A class can have more than one constructor.

Following is an example of a constructor −

Example

```
public class Puppy {
    public Puppy() {
    }
    public Puppy(String name) {
        // This constructor has one parameter, name.
    }
}
```

Creating an Object

As mentioned previously, a class provides the blueprints for objects. So basically, an object is created from a class. In Java, the **new** keyword is used to create new objects.

There are three steps when creating an object from a class −

- **Declaration** − A variable declaration with a variable name with an object type.
- **Instantiation** − The 'new' keyword is used to create the object.
- **Initialization** − The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

Following is an example of creating an object –

Example

```
public class Puppy {
    public Puppy(String name) {
        // This constructor has one parameter, name.
        System.out.println("Passed Name is :" + name );
    }
    public static void main(String []args) {
        // Following statement would create an object myPuppy
        Puppy myPuppy = new Puppy( "tommy" );
    }
}
```

If we compile and run the above program, then it will produce the following result –

Output
Passed Name is :tommy

EXERCISE 1

A student is to create a simple Java program about student enrolment that will classify regular and irregular students. The student type is either 'R' for regular and 'I' for Irregular. What could be the name of the method besides the main() method in the program?

_____

_____

_____

EXERCISE 2

Your school would like to create simple registration system to help students enrolled in the college. What are the variables to be used in the program?

_____

_____

_____

Navigation

ℹ️ Fair Warning

NOTICE: Please be reminded that it has come to the attention of the Publishing Team of eLearning Commons that learning materials published and intended for *free use only by students and faculty members within the eLearning Commons network were UNLAWFULLY uploaded in other sites without due and proper permission*.

PROSECUTION: Under Philippine law (Republic Act No. 8293), copyright infringement is punishable by the following: Imprisonment of between 1 to 3 years and a fine of between 50,000 to 150,000 pesos for the first offense. Imprisonment of 3 years and 1 day to six years plus a fine of between 150,000 to 500,000 pesos for the second offense.
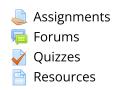
COURSE OF ACTION: Whoever has maliciously uploaded these concerned materials are hereby given an ultimatum to take it down within 24-hours. Beyond the 24-hour grace period, our Legal Department shall initiate the proceedings in coordination with the National Bureau of Investigation for IP Address tracking, account owner identification, and filing of cases for prosecution.

ℹ️ 2nd Semester Enrollment

🧩 Activities

- 📄 Assignments
- 💬 Forums
- ✔️ Quizzes
- 📄 Resources

Bestlink College of the Philippines
College Department

Powered byeLearning Commons