Static methods vs Instance methods in Java

Difficulty Level : Medium Last Updated : 28 Jun, 2021

Instance Method

Instance method are methods which require an object of its class to be created before it can be called. To invoke a instance method, we have to create an Object of the class in within which it defined.

```
public void geek(String name)
{
 // code to be executed....
}
// Return type can be int, float String or user defined data type.
```

Memory allocation: These methods themselves are stored in Permanent Generation space of heap but the parameters (arguments passed to them) and their local variables and the value to be returned are allocated in stack. They can be called within the same class in which they reside or from the different classes defined either in the same package or other packages depend on the **access type** provided to the desired instance method.
Important Points:

- Instance method(s) belong to the Object of the class, not to the class i.e. they can be called after creating the Object of the class.
- Instance methods are not stored on a per-instance basis, even with virtual methods. They're stored in a single memory location, and they only "know" which object they belong to because this pointer is passed when you call them.
- They can be overridden since they are resolved using **dynamic binding** at run time.

Below is the implementation of accessing the instance method:

```
// Example to illustrate accessing the instance method .
import java.io.*;
```

```
class Foo {

    String name = "";

    // Instance method to be called within the
    // same class or from a another class defined
    // in the same package or in different package.
    public void geek(String name) { this.name = name; }
}

class GFG {
    public static void main(String[] args)
    {

        // create an instance of the class.
        Foo ob = new Foo();

        // calling an instance method in the class 'Foo'.
        ob.geek("GeeksforGeeks");
        System.out.println(ob.name);
    }
}
```

Output

```
GeeksforGeeks
```

Static Method

Static methods are the methods in Java that can be called without creating an object of class. They are referenced by the **class name itself** or reference to the Object of that class.

```
public static void geek(String name)
{
 // code to be executed....
}

// Must have static modifier in their declaration.
// Return type can be int, float, String or user defined data type.
```

Memory Allocation:

They are stored in the Permanent Generation space of heap as they are associated with the class in which they reside not to the objects of that class. But their local variables and the passed argument(s) to them are stored in the stack. Since they belong to the class, so they can be called to without creating the object of the class.

Important Points:

- Static method(s) are associated with the class in which they reside i.e. they are called without creating an instance of the class i.e **ClassName.methodName(args)**.
- They are designed with the aim to be shared among all objects created from the same class.
- Static methods can not be overridden, since they are resolved using **static binding** by the compiler at compile time. However, we can have the same name methods declared **static** in both **superclass** and **subclass**, but it will be called **Method Hiding** as the derived class method will hide the base class method.

Below is the illustration of accessing the static methods:

```java
// Example to illustrate Accessing
// the Static method(s) of the class.
import java.io.*;

class Geek {

    public static String geekName = "";

    public static void geek(String name)
    {
        geekName = name;
    }
}

class GFG {
    public static void main(String[] args)
    {

        // Accessing the static method geek()
        // and field by class name itself.
        Geek.geek("vaibhav");
        System.out.println(Geek.geekName);

        // Accessing the static method geek()
        // by using Object's reference.
        Geek obj = new Geek();
```

```
            obj.geek("mohit");
            System.out.println(obj.geekName);
        }
    }
```

Output

```
vaibhav
mohit
```

Note: Static variables and their values (primitives or references) defined in the class are stored in **PermGen** space of memory.

What if static variable refers to an Object?

```
static int i = 1;
static Object obj = new Object();
```

In the first line, the value 1 would be stored in PermGen section. In second line, the reference obj would be stored in PermGen section and the Object it refers to would be stored in heap section.

When to use static methods?

- When you have code that can be shared across all instances of the same class, put that portion of code into static method.
- They are basically used to access static field(s) of the class.

Instance method vs Static method

- Instance method can access the instance methods and instance variables directly.
- Instance method can access static variables and static methods directly.
- Static methods can access the static variables and static methods directly.
- Static methods can't access instance methods and instance variables directly. They must use reference to object. And static method can't use this keyword as there is no instance for 'this' to refer to.

References

- https://docs.oracle.com/javase/tutorial/java/javaOO/classvars.html

- Stackoverflow

This article is contributed by **Nitsdheerendra**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.
Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.