**Class/Object/Method/Constructor/Method\_Overloading/Overriding**

**Q. What is Class?**

**A class is a blueprint or template from which individual objects are created.**

* Suppose Teacher Class....there may be teacher1, teacher2.......object
* Suppose fruit Class----Banana, Apple, Orange are objects.
* We created the Teacher Class for using it multiple times.

**OR**

* **Class is the final building block of Java where we have to define the properties.**
* Properties means number of variables and methods.
* What is variables and methods--are characteristics of any particular class.
* My name is Mahmud… I am a Class... human Class...
* What are my properties.... I can walk, I can talk, I can sleep, I can eat...These are my characteristics or properties.
* So, class is an entity where number properties are there.
* We have to define the number properties by number of method and variables.
* **A class is a group of objects that has common properties.**
* **A class in java contains---data member, method, constructor, block, class & interface**

**Q. Difference between object and class?**

|  |  |
| --- | --- |
| **Object** | **Class** |
| Object is an **instance** of a class. | Class is a **blueprint or template** from which objects are created. |
| Object is a **real-world entity** such as pen, laptop, mobile, bed, keyboard, mouse, chair etc. | Class is a **group of similar objects**. |
| Object is a **physical** entity. | Class is a **logical** entity. |
| Object is created through **new keyword** mainly e.g., Student s1=new Student (); | Class is declared using **class keyword** e.g., class Student {} |
| Object is created **many times** as per requirement. | Class is declared **once**. |
| Object **allocates memory when it is created**. | Class **doesn't allocated memory when it is created**. |
| There are **many ways to create object** in java such as new keyword, new Instance () method, clone () method, factory method and deserialization. | There is only **one way to define class** in java using class keyword. |

**Q. What are the different ways of creation of object in Java?**

* Using new operator - new xyzClass ()
* Using factory methods - xyzFactory.getInstance ( )
* Using newInstance ( ) method
* By cloning an already available object - (xyzClass)obj1.clone ( )

**Q. What is method?**

* **Whenever we want to perform any operation multiple times then we choose methods.**
* **A method is a block of code which only runs when it is called.**
* Access Modifier, Return Data Type, Method Name, Input/Arguments/Parameters, Method Body.
* You can pass data, known as parameters, into a method.
* Methods are used to perform certain actions, and they are also known as functions.
* **Why do we use methods?** To reuse code: define the code once and use it many times.
* A method must be declared within a class. It is defined with the name of the method, followed by parentheses ().
* **To call a method in Java, write the method's name followed by two parentheses () and a semicolon.**
* The most important method in Java is the main () method.
* **Method Signature:** Every method has a method signature. It is a part of the method declaration. It includes the method name and parameter list.

**\*\*\*CONSTRUCTOR\*\*\***

**Q. What is constructor?**

**Constructor is a special type of method used to initialize the object.**

**Q. What are type of Constructor? Default and parameterized constructor.**

**Q. Is duplicate constructor are allowed?** **NO**

**Q. Difference between constructor and method?**

|  |  |
| --- | --- |
| **Constructor** | **Method** |
| The constructor’s name must be same as the class name. | The method name may or may not be same as the class name. |
| A constructor must not have a return type not even void. | A method must have a return type. |
| A constructor is used to initialize the object. | A method is used to expose the behavior of an object. |

**Q. What is the purpose of a default constructor?**

The purpose of the default constructor is to **assign the default value to the objects**. The java compiler creates a default constructor implicitly if there is no constructor in the class.

**Q. Can we make constructor static?**

As we know that the static context (method, block, or variable) belongs to the class, not the object. Since Constructors are invoked only when the object is created, there is no sense to make the constructors static. However, if you try to do so, the compiler will show the compiler error.

**Q. Why does Java not Support a Static Constructor?**

When we mark anything with a static keyword, it belongs to class only, for example, static method, static variable, etc. Static methods cannot be inherited from their subclasses because they belong to the class in which they are declared.

Similarly, we cannot use a static variable in its subclasses.

In the case of a constructor, a constructor is a reusable block of code, which means we can call it from its subclasses during the creation of the objects. But, when we declare it as static, it cannot be used by its subclasses other than the declaring classes. So, it is illegal to declare a constructor as static. Thus, it will violate the whole motive of the inheritance concept.

If we declare a constructor as static, then it cannot be accessed by its subclasses and will belong to a class level only. The program will not be compiled and throw a compile-time error.

**Q. Does constructor return any value?**

Ans: yes, the constructor implicitly returns the current instance of the class.

**Q. Is constructor inherited?**

No, the constructor is not inherited. A subclass inherits all the members (fields, methods, and nested classes) from its superclass. Constructors are not members, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass.

https://www.geeksforgeeks.org/constructors-not-inherited-java/

**Q. Can you make a constructor final?**

No, the constructor cannot be final.

Explanation: In inheritance whenever you extend a class. The child class inherits all the members of the superclass except the constructors.

In other words, constructors cannot be inherited in Java therefore you cannot override constructors.

So, writing final before constructors makes no sense. Therefore, java does not allow final keyword before a constructor.

If you try, make a constructor final a compile time error will be generated saying “modifier final not allowed here”.

**Q. Can we make the constructor abstract?**

When you set a method as abstract it means:

"This method doesn't have a body and it should be implemented in a child class." But the constructor is called implicitly when the new keyword is used so it can't lack a body.

**Q. Can we overload the constructors?**

Yes, the constructors can be overloaded by changing the number of arguments accepted by the constructor or by changing the data type of the parameters.

**Q. Can we override the constructor?**

Constructor Overriding is never possible in Java.

This is because, Constructor looks like a method, but name should be as class name and no return value. Overriding means what we have declared in Super class, that exactly we have to declare in Sub class it is called Overriding.

**Q. Can I declare constructor as private?**

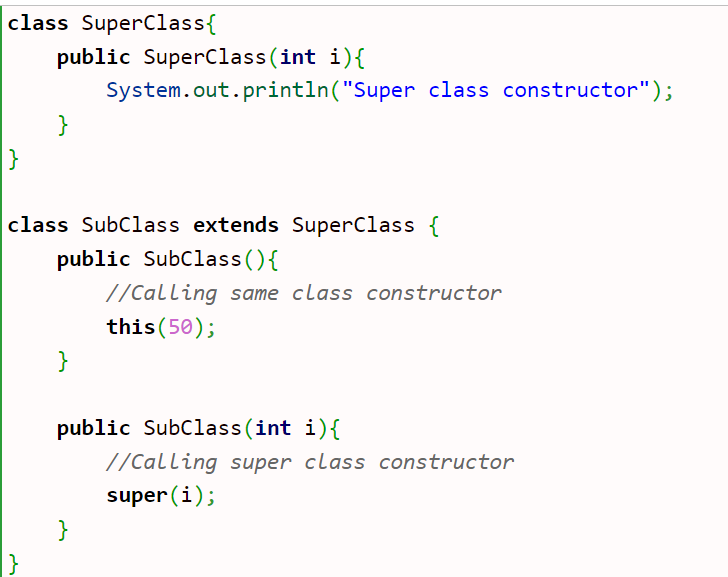
Yes, we can declare a constructor as private. If we declare a constructor as private, we are not able to create an object of a class.

**Q. What is constructor chaining?**

When a constructor calls another constructor of **same class** then this is called constructor chaining.

**Q. Can a constructor call another constructor in Java?**

Yes, a constructor can be called by another constructor in java. We can use this() to call same class constructor and super() to call super class constructor.



**Q. What if you implement only parameterized constructor in class?**

//It will throw a compilation error.

Note:

* Interfaces do not have constructors.
* A constructor can also invoke another constructor of the same class – By using this ().
* Private constructor---Singleton Class

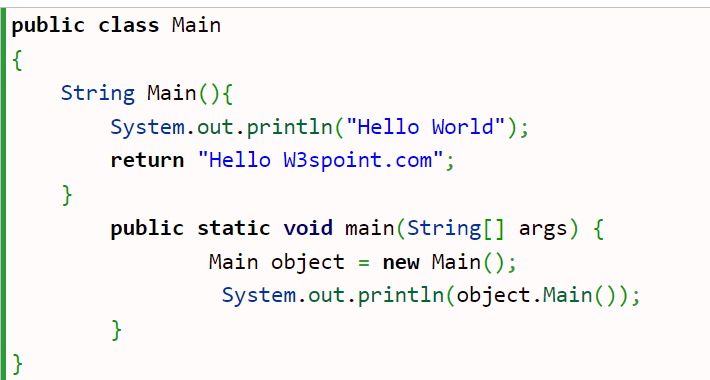
**Q. Can we call subclass constructor from super class constructor?**

No. We can’t. It will give an error.

# **Q. What happens if you keep return type for a constructor?**

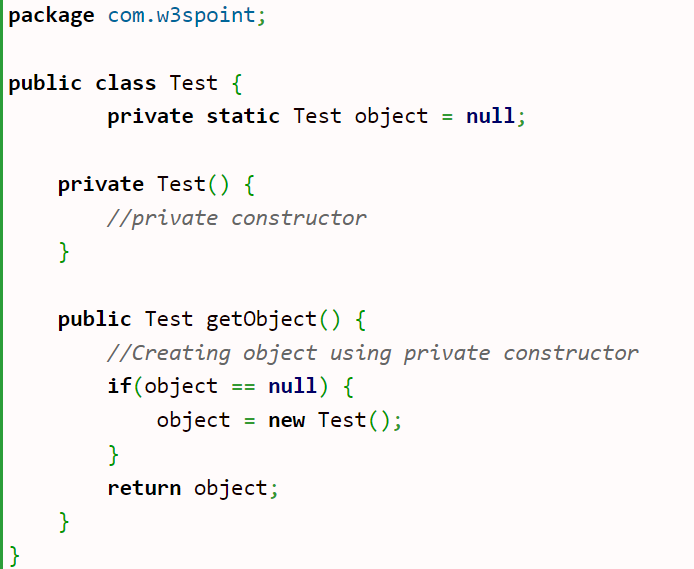
As we discussed in previous questions that we can overload a constructor so if we keep return type for a constructor it will be treated as a normal method.

**Note: Compiler gives a warning message that method has a constructor name.**



**Q.** [**What is the use of private constructor in java?**](https://www.w3schools.blog/private-constructor-in-java)

We can create private constructor in java. It is used to restrict the instantiation of a class. We cannot create an object outside of the class, if we create the private constructor. It is used to implement Singleton pattern. The main purpose of singleton pattern is to control object creation i.e., keep only one instance of a class at any time.



**\*\*\*METHOD OVERLOADING\*\*\***

**Q1. What is method overloading?**

Same method name with different parameters within the same Class.

**Q2. How can we achieve method overloading?**

There are two ways to overload the method in java:

1) By changing number of arguments or parameters

2) By changing the data type of arguments.

Method overloading increases the readability of the program.

**Q3. Why is method overloading not possible by changing the return type in java?**

Ans: In java, method overloading is not possible by changing the return type of the method only because of ambiguity. Let's see how ambiguity may occur:

public class Sum {

public static int sum(int a, int b) {

return a+b;

}

public static double sum(int a, int b) {

return a+b;

}

public static void main (String[] args) {

System.out.println(Sum.sum(10, 10)); //ambiguity

//Here, how can java determine which sum () method should be called?

//Note: Compile Time Error is better than Run Time Error.

//So, java compiler renders compiler time error if you declare the same method having same parameters.

}

}

**Q4. Can we overload java main () method?**

Yes, by method overloading. You can have any number of main methods in a class by method overloading. But JVM calls main () method which receives string array as arguments only.

class TestOverloading4{

public static void main (String [] args) {System.out.println("main with String []");}

public static void main (String args) {System.out.println("main with String");}

public static void main () {System.out.println ("main without args");}

}

**Q5. Can we overload the methods by making them static?**

No, we cannot overload the methods by just applying the static keyword to them (number of parameters and types are the same).

<https://www.javatpoint.com/corejava-interview-questions>

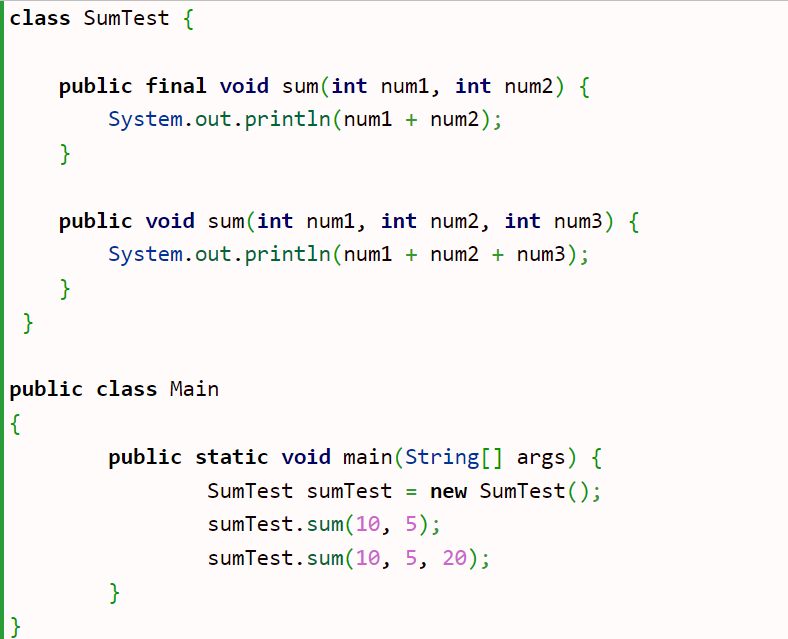
**Q6. Can we declare an Overloaded Method as Static and another one as Non-Static? YES**

**Q7. Can overloaded methods be synchronized? YES**

**Q8. What is method overloading with type promotion?**

* The byte can be promoted to short, int, long, float or double.
* The short datatype can be promoted to int, long, float or double.
* The char datatype can be promoted to int, long, float or double and so on.

**Q9. Can we declare overloaded method as Final? YES**

****

**\*\*\*METHOD OVERRIDING\*\*\***

**Q1. What is method overriding?**

When same method is present in parent class as well as in child class with same name and same input parameter or argument.

**Q2. Rules for Method overriding:**

* The method must have the same name as in the parent class.
* The method must have the same signature as in the parent class.
* Two classes must have an IS-A relationship between them.

**Q3. Can we override the static method?**

No, you can't override the static method because they are the part of the class, not the object.

**Q4. Why cannot we override static method?**

It is because the static method is the part of the class, and it is bound with class whereas instance method is bound with the object, and static gets memory in class area, and instance gets memory in a heap.

**Q5. Can we override the overloaded method?**

Yes. We can override a method which is overloaded in super class.

**Q6. Can we override the private methods?**

No, we cannot override the private methods because the scope of private methods is limited to the class and we cannot access them outside of the class.

**Q7. Can we override Final method?**

No. Final method cannot be overridden.

**Q8. Can we change the scope of the overridden method in the subclass?**

Yes, we can change the scope of the overridden method in the subclass.

However, we must notice that we cannot decrease the accessibility of the method. The following point must be taken care of while changing the accessibility of the method.

* The private can be changed to protected, public, or default.
* The protected can be changed to public or default.
* The default can be changed to public.
* The public will always remain public.

**Q9. Is it possible to override non-static method as static? NO**

**Q10. Can we synchronize override method?**

We can synchronize the overridden method in subclass.