## 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**

```
class Vehicle{
 int speed=50;
}

class Bike extends Vehicle{
 int speed=100;

 void display(){
  System.out.println(speed);
 }
 public static void main(String args[]){
  Bike b=new Bike();
  b.display();

}
}
```

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**

```
//example of super keyword
class Vehicle{
  int speed=50;
}

class Bike extends Vehicle{
  int speed=100;

  void display(){
   System.out.println(super.speed);//will print speed of Vehicle now  //50
  }
  public static void main(String args[]){
   Bike b=new Bike();
   b.display();
}
}
```

Output:50

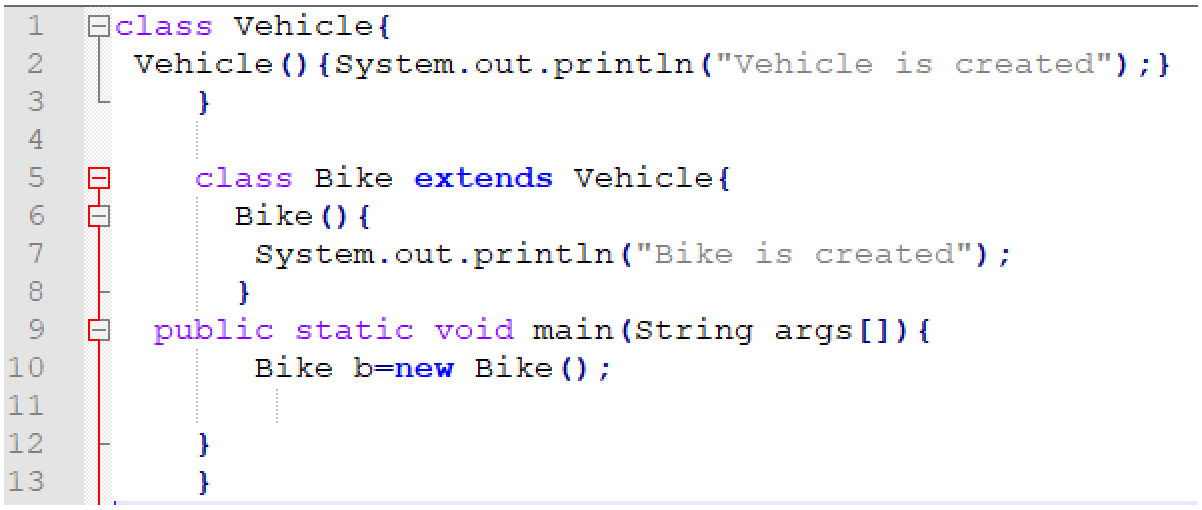
**2) SUPER IS USED TO INVOKE THE PARENT CLASS CONSTRUCTOR.**

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

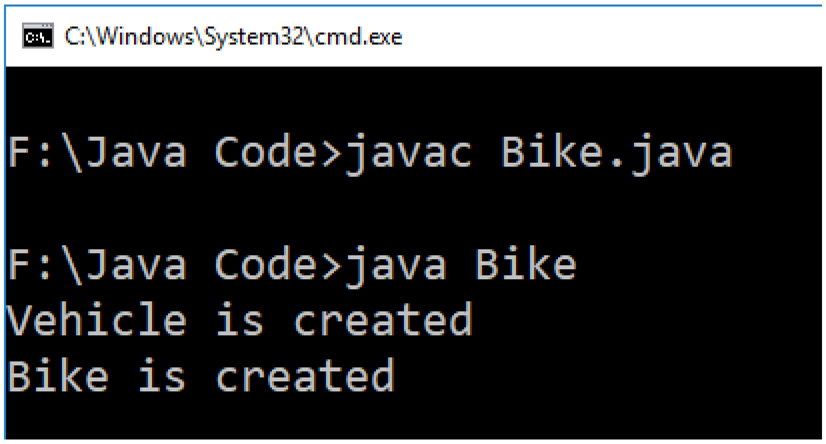
```
class Vehicle{
  Vehicle(){System.out.println("Vehicle is created");}
}

class Bike extends Vehicle{
  Bike(){
   super();//will invoke parent class constructor
   System.out.println("Bike is created");
  }
  public static void main(String args[]){
   Bike b=new Bike();

}
}
```

Output: Vehicle is created
            Bike is created

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

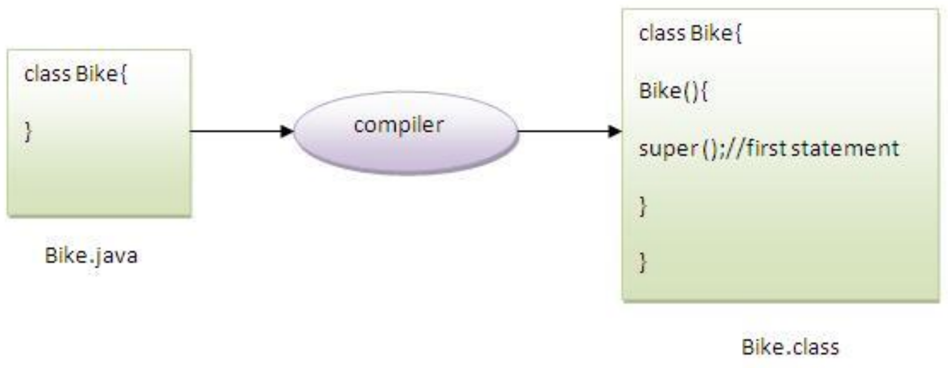


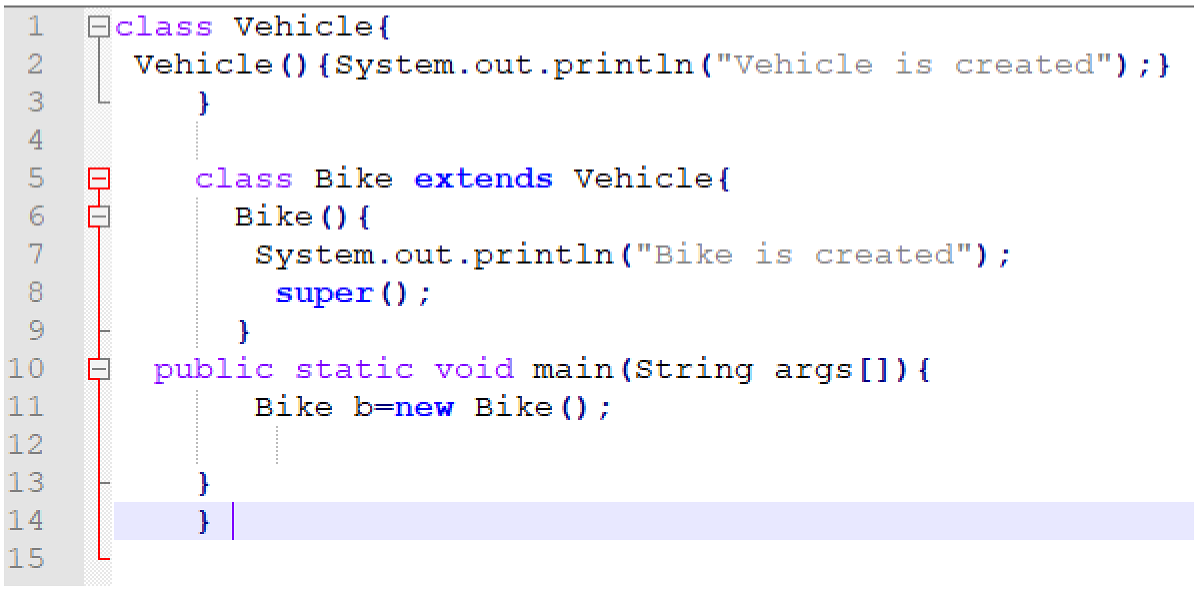
```
class Vehicle{
 Vehicle(){System.out.println("Vehicle is created");}
    }

    class Bike extends Vehicle{
      Bike(){
       System.out.println("Bike is created");
      }
  public static void main(String args[]){
       Bike b=new Bike();

    }
    }
```



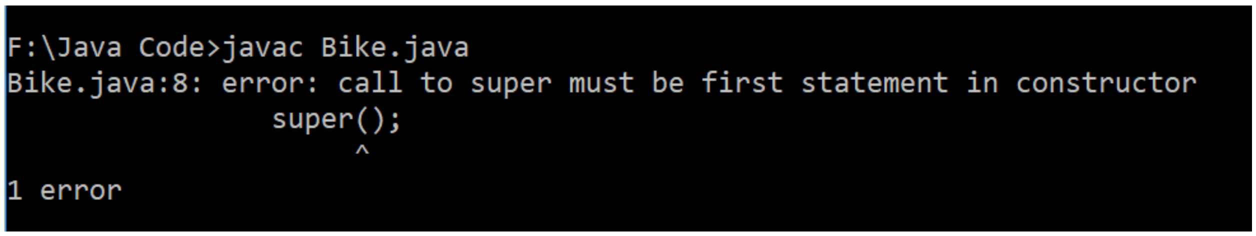
```
F:\Java Code>javac Bike.java

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.





```java
class Vehicle{
 Vehicle(){System.out.println("Vehicle is created");}
    }

    class Bike extends Vehicle{
      Bike(){
       System.out.println("Bike is created");
        super();
      }
  public static void main(String args[]){
       Bike b=new Bike();

    }
    }
```



```
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:

```
class Person
{
void message()
{
System.out.println("Welcome");
}
}


class Student extends Person
{
void message()//overridden method
{
System.out.println("Welcome to java");
}
  void display()
{
message();//will invoke current class message() method
super.message();//will invoke parent class message() method
}
public static void main(String args[])
{
Student s=new Student();
s.display();
}
}
```

**Output:**

```
Welcome to java
 Welcome
```

In the above example Student and Person 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 is given to local.

In case there is no method in the subclass as a 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 in case super is not required.**

```
class Person{
void message(){System.out.println("welcome");}
}

class Student extends Person{

void display(){
message();//will invoke parent class message() method
}

public static void main(String args[]){
Student s=new Student();
s.display();
}
}
```

Output:welcome