**Super keyword in java**

**Super** keyword in java is a reference variable that is used to refer parent class object. **Super** is an implicit keyword create by JVM and supply each and every java program for performing important role in three places.

* At variable level
* At method level
* At constructor level

**Need of super keyword:**

Whenever the derived class is inherits the base class features, there is a possibility that base class features are similar to derived class features and JVM gets an ambiguity. In order to differentiate between base class features and derived class features must be preceded by super keyword.

**Syntax**

super.baseclass features.

**Super at variable level:**

Whenever the derived class inherit base class data members there is a possibility that base class data member are similar to derived class data member and JVM gets an ambiguity.

In order to differentiate between the data member of base class and derived class, in the context of derived class the base class data members must be preceded by super keyword.

**Syntax**

super.baseclass datamember name

if we are not writing super keyword before the base class data member name than it will be referred as current class data member name and base class data member are hidden in the context of derived class.

**Program without using super keyword**

**Example**

**class** Employee

{

**float** salary=10000;

}

**class** HR **extends** Employee

{

**float** salary=20000;

**void** display()

{

System.**out**.println("Salary: "+salary);//print current class salary

}

}

**class** Supervarible

{

**public** **static** **void** main(String[] args)

{

HR obj=**new** HR();

obj.display();

}

}

**Output**

Salary: 20000.0

In the above program in Employee and HR class salary is common properties of both class the instance of current or derived class is referred by instance by default but here we want to refer base class instance variable that is why we use super keyword to distinguish between parent or base class instance variable and current or derived class instance variable.

**Program using super keyword al variable level**

**Example**

**class** Employee

{

**float** salary=10000;

}

**class** HR **extends** Employee

{

**float** salary=20000;

**void** display()

{

System.**out**.println("Salary: "+**super**.salary);//print base class salary

}

}

**class** Supervarible

{

**public** **static** **void** main(String[] args)

{

HR obj=**new** HR();

obj.display();

}

}

**Output**

Salary: 10000.0

**Super at method level**

The **super keyword** can also be used to invoke or call parent class method. It should be use in case of method overriding. In other word **super keyword** use when base class method name and derived class method name have same name.

**Example of super keyword at method level**

**Example**

**class** Student

{

**void** message()

{

System.**out**.println("Good Morning Sir");

}

}

**class** Faculty **extends** Student

{

**void** message()

{

System.**out**.println("Good Morning Students");

}

**void** display()

{

message();//will invoke or call current class message() method

**super**.message();//will invoke or call parent class message() method

}

**public** **static** **void** main(String args[])

{

Student s=**new** Student();

s.display();

}

}

**Output**

Good Morning Students

Good Morning Sir

In the above example Student and Faculty both classes have message() method if we call message() method from Student class, it will call the message() method of Student class not of Person class because priority of local is high.

In case there is no method in subclass as parent, there is no need to use super. In the example given below message() method is invoked from Student class but Student class does not have message() method, so you can directly call message() method.

**Program where super is not required**

**Example**

**class** Student

{

**void** message()

{

System.**out**.println("Good Morning Sir");

}

}

**class** Faculty **extends** Student

{

**void** display()

{

message();//will invoke or call parent class message() method

}

**public** **static** **void** main(String args[])

{

Student s=**new** Student();

s.display();

}

}

**Output**

Good Morning Sir

**Super at constructor level**

The super keyword can also be used to invoke or call the parent class constructor. Constructor are calling from bottom to top and executing from top to bottom.

To establish the connection between base class constructor and derived class constructors JVM provides two implicit methods they are:

* Super()
* Super(...)

**Super()**

**Super()** It is used for calling super class default constructor from the context of derived class constructors.

**Super keyword used to call base class constructor**

**Syntax**

**class** Employee

{

Employee()

{

System.**out**.println("Employee class Constructor");

}

}

**class** HR **extends** Employee

{

HR()

{

**super**(); //will invoke or call parent class constructor

System.**out**.println("HR class Constructor");

}

}

**class** Supercons

{

**public** **static** **void** main(String[] args)

{

HR obj=**new** HR();

}

}

**Output**

Employee class Constructor

HR class Constructor

**Note:**super() is added in each class constructor automatically by compiler.

In constructor, default constructor is provided by compiler automatically but it also adds **super()** before the first statement of constructor.If you are creating your own constructor and you do not have either this() or super() as the first statement, compiler will provide super() as the first statement of the constructor.

**Super(...)**

**Super(...)** It is used for calling super class parameterize constructor from the context of derived class constructor.

**Important rules**

Whenever we are using either super() or super(...) in the derived class constructors the **super** always must be as a first executable statement in the body of derived class constructor otherwise we get a compile time error.

**The following diagram use possibilities of using super() and super(........)**

**Rule 1 and Rule 3**

Whenever the derived class constructor want to call default constructor of base class, in the context of derived class constructors we write super(). Which is optional to write because every base class constructor contains single form of default constructor?

**Rule 2 and Rule 4**

Whenever the derived class constructor wants to call parameterized constructor of base class in the context of derived class constructor we must write super(...). which is mandatory to write because a base class may contain multiple forms of parameterized constructors.