Object Oriented Programming with Java (Subject Code: BCS-403)

Unit 1

Lecture 6

Lecture 6

- Class
- Object
- Constructors
- Methods

Class in Java

- Class is a template or blueprint from which objects are created.

A class in java can contain:

- **data member**
- **method**
- **constructor**
- **block**
- **class and interface**

Syntax to declare a class:

```
class <class_name>
{
   data member;
   method;
}
```

Object in Java

- **Object is an instance of a class.** Class is a template or blueprint from which objects are created. So object is the instance(result) of a class.

An object has three characteristics:

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

Constructor in Java

- **Constructor in java** is a *special type of method* that is used to initialize the object.
- Java constructor is *invoked at the time of object creation*.
- It constructs the values i.e. provides data for the object that is why it is known as constructor.

Rules for creating java constructor

There are basically two rules defined for the constructor.

- Constructor name must be same as its class name
- Constructor must have no explicit return type

Types of java constructors

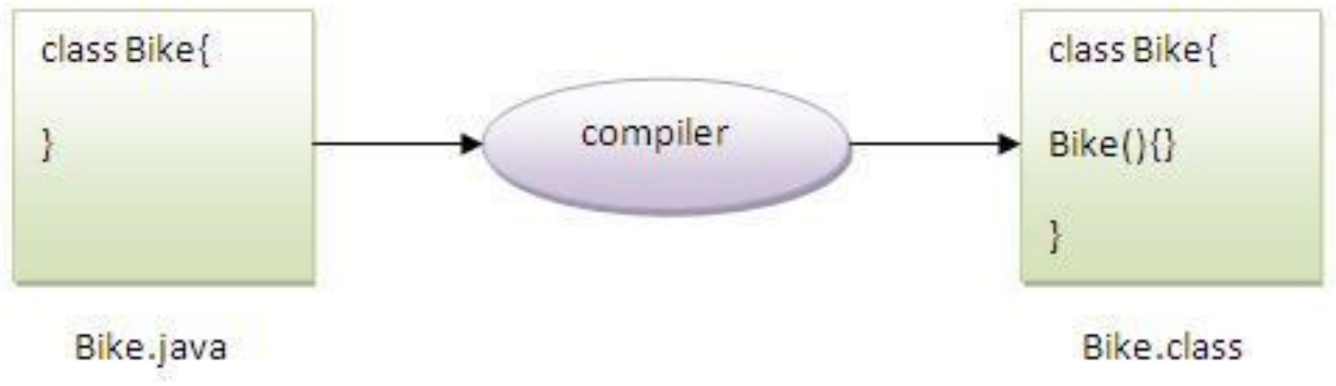
There are two types of constructors:

- Default constructor (no-arg constructor)
- Parameterized constructor

Example of default constructor

- In this example, we are creating the no-arg constructor in the Bike class. It will be invoked at the time of object creation.

```
class Bike1{
Bike1(){System.out.println("Bike is created");}
public static void main(String args[]){
Bike1 b=new Bike1();
}}
```



```
class Bike{

}
```

```
class Bike{

Bike(){}

}
```

Rule: If there is no constructor in a class, compiler automatically creates a default constructor.

Example of parameterized constructor

```
class Student4{
    int id;
    String name;

    Student4(int i,String n){
    id = i;
    name = n;
    }
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
    Student4 s1 = new Student4(111,"Karan");
    Student4 s2 = new Student4(222,"Aryan");
    s1.display();
    s2.display();
   }
}
```

Constructor Overloading in Java

- Constructor overloading is a technique in Java in which a class can have any number of constructors that differ in parameter lists.
- The compiler differentiates these constructors by taking into account the number of parameters in the list and their type.

```java
class Student5{
    int id;
    String name;
    int age;
    Student5(int i,String n){
    id = i;
    name = n;
    }
    Student5(int i,String n,int a){
    id = i;
    name = n;
    age=a;
    }
    void display(){System.out.println(id+" "+name+" "+age);}

    public static void main(String args[]){
    Student5 s1 = new Student5(111,"Karan");
    Student5 s2 = new Student5(222,"Aryan",25);
    s1.display();
    s2.display();
   }
}
```

Java Copy Constructor

- There is no copy constructor in java. But, we can copy the values of one object to another like copy constructor in C++.
- There are many ways to copy the values of one object into another in java. They are:
- By constructor
- By assigning the values of one object into another

```
class Student6{
    int id;
    String name;
    Student6(int i,String n){
    id = i;
    name = n;
    }
    Student6(Student6 s){
    id = s.id;
    name =s.name;
    }
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
    Student6 s1 = new Student6(111,"Karan");
    Student6 s2 = new Student6(s1);
    s1.display();
    s2.display();
   }
}
```

Methods in Java

Methods of Java is a collection of statements that perform some specific task and return the result to the caller.

1. A method is like a function i.e. used to expose the behavior of an object.
2. It is a set of codes that perform a particular task.

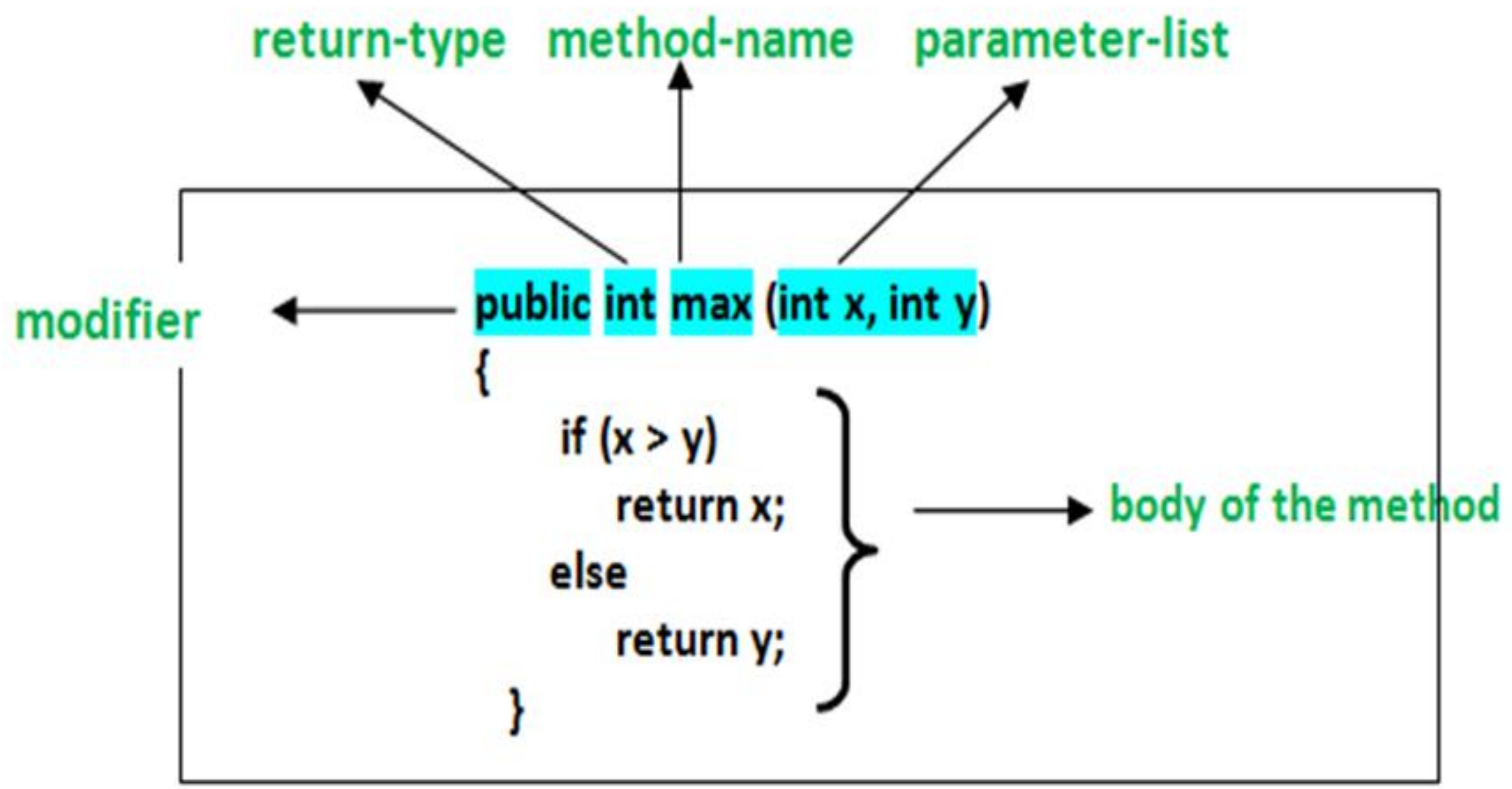
Syntax of Method

```
<access_modifier> <return_type> <method_name>(
list_of_parameters)
{
   //body
}
```

Advantage of Method

- Code Reusability
- Code Optimization

Method Declaration



```
public int max (int x, int y)
{
     if (x > y)
          return x;
     else
          return y;
 }
```

Method Declaration

In general, method declarations have 6 components:

1.Modifier: It defines the access type of the method i.e. from where it can be accessed in your application. In Java, there 4 types of access specifiers.

- public: It is accessible in all classes in your application.
- protected: It is accessible within the class in which it is defined and in its subclass/es
- private: It is accessible only within the class in which it is defined.
- default: It is declared/defined without using any modifier. It is accessible within the same class and package within which its class is defined.

Note: It is Optional in syntax.

2. The return type: The data type of the value returned by the method or void if does not return a value. It is Mandatory in syntax.

3. Method Name: the rules for field names apply to method names as well, but the convention is a little different. It is Mandatory in syntax.

4. Parameter list: Comma-separated list of the input parameters is defined, preceded by their data type, within the enclosed parenthesis. If there are no parameters, you must use empty parentheses (). It is Optional in syntax.

5. Exception list: The exceptions you expect by the method can throw, you can specify these exception(s). It is Optional in syntax.

6. Method body: it is enclosed between braces. The code you need to be executed to perform your intended operations. It is Optional in syntax.

Example

```
class Addition {
    // Initially taking sum as 0
    // as we have not started computation
    int sum = 0;
      // Method
    // To add two numbers
    public int addTwoInt(int a, int b)
    {
          // Adding two integer value
        sum = a + b;

        // Returning summation of two values
        return sum;
    }
}
```