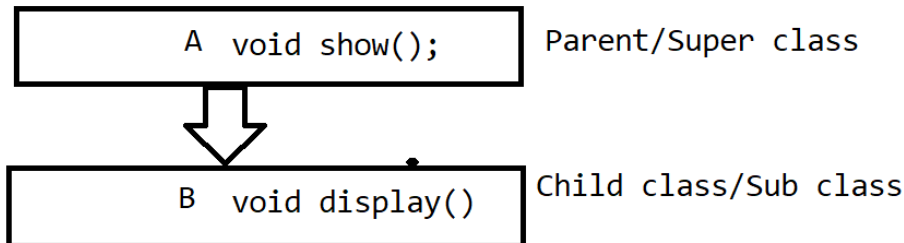


## Inheritance

---

Inheritance means to transfer the properties of one class to another class called as inheritance.



Note : in Class B contain two function name as `display()` and `show()`  
Beacause B is Child class of A means All properties from class A present in class B.

## Why use the inheritance or what is the benefit of inheritance?

---

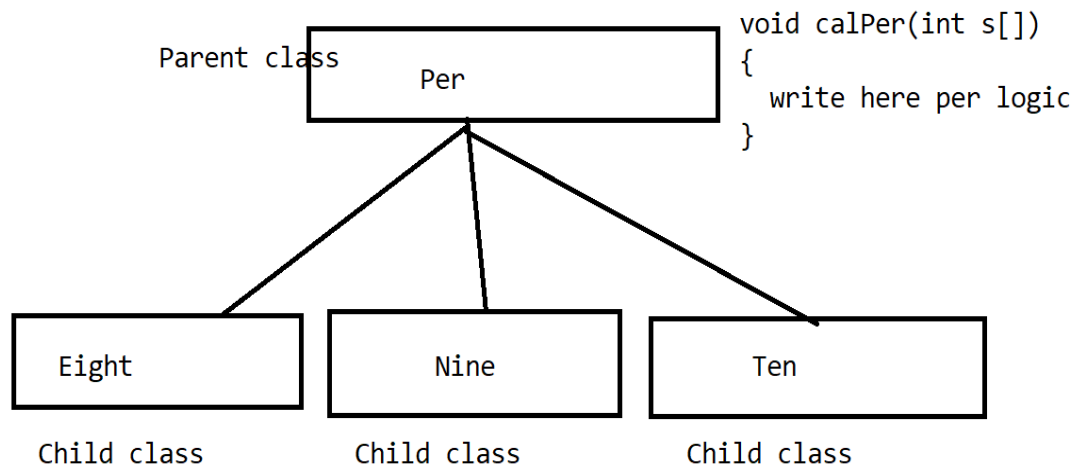
The major benefit of inheritance given below

- 1) Code reusability:** means as per our example we not need to create the object of class A we can access the all member of class A using class B object
- 2) Generalization:** generalization means if we have some logics and we want to reuse multiple then we can create the logic and use it multiple places without creating object just we have to inherit it.

## Suppose consider we have the example

---

We have the one class name as per with method `calPer()` and we have the three different classes name as Eight , Nine and Ten we want to calculate the percentage for all classes so writing `calPer(int [])` in every class we can declare the method in single parent class and inherit the parent class in Eight Nine and Ten classes.



Note : Here we not need not create the object of Per class  
Per logic present in Eight , Nine And Ten Virtually from its parent class

## How to perform the inheritance using java or How to transfer the properties from one class to another class in java

---

If we want to perform or if we want to transfer the properties of one class to another class in java we have the extends keyword.

```
class A //parent class
{
}
class B extends A //child class
{
}
```

**Note:** In The case of java we cannot create any program without inheritance.

Because in the case of java every class having a default parent class called as Object class.

### Example

---

### Internal meaning

<pre>class A { } </pre>	<pre>class A extends java.lang.Object { } </pre>
-------------------------	--

## Q. Why java provide Object class as parent to every class?

Because Object class contain the some common methods those required to every class in java

1) boolean equals(Object) 2) int hashCode() 3) String toString()  
 4) Object clone() 5) Class getClass() 6) void finalize() 7) void wait()  
 8) void wait(int) 9) void notify() 10) void notifyAll() 11) static { } block

If we want to see this method we can the following command on command prompt.

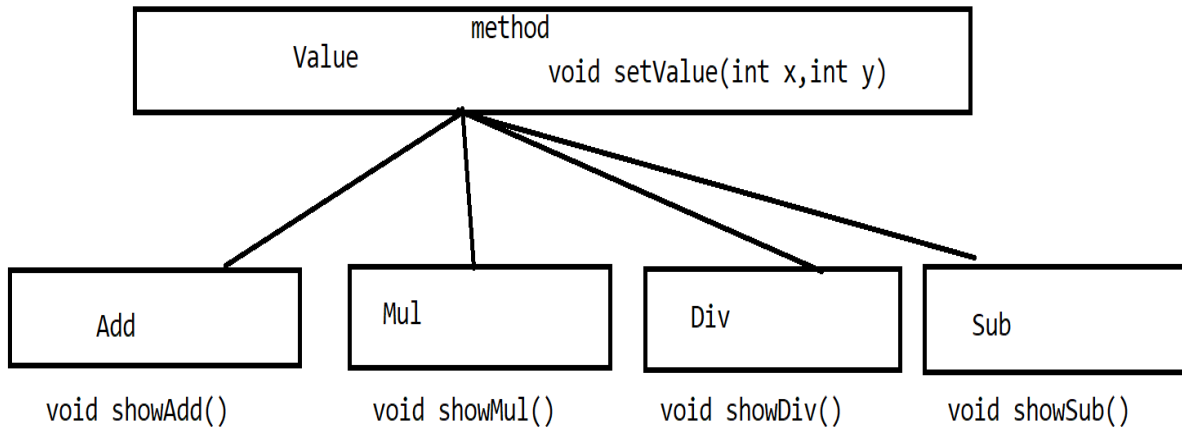
```
C:\Users\Admin>javap java.lang.Object      javap shwo the detail of every class
Compiled from "Object.java"
public class java.lang.Object {
    public java.lang.Object();
    public final native java.lang.Class<?> getClass();
    public native int hashCode();
    public boolean equals(java.lang.Object);
    protected native java.lang.Object clone() throws java.lang.CloneNotSupportedException;
    public java.lang.String toString();
    public final native void notify();
    public final native void notifyAll();
    public final native void wait(long) throws java.lang.InterruptedException;
    public final void wait(long, int) throws java.lang.InterruptedException;
    public final void wait() throws java.lang.InterruptedException;
    protected void finalize() throws java.lang.Throwable;
    static {};
}
```

**Note:** all methods we will discuss in some later chapters.

In the case of inheritance we not need to create the object of parent class we can access the parent member using child object.

## Example

WAP to create the Calculate Example using Inheritance following Diagram shows the Example detail



In above example we have the parent class name as **Value** with function name as `setValue()` with two parameter and we have the four child classes name as **Add** which is used for perform addition operation using `showAdd()` function, **Mul** class which is used for perform multiplication with `showMul()`, **Div** class which is used for calculate division with `showDiv()` function and **Sub** class which is used for calculate the subtraction with `showSub()` function

Here we have the some option like as

Case 1: for addition operation

Case 2: for multiplication

Case 3: division operation

Case 4: subtraction operation

As per our example we not need to create the object of **Value** class we can use the **Value** content using **Add**, **Sub**, **Mul** and **Div** class objects.

Source Code given below

---

```
package org.techhub;

import java.util.*;
class Value {
    int a, b;

    void setValue(int x, int y) {
        a = x;
        b = y;
    }
}

class Add extends Value {
    void showAdd() {
        System.out.printf("Addition is %d\n", a + b);
    }
}

class Mul extends Value {
    void showMul() {
        System.out.printf("Multiplication is %d\n", a * b);
    }
}

class Sub extends Value {
    void showSub() {
        System.out.printf("Substraction is %d\n", a - b);
    }
}
```

```
}
```

```
class Div extends Value {  
    void showDiv() {  
        System.out.printf("Division is %d\n", a / b);  
    }  
}
```

```
public class CalculatorApplication {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Scanner xyz = new Scanner(System.in);  
        int choice;  
        System.out.println("1:Addition");  
        System.out.println("2:Multiplication");  
        System.out.println("3:Division");  
        System.out.println("4:Subtraction");  
        System.out.println("Enter the two values");  
        int a = xyz.nextInt();  
        int b = xyz.nextInt();  
        System.out.println("Enter your choice");  
        choice = xyz.nextInt();  
        switch (choice) {  
            case 1:  
                Add ad = new Add();  
                ad.setValue(a, b);  
                ad.showAdd();  
                break;  
            case 2:  
                Mul m = new Mul();  
                m.setValue(a, b);
```

```

        m.showMul();
        break;
    case 3:
        Div d = new Div();
        d.setValue(a, b);
        d.showDiv();
        break;
    case 4:
        Sub s = new Sub();
        s.setValue(a, b);
        s.showSub();
        break;
    default:
        System.out.println("Wrong choice");
    }
}
}
Output

```

---

```

1:Addition
2:Multiplication
3:Division
4:Substraction
Enter the two values
10
20
Enter your choice
1
Addition is 30

```

## Constructor in Inheritance

---

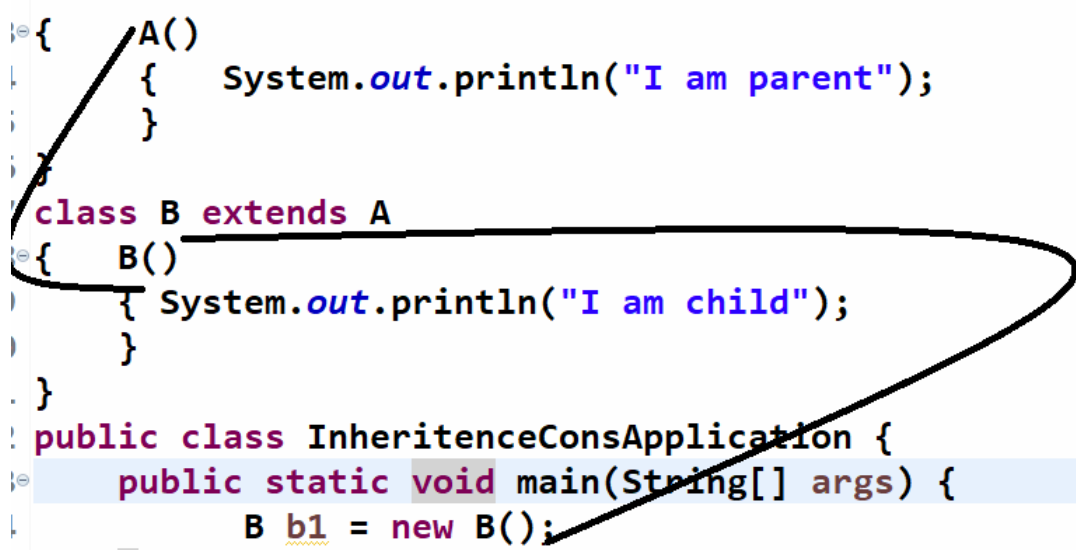
When we have the default constructor in parent class and constructor in child class and when we create the object of child class then by default parent constructor get executed before child class constructor.

Means JVM call the parent object internally before child object  
So parent constructor gets executed automatically before child constructor.

### Source Code Example

---

```
package org.techhub;  
class A  
{  
    A()  
    {  
        System.out.println("I am parent");  
    }  
}  
class B extends A  
{  
    B()  
    {  
        System.out.println("I am child");  
    }  
}  
public class InheritanceConsApplication {  
    public static void main(String[] args) {  
        B b1 = new B();  
    }  
}
```



### Output

---

```
I am parent  
I am child
```



**Note:** when we have the parameter in parent class constructor then JVM not execute the parent of constructor before child constructor automatically in this case we have to use the `super()` constructor for passing parameter to parent constructor from child class constructor.

`super()` constructor must be first line of code in child class constructor

Example given below

---

```
1 package org.techhub;
2 class A
3 {   A(int x)                                100
4     {   System.out.println("I am parent"+x);
5     }
6 }
7 class B extends A
8 {   B()
9     {   super(100);
10        System.out.println("I am child");
11    }
12 }
13 public class InheritanceConsApplication {
14     public static void main(String[] args) {
15         B b1 = new B();
16     }
17 }
18
```

**Output**

---

```
I am parent100  
I am child
```

Q. what is the super () constructor?

---

Super() constructor is used for perform constructor chaining in inheritance between child and parent means when we have parameter present in parent class then child class having responsibility to pass parameter to parent constructor then we have to use the super() constructor in child class constructor with first line.