

SUPER KEYWORD

The **super** is a reference variable that is used to refer to an immediate parent class object. Whenever you create the instance of a subclass, an instance of the parent class is created implicitly i.e., referred to by the super reference variable.

USAGE OF SUPER KEYWORD

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

1) SUPER IS USED TO REFER IMMEDIATE PARENT CLASS INSTANCE VARIABLE.

The problem without super keyword

```
1. class Vehicle{
2.     int speed=50; // parent instance variable
3. }
4.
5. class Bike extends Vehicle{
6.     int speed=100; // child instance variable
7.
8.     void display(){
9.         System.out.println(speed);
10.    }
11.    public static void main(String args[]){
12.        Bike b=new Bike();
13.        b.display();
14.
15.    }
16. }
```

Output:100

In the above example Vehicle and Bike, both classes have a common property speed. An instance variable of the current class is referred to by instance by default, but I must refer parent class instance variable which is why we use the super keyword to differentiate between the parent class instance variable and the current class instance variable.

Solution by super keyword

```
1. //example of super keyword
2. class Vehicle{
3.     int speed=50;
4. }
5.
6. class Bike extends Vehicle{
7.     int speed=100;
8.
9.     void display(){
10.        System.out.println(super.speed);//will print speed of Vehicle now //50
11.    }
12.    public static void main(String args[]){
13.        Bike b=new Bike();
14.        b.display();
15.    }
16. }
```

Output:50

2) SUPER IS USED TO INVOKE THE PARENT CLASS CONSTRUCTOR.

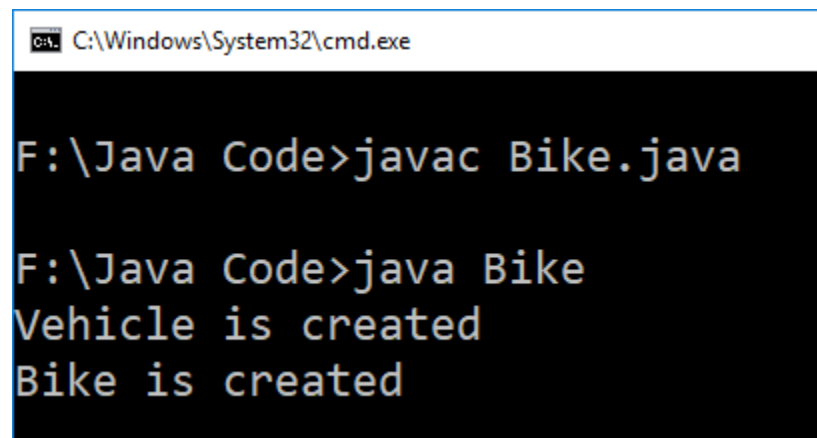
The super keyword can also be used to invoke the parent class constructor as given below:

```
1. class Vehicle{
2.     Vehicle(){System.out.println("Vehicle is created");}
3. }
4.
5. class Bike extends Vehicle{
6.     Bike(){
7.         super();//will invoke parent class constructor
8.         System.out.println("Bike is created");
9.     }
10.    public static void main(String args[]){
11.        Bike b=new Bike();
12.
13.    }
14. }
```

Output: Vehicle is created
Bike is created

EXAMPLE OF SUPER KEYWORD WHERE SUPER() IS PROVIDED BY THE COMPILER IMPLICITLY.

```
1  class Vehicle{
2      Vehicle(){System.out.println("Vehicle is created");}
3      }
4
5      class Bike extends Vehicle{
6          Bike(){
7              System.out.println("Bike is created");
8          }
9      public static void main(String args[]){
10         Bike b=new Bike();
11
12     }
13 }
```



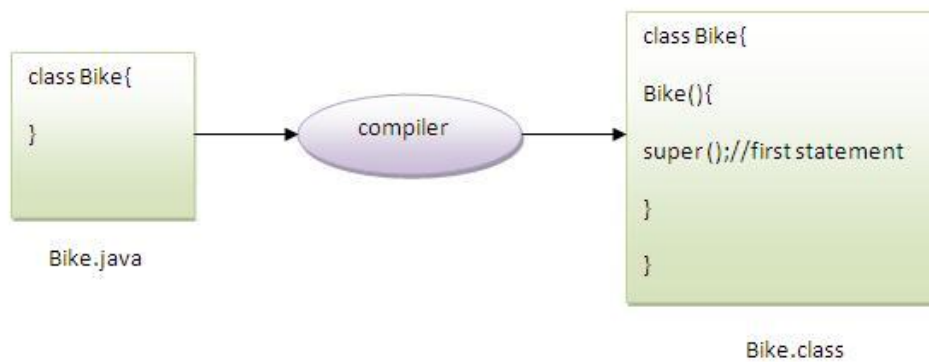
```
C:\Windows\System32\cmd.exe

F:\Java Code>javac Bike.java

F:\Java Code>java Bike
Vehicle is created
Bike is created
```

Reason:

As we know well the default constructor is provided by the compiler automatically, but it also adds `super ()` for the first statement. If you are creating your own constructor and you don't have either this `()` or `super()` as the first statement, the compiler will provide `super()` as the first statement of the constructor.

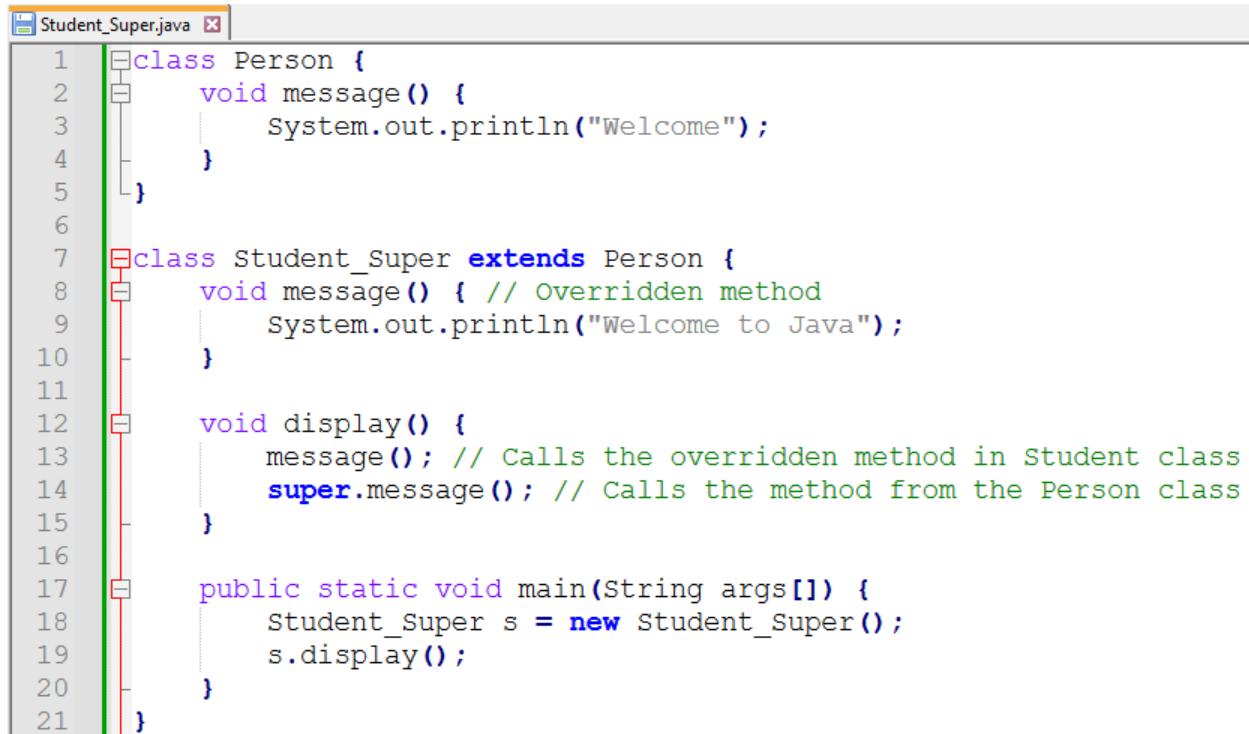


```
1  class Vehicle{
2      Vehicle() {System.out.println("Vehicle is created");}
3  }
4
5      class Bike extends Vehicle{
6          Bike() {
7              System.out.println("Bike is created");
8              super();
9          }
10     public static void main(String args[]){
11         Bike b=new Bike();
12     }
13 }
14 }
15
```

```
F:\Java Code>javac Bike.java
Bike.java:8: error: call to super must be first statement in constructor
    super();
    ^
1 error
```

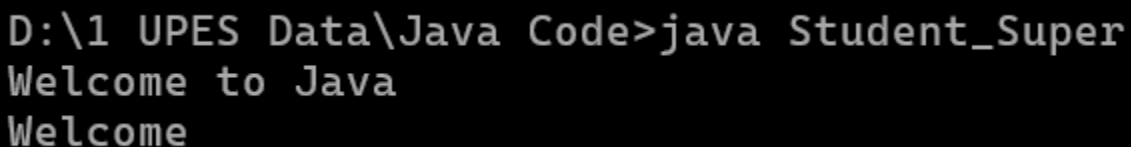
3) SUPER CAN BE USED TO INVOKE THE PARENT CLASS METHOD.

The super keyword can also be used to invoke the parent class method. It should be used in case the subclass contains the same method as the parent class as in the example given below:



```
1 class Person {
2     void message() {
3         System.out.println("Welcome");
4     }
5 }
6
7 class Student_Super extends Person {
8     void message() { // Overridden method
9         System.out.println("Welcome to Java");
10    }
11
12    void display() {
13        message(); // Calls the overridden method in Student class
14        super.message(); // Calls the method from the Person class
15    }
16
17    public static void main(String args[]) {
18        Student_Super s = new Student_Super();
19        s.display();
20    }
21 }
```

Output:



```
D:\1 UPES Data\Java Code>java Student_Super
Welcome to Java
Welcome
```

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

Note: Why does message(); work inside display() without an object?

In the display() method of Student, message(); is called **directly**, without an object. This works because display() and message() both belong to the same instance of Student. When calling a non-static method inside another non-static method of the same class, the method is implicitly invoked using this.message();.

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

Program in case super is not required.

```
class Person {  
    void message() {  
        System.out.println("welcome");  
    }  
}  
  
class Student extends Person {  
    void display() {  
        message(); // Calls the inherited message() method from Person class  
    }  
  
    public static void main(String args[]) {  
        Student s = new Student();  
        s.display();  
    }  
}
```

Output:

welcome