Super Keyword in Java

The**super** keyword in java is a reference variable that is used to refer to parent class objects. An understanding of [Inheritance](https://www.geeksforgeeks.org/inheritance-in-java/)and [Polymorphism](https://www.geeksforgeeks.org/polymorphism-in-java/)is needed in order to understand the super keyword. The keyword “super” came into the picture with the concept of Inheritance. It is majorly used in the following contexts:

* Use of super with variables
* Use of super with methods
* Use of super with constructors

## **Use of super with variables**

This scenario occurs when a derived class and base class has the same data members. In that case, there is a possibility of ambiguity for the [JVM](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/). We can understand it more clearly using this code snippet:

// Java code to show use of super keyword with variables

// Base class vehicle

class Vehicle {

    int maxSpeed = 120;

}

// sub class Car extending vehicle

class Car extends Vehicle {

    int maxSpeed = 180;

    void display()

    {

        // print maxSpeed of base class (vehicle)

        System.out.println("Maximum Speed: "

                           + super.maxSpeed);

    }

}

// Driver Program

class Test {

    public static void main(String[] args)

    {

        Car small = new Car();

        small.display();

    }

}

In the above example, both base class and subclass have a member maxSpeed. We could access maxSpeed of base class in subclass using super keyword.

## **Use of super with methods**

This is used when we want to call the parent class [method](https://www.geeksforgeeks.org/methods-in-java/). So whenever a parent and child class have the same-named methods then to resolve ambiguity we use the super keyword. This code snippet helps to understand the said usage of the super keyword.

super.message();

we have seen that if we only call method message() then, the current class message() is invoked but with the use of the super keyword, message() of superclass could also be invoked.

## **Use of super with constructors**

The super [keyword](https://www.geeksforgeeks.org/list-of-all-java-keywords/)can also be used to access the parent class constructor. One more important thing is that ‘super’ can call both parametric as well as non-parametric constructors depending upon the situation. Following is the code snippet to explain the above concept

Class Student extends Person {

    Student()

    {

        // invoke or call parent class constructor

        super();

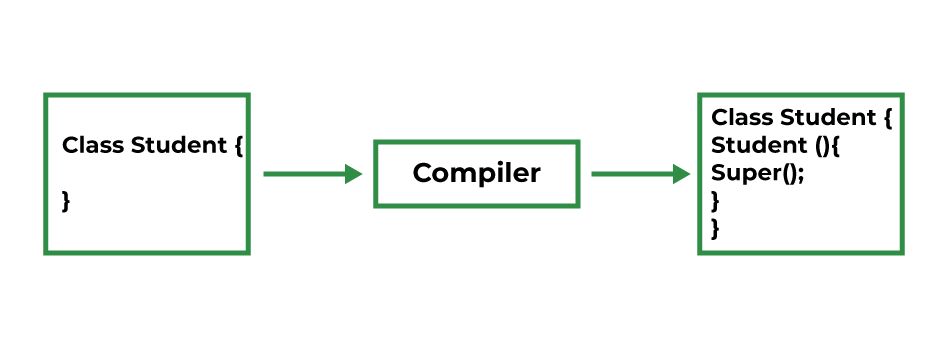
        System.out.println("Student class Constructor");

    }

}

### **Important Points to Remember while using Super Keyword**

* Call to super() must be the first statement in the Derived(Student) Class constructor because if you think about it, it makes sense that the superclass has no knowledge of any subclass, so any initialization it needs to perform is separate from and possibly prerequisite to any initialization performed by the subclass. Therefore, it needs to complete its execution first.
* If a constructor does not explicitly invoke a superclass constructor, the Java compiler automatically inserts a call to the no-argument constructor of the superclass. If the superclass does not have a no-argument constructor, you will get a compile-time error. The object *does* have such a constructor, so if the Object is the only superclass, there is no problem.



* If a subclass constructor invokes a constructor of its superclass, either explicitly or implicitly, you might think that a whole chain of constructors is called, all the way back to the constructor of Object. This, in fact, is the case. It is called [*constructor chaining*](https://www.geeksforgeeks.org/constructor-chaining-java-examples/)*.*

https://www.sitesbay.com/interview/java-super-keyword-interview-questions-in-java