05-Methods

Methods in Java

Definition: Methods are functions or behaviours within a class that perform specific tasks. A method in Java is a collection of statements that execute certain operations and can return a result to the caller. However, not all methods must return a value. Java methods allow for code reuse, reducing the need to rewrite the same code multiple times. Unlike some other programming languages like C, C++, and Python, in Java, every method must belong to a class.

A method can be compared to a function in that it exposes the behaviour of an object. It consists of a set of code that performs a particular task. For example, the main method is the starting point of a Java program, where the execution begins.

Syntax of a Method:

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

Advantages of Using Methods:

- **Code Reusability:** Methods allow the reuse of code, eliminating the need to write the same code multiple times.
- **Code Optimization:** By using methods, the code becomes more organized and easier to manage.

Example of Methods in Real Life

Consider the process of building a product. The product is made up of various components, and these components are created using modules. In Java, a class represents a component, and the functions or behaviours of that component are represented by methods.

Example:
If we consider a car as the final product, its components are the wheels, engine, dashboard, etc. Each of these components corresponds to a class, and the behaviour or functions of each component (like the rotation of wheels or the ignition of the engine) are represented by methods within those classes.

Types of Methods in Java

There are two primary types of methods in Java:

1. **Predefined Methods:**

- These methods are already defined in the Java class libraries and can be used directly in your programs. They are also known as standard library methods or built-in methods.
- **Example:** System.out.println() is a predefined method in Java.

2. **User-Defined Methods:**
   - These are methods created by the programmer to perform specific tasks as per the requirements.
   - **Example:**

```java
public void playMusic() {
    System.out.println("Playing music");
}

public String giveMePen(int cost) {
    if (cost >= 10)
        return "Pen";
    return "Nothing";
}
```

   - **Usage:**

```java
public static void main(String[] args) {
    Computer obj = new Computer();
    obj.playMusic();
    String str = obj.giveMePen(10);
    System.out.println(str);
}
```

Static Method

A method that has static keyword is known as static method. In other words, a method that belongs to a class rather than an instance of a class is known as a static method. We can also create a static method by using the keyword **static** before the method name.

The main advantage of a static method is that we can call it without creating an object. It can access static data members and also change the value of it. It is used to create an instance method. It is invoked by using the class name. The best example of a static method is the **main()** method.

Example:

```
public class Display
{
public static void main(String[] args)
{
show();
}
static void show()
{
System.out.println("It is an example of static method.");
}
}
```

Instance Method

The method of the class is known as an **instance method**. It is a **non-static** method defined in the class. Before calling or invoking the instance method, it is necessary to create an object of its class. Let's see an example of an instance method.

Example:

```
public class Calculator
{
public static void main(String [] args)
{
//Creating an object of the class
Calculator   obj = new Calculator   ();
//invoking instance method
System.out.println("The sum is: "+obj.add(12, 13));
}
int s;
//user-
defined method because we have not used static keyword
public int add(int a, int b)
{
s = a+b;
//returning the sum
return s;
}
}
```

Abstract Method

The method that does not has method body is known as abstract method. In other words, without an implementation is known as abstract method. It always declares in the **abstract class**. It means the class itself must be abstract if it has abstract method. To create an abstract method, we use the keyword **abstract**.

TELUSKO

Example:

```
abstract class Demo //abstract class
{
//abstract method declaration
abstract void display();
}
public class MyClass extends Demo
{
//method impelmentation
void display()
{
System.out.println("Abstract method?");
}
public static void main(String args[])
{
//creating object of abstract class
Demo obj = new MyClass();
//invoking abstract method
obj.display();
}
}
```

Final Method:

The **final** is a keyword in java. The final keyword is used to denote the constants.
The final keyword is usually used with methods, classes, and variables. Whenever we define any variable or class, or method as final, then the particular entity cannot be changed after it has been assigned the final method in java is the method that cannot be changed or overridden once defined. Hence, we cannot modify a final method from the subclass.

Example:

```
class Calculator {
    // defining a final method inside the parent class.
    public final void display() {
        System.out.println("This is a final method.");
    }
}

public class Demo extends Calculator {
    // overriding the final method of the parent class.
    public final void display() {
        System.out.println("Overriding the final method.");
    }

    public static void main(String[] args) {
        // creating an object of the child class (Test)
        Demo obj = new Demo ();
        // calling the final method from the child class object.
        obj.display();
    }
}
```

Output:

```
Demo.java:8: error: display() in Demo cannot override
display() in Calculator
    public final void display() {
                      ^
  The overridden method is final
1 error
```

FAQs on Methods in Java

Q1. What is a method in Java programming?
A method in Java is a collection of statements that perform a specific task and return the result to the caller.

Q2. What are the parts of a method in Java?
The main components of a method in Java include:

- **Modifiers:** Keywords that define the access level and other properties of the method.
- **Return Type:** The data type of the value the method returns (e.g., void if it returns nothing).
- **Method Name:** The name of the method used to call it.
- **Parameters:** A list of input values the method accepts.
- **Method Body:** The block of code that defines what the method does.