## **Java Instance Methods**

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

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 depending on the **access type** provided to the desired instance method.

* 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.
* They can be overridden

## **Java Static Methods**

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.

* 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

### **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 a static method.*
* *They are basically used to access static field(s) of the class.*

### **Difference Between Instance method vs Static method**

* *Instance methods can access the instance methods and instance variables directly.*
* *Instance methods 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 references to objects. And static methods can’t use* ***this*** *keyword as there is no instance for ‘this’ to refer to.*

| **Variable Type** | **Location** |
| --- | --- |
| Static variable | Metaspace |
| Instance variable | Heap |

* Metaspace:
* The Metaspace is a shared memory region that stores class metadata, static variables, and method code. It is created when the JVM starts up and is not garbage collected.
* Heap:
* The Heap is a shared memory region that stores objects and their instance variables. It is garbage collected, which means that the JVM automatically reclaims memory that is no longer being used by objects.

