SESSION_16

12 October 2021 09:12 AM

----------**CONSTRUCTOR**----------

In below example Hello is a constructor.

Conditions for constructors are:

- ***Name** of constructor is **same as main class**.*
- *It **cannot return** anything.*
- *It can take a **parameter**, this type of constructor is called as DEFAULT CONSTRUCTOR.*

Type of Constructor:

1. *Default Constructor*
2. *Constructor which takes parameter.*

```
1 import java.util.Scanner;
2
3 public class Hello{
4         int age;
5         String name;
6         public Hello(int a,String n){
7                 age=a;
8                 name=n;
9         }
10
11        public void Print(){
12                System.out.println(age+" " + name);
13        }
14
15        public static void main(String ar[]){
16                Hello h = new Hello(10,"Ram");
17                Hello h2 = new Hello(20,"Nayan");
18
19
20                h.Print();
21                h2.Print();
22        }
23
24 }
```

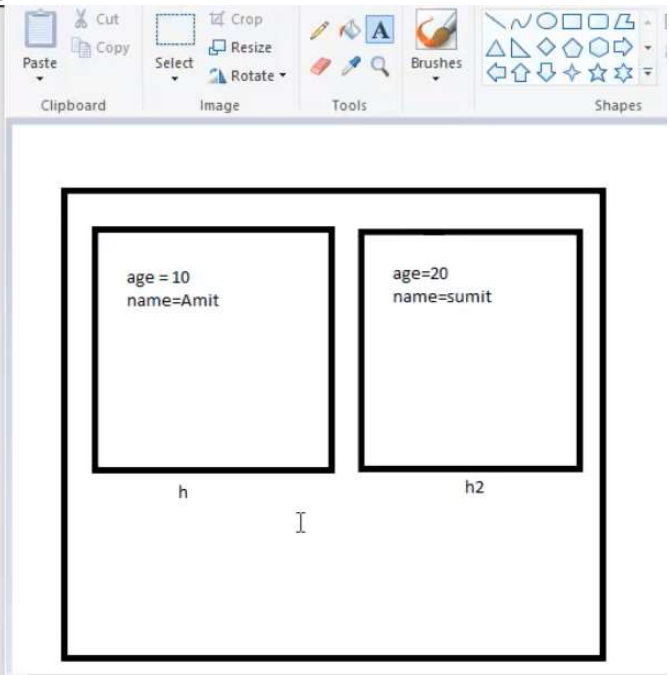
Use **this** keyword to point at the **current instance** of the constructor.

*In below example we have **two variables age, name in main class hello and constructor hello**.*

*So whenever we create instance of constructor 'hello' to use age and name. **Complier** will be **confused between parameter of constructor and variable created in the main class** as **constructor has same name as main class**.*

*That's why we **use this keyword to point out the current instance of an object** we create **of constructor.***

```
import java.util.Scanner;

public class Hello{
        int age;
        String name;

        public Hello(){}

        public Hello(int age, String name){
                this.age=age;
                this.name=name;
        }

        public void Print(){
                System.out.println(age+" " + name);
        }

        public static void main(String ar[]){
                Hello h = new Hello(10,"Ram");
                h.Print();

                Hello h2 = new Hello(20,"Sumit");
                h2.Print();

                h.Print();
        }

}
```



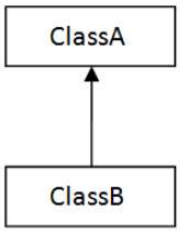
```java
public class Hello{

    int age;
    String name;
    int roll;
    float fees;

    public Hello(){
        System.out.println("1 Welcome To System");
        this.Print();
    }
    public Hello(int roll, String name, int age){
        this();
        System.out.println("2.1");
        this.age=age;
        this.name=name;
        this.roll=roll;
        System.out.println("2.2");
        this.Print();
    }
    public Hello(int roll, String name, int age, float fees){
        this(roll,name,age);
        System.out.println("3.2");
        this.fees=fees;
        this.Print();
    }
    public void Print(){
        System.out.println(age+" "+name+" "+roll+" "+fees);
    }
    public static void main(String ar[]){
        Hello h = new Hello(101,"Amit",25,10000);
        h.Print();
    }

}
```

Output

```
java -cp /tmp/btsLU2Zz1v Hello
1 Welcome To System
0 null 0 0.0
2.1
2.2
25 Amit 101 0.0
3.2
25 Amit 101 10000.0
25 Amit 101 10000.0
```

----------**INHERITENCE**----------

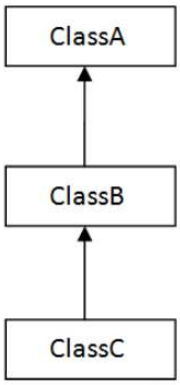
Above is the simple type of Inheritance, where a class inherited the property of another class using " **extendeds class_parent_name**". In our case class B() extends A.



1) Single

```java
public class Hello{

        public static void main(String []args){
                B b=new B();
                b.funB();
                b.funA();
        }

}

class A{
        public void funA(){
                System.out.println("A");
        }
}

class B extends A{
        public void funB(){
                System.out.println("B");
        }
}
```

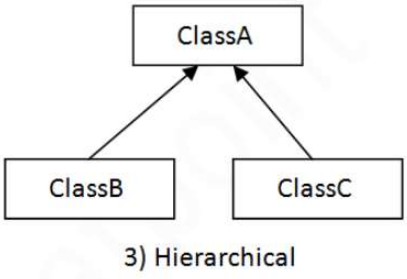
```
coder@ubuntu:~/Desktop/Java Class$ javac Hello.java
coder@ubuntu:~/Desktop/Java Class$ java Hello
B
A
```



2) Multilevel

```java
public class Hello{

	public static void main(String []args){
		C b=new C();
		b.funA();

		b.funB();

		b.funC();
	}

}

class A{
	public void funA(){
		System.out.println("A");
	}
}

class B extends A{
	public void funB(){
		System.out.println("B");
	}
}
class C extends B{
	public void funC(){
		System.out.println("C");
	}
}
```

Runtime polymorphism and overriding.(if every class has a same function):
Parent obj = new child; *//using memory of child class.*
Grandparent obj = new child; *//using the memory of child class by grandparent class.*
Grandparent obj = new parent; *//using the memory of parent class by grandparent class.*

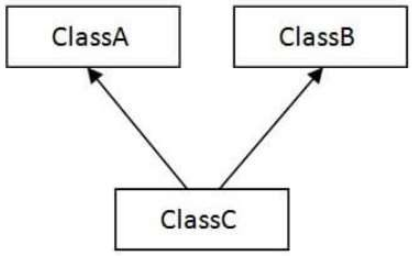


3) Hierarchical

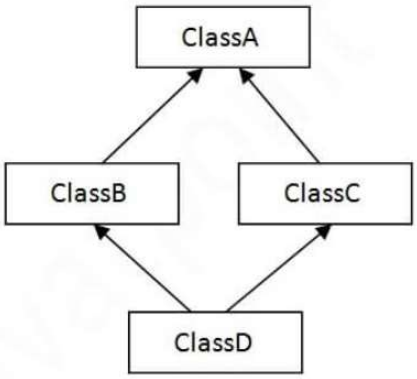
```java
public class Hello{

	public static void main(String []args){
		A obj=new C();
		obj.funA();
		//obj.funB();
		//obj.funC();
	}

}

class A{
	public void funA(){
		System.out.println("A");
	}
}

class B extends A{
	public void funB(){
		System.out.println("B");
	}
}
class C extends A{
	public void funC(){
		System.out.println("C");
	}
}
```

In below example

```
public class Hello{

        public static void main(String []args){
                A obj = new A();
                obj.fun();
        }

}

class A{
        public void fun(){
                System.out.println("A");
        }
}

class B extends A{
        public void fun(){
                System.out.println("B");
        }
}
class C extends B{
        public void fun(){
                System.out.println("C");
        }
}
```



4) Multiple



5) Hybrid

----------**ACCESS MODIFIER**----------

Default and Protected can be accessible within the **package** not outside the package. But if we import the package which contains protected class then we can use protected class outside package.

Private can be accessible **within the class only** not outside the main class. Not even if we import the package where private class is.

As you can see funB0() and funB3() which is default and private can't be accessed in newpackage even we import the package batch 89.

```java
package newpackage;

import batch89.B;

public class XYZ extends B {
    public static void main(String[] args) {
        XYZ obj=new XYZ();
        obj.funA0();
        obj.funB2();
        obj.funA2();
    }
}
```

```java
package batch89;

public class B extends A{
    void funB0(){
        System.out.println("finB0");
    }
    public void funB1(){
        System.out.println("Public finB1");
    }
    protected void funB2(){
        System.out.println("protected finB2");
    }
    private void funB3(){
        System.out.println("private finB3");
    }
}
```

UPCASTING

*Here in below case we are converting **byte to double** which is possible as size of double is greater than byte.*

```java
package newpackage;


public class NewClass {
    public static void main(String[] args) {
        byte b = 20;
        int i = b;
        float f = i;
        double d = f;
    }
}
```

DOWNCASTING

*Here in below case we are converting **double to byte** which is not possible directly as size of double is greater than byte. i.e. why we specify float, int , byte before the identifiers.*

```java
package newpackage;


public class NewClass {
    public static void main(String[] args) {
        byte b;
        int i;
        float f ;
        double d;
        d=3654.589354;
        f=(float)d;
        i=(int)f;
        b=(byte)i;

    }
}
```

Swapping a 2 variable with 3rd variable.

```java
import java.util.Scanner;
public class swap_three {

  public static void main(String[] args)
  {
            Scanner scan = new Scanner(System.in)

    System.out.println("Enter the  1st Number:" );
    int x = scan.nextInt();,
            System.out.println("Enter the  2st Number:" );
            int y = scan.nextInt();

    int temp = x;
    x = y;
    y = temp;

    System.out.print("Now after swap");
    System.out.println("1st Number is  = " + x);
    System.out.println("2nd Number is = " + y);
  }
}
```

Without a 3rd variable

```java
import java.util.Scanner;
public class swap{
```

```
  public static void main(String[] args)
  {
            Scanner scan = new Scanner(System.in)

    System.out.println("Enter the  1st Number:" );
    int x = scan.nextInt();,
            System.out.println("Enter the  2st Number:" );
            int y = scan.nextInt();

    x = x + y;
    y = x - y;
    x = x - y;

    System.out.print("Now after swap");
    System.out.println("1st Number is  = " + x);
    System.out.println("2nd Number is = " + y);
  }
}
```

Create a calculator using switch case

```
import java.util.Scanner;

public class cal {
 public static void main(String[] args) {

  int operator;
  Double number1, number2, result;


  Scanner scan = new Scanner(System.in);


  System.out.println("Choose an operator: 1(+), 2(-), 3(*), or 4(/)");
  operator = scan.nextInt();


  System.out.println("Enter first number");
  number1 = input.nextDouble();

  System.out.println("Enter second number");
  number2 = input.nextDouble();

  switch (operator) {


   case 1:
    result = number1 + number2;
    System.out.println(number1 + " + " + number2 + " = " + result);
    break;


   case 2:
    result = number1 - number2;
    System.out.println(number1 + " - " + number2 + " = " + result);
    break;


   case 3:
    result = number1 * number2;
    System.out.println(number1 + " * " + number2 + " = " + result);
    break;


   case 4:
    result = number1 / number2;
    System.out.println(number1 + " / " + number2 + " = " + result);
    break;

   default:
    System.out.println("Invalid number!");
    break;
  }


 }
}
```

Bubble sort.

```
import java.util.Scanner;
```

```
public class BubbleSort{

        static void bubbleSort(int[] arr) {
      int n = arr.length;
      int temp = 0;
       for(int i=0; i < n; i++){
              for(int j=1; j < (n-i); j++){
                     if(arr[j-1] > arr[j]){

                           temp = arr[j-1];
                           arr[j-1] = arr[j];
                           arr[j] = temp;
                     }

              }
       }

   }
   public static void main(String[] args) {
              Sacanner scan = new Scanner(System.in);
                                  int arr[] = new int[5];
              System.out.println("Enter the number");
                                  for(int i=0; i < arr.length; i++){
                    arr[i] = scan.nextInt();
              }

              System.out.println("Array Before Bubble Sort");
              for(int i=0; i < arr.length; i++){
                    System.out.print(arr[i] + " ");
              }
              System.out.println();

              bubbleSort(arr);

              System.out.println("Array After Bubble Sort");
              for(int i=0; i < arr.length; i++){
                    System.out.print(arr[i] + " ");
              }

      }
}
```

Find the index and count how many time a number is repeated in an array.

```
import java.util.Scanner;
public class Main
{

   static int count(int arr[], int n, int x)
   {
      int res = 0, index=0, num = 0;
      for (int i=0; i<n; i++){
         if (arr[i] == arr[i+1])
          num=arr[i];
                    index = i;
                    res++;
          }
      System.out.prinln(num+"has repeated"+res+"times it's last position is"+index);
   }

   public static void main(String args[])
   {
      int arr[] = new int[10];
                Scanner scan = new Scanner(System.in);
                System.out.println("Enter the numbers");
                for (int i=0; i<n; i++){
         arr[i]=scan.nextInt();
          }
                System.out.println();
      count(arr, arr.length, x));
   }
}
```