Constructor

- A constructor initializes the instance variables of an object.
- It is called immediately after the object is created but before the new operator completes.

  1) it is syntactically similar to a method:

  2) it has the same name as the name of its class

  3) it is written without return type; the default return type of a class constructor is the same class
- When the class has no constructor, the default constructor automatically initializes all its instance variables with zero.

Example: Constructor

- class Box {

```
    double width;
    double height;
    double depth;
    Box() {
        System.out.println("Constructing Box");
        width = 10; height = 10; depth = 10;
        }
    double volume() {
        return width * height * depth;
        }
    }
```

Example: Constructor

```
class BoxDemo6 {
    public static void main(String args[]) {
    Box mybox1 = new Box();
    Box mybox2 = new Box();
    double vol;
    vol = mybox1.volume();
    System.out.println("Volume is " + vol);
    vol = mybox2.volume();
    System.out.println("Volume is " + vol);
    }
}
```

Parameterized Constructor

- So far, all boxes have the same dimensions.
- We need a constructor able to create boxes with different dimensions:

```
class Box {
      double width;
      double height;
      double depth;
      Box (double w, double h, double d) {
            width = w; height = h; depth = d;
            }
double volume() {
            return width * height * depth; }
}
```

Parameterized Constructor

```
• class BoxDemo7 {
      public static void main(String args[]) {
      Box mybox1 = new Box(10, 20, 15);
      Box mybox2 = new Box(3, 6, 9);
      double vol;
      vol = mybox1.volume();
      System.out.println("Volume is " + vol);
      vol = mybox2.volume();
      System.out.println("Volume is " + vol);
      }
  }
```

Difference between Constructor and Method

1. The purpose of constructor is to create object of a class while the purpose of a method is to perform a task by executing java code.
2. Constructors cannot be abstract, final, static and synchronised while methods can be.
3. Constructors do not have return types while methods do.

Static Members

- Static keyword can be used with class, variable, method and block.
- Variables and methods declared using keyword static are called static members of a class.
- The non-static variables and methods belong to instance.
- The static members (variables, methods) belong to class. Static members are not part of any instance of the class.
- Static members can be accessed using class name directly, in other words, there is no need to create instance of the class specifically to use them.

Static Variables

- Static variables are also called class variables because they can be accessed using class name.
- Static variable in Java is variable which belongs to the class and initialized only once at the start of the execution. It is a variable which belongs to the class and not to object(instance).
- Non static variables are called instance variables and can be accessed using instance reference only.
- Static variables occupy single location in the memory. These can also be accessed using instance reference.
- Static variables are initialized only once, at the start of the execution. These variables will be initialized first, before the initialization of any instance variables.

Static Variables

```
class Test {
   // static variable
   static int max = 10;
     // non-static variable
   int min = 5;
}
public class StaticVar {
   public static void main(String[] args) {
       Test obj = new Test();
       // access the non-static variable
       System.out.println("min + 1 = " + (obj.min + 1));
       // access the static variable
       System.out.println("max + 1 = " + (Test.max + 1));
   }
}
```

Static Methods/Class Methods

- Static methods are also called class methods. A static method belongs to class, it can be used with class name directly. It is also accessible using instance references.
- Static methods can use static variables only, whereas non-static methods can use both instance variables and static variables.
- A static method can call only other static methods and can not call a non-static method from it.
- A static method can be accessed directly by the class name and doesn’t need any object
- A static method cannot refer to "this" or "super" keywords in anyway.
- Note: main method is static, since it must be accessible for an application to run, before any instantiation takes place.

Static Methods/Class Methods

```
class StaticTest {
    // non-static method
    int multiply(int a, int b){
        return a * b;
    }
    // static method
         static int add(int a, int b){
        return a + b;
    }
 }
      public class Main {
          public static void main( String[] args ) {
         StaticTest st = new StaticTest();
        // call the nonstatic method
        System.out.println(" 2 * 2 = " + st.multiply(2,2));
        // call the static method
        System.out.println(" 2 + 3 = " + StaticTest.add(2,3));
   }
}
```

Static Block / static clause

- Static block is used for initializing the static variables.
- This code inside static block is executed only once: the first time the class is loaded into memory.
- A static block in a program is a set of statements which are executed by the JVM (Java Virtual Machine) before the main method.
- At the time of class loading, if we want to perform any task we can define that task inside the static block, this task will be executed at the time of class loading.
- In a class, any number of a static block can be defined, and this static blocks will be executed from top to bottom.

Static Block / static clause

$$\begin{tikzcd}\boxed{\text{Main( )}} \arrow[r, "\text{Static block will execute before the main() method}"] & \boxed{\text{static\{ \}}}\end{tikzcd}$$

Example:

```
class StaticDemo1{
    static{
             System.out.println("This is static block");
            }
    public static void main(String as[]){
             System.out.println("This is main() method");
           }
}Out put : This is static block
               This is main() method
```

finalize() Method

- A constructor helps to initialize an object just after it has been created.
- In contrast, the finalize method is invoked just before the object is destroyed:

  1) implemented inside a class as:

  ```
  protected void finalize() { … }
  ```

  2) implemented when the usual way of removing objects from memory is insufficient, and some special actions has to be carried out
- How is the finalize method invoked?

finalize() Method

- To add a finalizer to a class, you simply define the finalize( ) method.
- The Java run time calls that method whenever it is about to recycle an object of that class.
- Inside the finalize( ) method you will specify those actions that must be performed before an object is destroyed.
- The garbage collector runs periodically, checking for objects that are no longer referenced by any running state or indirectly through other referenced objects.
- Right before an asset is freed, the Java run time calls the finalize( ) method on the object

Garbage Collection

- Garbage collection is a mechanism to remove objects from memory when they are no longer needed.
- Garbage collection is carried out by the garbage collector:

1. The garbage collector keeps track of how many references an object has.
2. It removes an object from memory when it has no longer any references.
3. Thereafter, the memory occupied by the object can be allocated again.
4. The garbage collector invokes the finalize method.

Keyword this

- Keyword this allows a method to refer to the object that invoked it.
- It can be used inside any method to refer to the current object:

-
  ```
  Box(double width, double height, double depth) {
        this.width = width;
        this.height = height;
        this.depth = depth;
  }
  ```
- The above use of this is redundant but correct.
- When is this really needed?

Instance Variable Hiding

- Variables with the same names:
  1. it is illegal to declare two local variables with the same name inside the same or enclosing scopes
  2. it is legal to declare local variables or parameters with the the same name as the instance variables of the class.
- As the same-named local variables/parameters will hide the instance variables, using this is necessary to regain access to them:

```
Box(double width, double height, double depth) {
        this.width = width;
        this.height = height;
        this.depth = depth;
}
```