Super Keyword in Java

The **super** keyword in Java is a reference variable which is used to refer immediate parent class object.

The Super Keyword in Java is used to access the data members, methods as well as the constructor of the parent class. The most important use of the Super Keyword in Java is to **avoid ambiguity** between the child class and its parent class that has the same data members or methods.

Whenever we create an instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

Usage of Java super Keyword

1. super can be used to refer immediate parent class instance variable.
2. super can be used to invoke immediate parent class method.
3. super() can be used to invoke immediate parent class constructor.

Refer immediate parent class variables

Suppose there are two classes School and Teacher and they are used to store the information of all schools in a particular region and the teachers working in those schools. The Teacher class extends the School class.

Both classes have a name variable of type *String*. In this example, we will see how to access the name variable of the parent class i.e. School.

```
// Parent class
class Parent{
  // Instance variables
  int id;
  String name = "ITM";
}

class Child extends Parent{
  // Instance variable in the child class
  int id;
  String name="SSS(M)";

  void printName() {
    // Using super.name we can access the name variable of the parent class
    // Using this.name we can access the name variable of child class
    System.out.println("Parent Name: " + super.name);
    System.out.println("Child Name: " + this.name);
  }

  public static void main(String[] args) {
    Child ob = new Child ();
    ob.printName();
  }
}
```

Invoke immediate parent class methods

Consider the same example we discussed above. Now, both the base and derived classes contain the same method, printID(). Let's see how we can access the printID() method of the parent class inside the child class.

```
// Parent class
class Parent{
  // Instance variables
  int id=1;
  String name = "ITM";

   void printID() {
    System.out.println("Parent ID: " + this.id);
  }
}

class Child extends Parent{
  // Instance variable in the child class
  int id=2;
  String name="SSS(M)";

  void printName() {
    // Using super.name we can access the name variable of the parent class
    // Using this.name we can access the name variable of child class
    System.out.println("Parent Name: " + super.name);
    System.out.println("Child Name: " + this.name);
  }

void printID() {
    super.printID();
    System.out.println("Child ID: " + this.id);

  }

  public static void main(String[] args) {
    Child ob = new Child ();
    ob.printName();
  }
}
```

Invoke immediate parent class constructor

The super keyword can invoke the parent class's default as well as parameterized constructors in the child class. We may need to call the parameterized constructors of the parent class in the case when the instance members of the parent class are initialized from the child class.

```
// Base class Person
class Person {

  Person() {
    System.out.println("Person class Constructor");
  }
}

// Derived class Teacher
class Teacher extends Person {

  Teacher() {
    // Invoke the parent class constructor
    //super();
    System.out.println("Teacher class Constructor");
  }
}

class Main {
  public static void main(String[] args) {
    // Invoke the constructor of the Teacher class
    Teacher s = new Teacher();
  }
}
```

Invoking parameterized parent class constructor:

We can call the parameterized constructor of the parent class from the subclass using the Super Keyword in Java.

```
class Parent{

  int age;

  Parent(int age) {
    this.age = age;
    System.out.println("Parent class constructor invoked!");
  }
}

public class Child extends Parent{
  String name;
  Child (String name, int age) {
    super(age);
```

```
      this.name = name;
      System.out.println("Child class constructor invoked!");
    }
    void printDetails() {
      System.out.println("Name: " + this.name);
      System.out.println("Age: " + this.age);
    }
    public static void main(String[] args) {
      Child ob = new Child("Debasish", 5);
      ob.printDetails();
    }
  }
```

1. 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 carried out by the subclass.

   For this reason, call to super() must be the first statement in the Derived(Student) Class constructor. As a result, it must finish execution first.

2. The Java compiler automatically inserts a call to the superclass's no-argument constructor if a constructor doesn't explicitly call one of its superclass constructors.

   A compile-time error will appear if the superclass does not have a no-argument constructor. There is no issue if the Object is the lone superclass because the object does have such a constructor.

3. Aassume that a full chain of constructors is called, all the way back to the constructor of Object, if a subclass constructor explicitly or implicitly calls a constructor of its superclass. In actuality, this is true. **Constructor Chaining** is the term for it.