Super keyword

Super keyword in java is a reference variable that is used to refer parent class object. Super is an implicit keyword created 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 inherits 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 at 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
}
}
```

of Student class notof Person class because priority of local is high.
```
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

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 constructor.

Super keyword used to call base class constructor

```
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.

super(….) (with parameters) it is used for calling super class parameterized constructor from the context of derived class constructor explicitly.

```
 class Person
    {
    int id;
    String name;
    Person(int id,String name)
    {
    this.id=id;
    this.name=name;
    }
    }
    class Emp extends Person
    {
    float salary;
    Emp(int id,String name,float salary)
    {
    super(id,name);//reusing parent constructor
    this.salary=salary;
    }
    void display()
    {
    System.out.println(id+" "+name+" "+salary);
    }
 }
 class TestSuper5
{
 public static void main(String[] args){
 Emp e1=new Emp(1,"ankit",45000f);
 e1.display();
 }
 }
```
Note: When the super class has a parameterized constructor and if you want to use that constructor in a sub class constructor, you have to call the super class constructor explicitly otherwise the compiler reports an error.