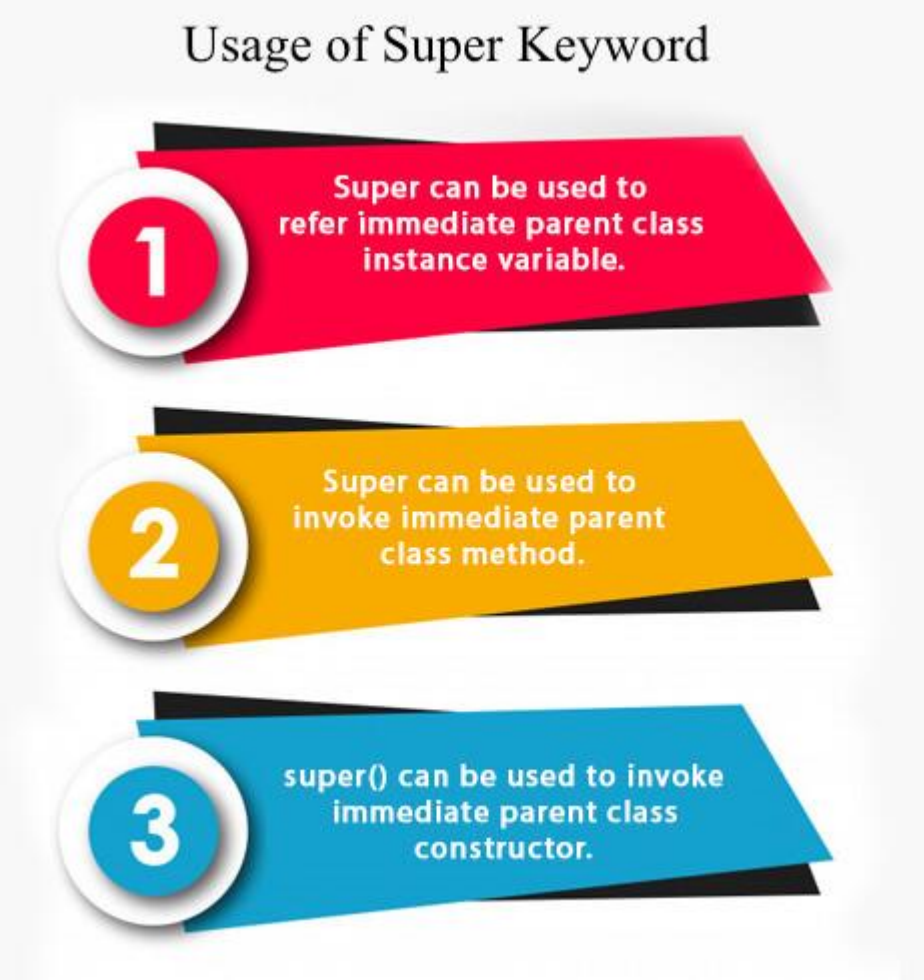
Super Keyword in Java

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

Whenever you create the 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.



1**) super is used to refer immediate parent class instance variable.**

We can use super keyword to access the data member or field of parent class. It is used if parent class and child class have same fields.

```
class Animal
{
String color="white";
}
class Dog extends Animal{
String color="black";
void printColor(){
System.out.println(color);//prints color of Dog class
System.out.println(super.color);//prints color of Animal class
}
}
class TestSuper1{
public static void main(String args[]){
Dog d=new Dog();
d.printColor();
}
}
```

Output:

```
black
white
```

2) super can be used to invoke parent class method

```
        class Animal{
        void eat(){System.out.println("eating...");}
        }
        class Dog extends Animal{
        void eat(){System.out.println("eating bread...");}
        void bark(){System.out.println("barking...");}
        void work(){
        super.eat();
        bark();
        }
}
class TestSuper2{
public static void main(String args[]){
```

```
        Dog d=new Dog();
d.work();
}}
```

Output:

```
eating...
barking...
```

3) super is used to invoke parent class constructor.

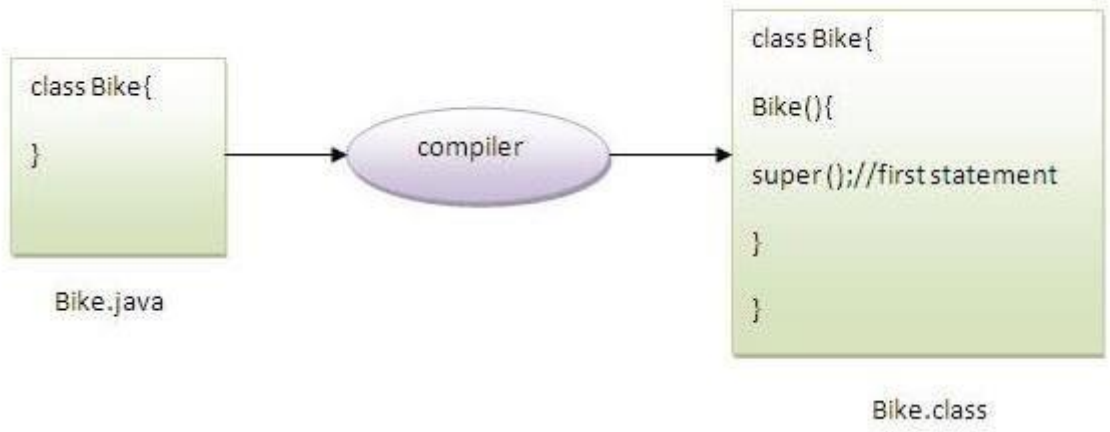
The super keyword can also be used to invoke the parent class constructor. Let's see a simple example:

```
class Animal{

Animal()

{

System.out.println("animal is created");

}

        }
        class Dog extends Animal
        {
        Dog(){
        super();
        System.out.println("dog is created");
        }
        }
class TestSuper3
{
public static void main(String args[])
{
Dog d=new Dog();
}}
```

Output:

animal is created
dog is created

> ***Note: super() is added in each class constructor automatically by compiler if there is no super() or this().***



As we know well that default constructor is provided by compiler automatically if there is no constructor. But, it also adds super() as the first statement.

Another example of super keyword where super() is provided by the compiler implicitly.

```
class Animal{
Animal(){System.out.println("animal is created");}
}
class Dog extends Animal{
Dog(){
System.out.println("dog is created");
}
}
class TestSuper4{
public static void main(String args[]){
Dog d=new Dog();
```

```
}}
```

Output:

```
animal is created
dog is created
```

super example: real use

```
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();
}}
```

Output:

```
1 ankit 45000
```