DAY-33

28 July 2023 15:38

CONSTRUCTOR CHAINING

WHAT IS CONSTRUCTOR CHAINING ?

- THE PROCESS OF CALLING CONSTRUCTOR OF ONE CLASS FROM CONSTRCUTOR OF ANOTHER CLASS IS CALLED CONSTRUCTOR CHAINING
- FOR CONSTRUCTOR CHAINING PROCESS **IS-A RELATIONSHIP** IS MANDATORY
- THE CONSTRCUTOR CHAINING PROCESS CAN HAPPENS BOTH IMPLICITLY AS WELL AS EXPLICITLY
- IMPLICITLY IT IS DONE BY **JVM** AND EXPLICITLY IT HAS TO BE DONE BY USER
- WHENEVER, THE SUPER CLASS CONTAINS ARGUMENTED CONSTRUCTOR, THE USER HAS TO EXPLICITLY PERFROM CONSTRUCTOR CALLING, BUT IF SUPER CLASS CONTAINS NO ARGUMENTED CONSTRUCTOR, THEN THE CONSTRUCTOR CHAINING WILL TAKE PLACE IMPLICITILY , BECAUSE BY DEFAULT THE FIRST STATEMENT WILL CONSTRUCTOR CHAINING STATEMENT SUPER().
- WE CAN PERFROM CONSTRUTOR CHAINING PROCESS BY MAKING USE OF SUPER()
- SUPER() AND THIS() ALWAYS E FIRST STATEMENT OF CONSTRUCTOR BODY

CASE -1 : EXAMPLE OF EXPLICIT CONSTRUCTOR CHAINING

```
class Test1
{

	int a = 5;

	public Test1()
	{
		System.out.println("inside parent constructor");
	}

}

class Test2 extends Test1
{
	public Test2()
	{
		System.out.println("inside child constructor");
	}
	public static void main(String[] args)
	{
		Test2 a1 = new Test2();
	}

}
```

NOTE : IF WE DON'T DECLARE WITH SUPER CALLING METHOD IT WILL AUTOMATICALLY TAKE THE PARENT CLASS

CASE-2 : IF CHILD IS NOT NON-PARAMEMTERIZED AND PARENT CLASS IS PARAMETERIZED

```
class Test1
{



	public Test1(int a)
	{
		System.out.println("inside parent constructor");
	}

}

class Test2 extends Test1
{
	public Test2()
	{
		System.out.println("inside child constructor");
	}
	public static void main(String[] args)
	{
		Test2 a1 = new Test2();
	}

}
```

OUTPUT : ERROR

CASE - 3 :

IF CHILD DOES NOT HAVE ANY CONSTRCUTOR AND PARENT CLASS HAVE DEFAULT CONSTUCTOR

```
class Test1
{



	public Test1()
	{
		System.out.println("inside parent constructor");
	}

}

class Test2 extends Test1
{

	public static void main(String[] args)
	{
		Test2 a1 = new Test2();
```

```
	}

}
```

OUTPUT : INSIDE PARENT CONSTRUCTOR

CASE - 4

IF CHILD CONSTRCTOR IS PARAMTERIZED

```
class Test1
{



	public Test1()
	{
		System.out.println("inside parent constructor");
	}

}

class Test2 extends Test1
{

	public Test2(int a)
	{
		System.out.println("inside child constructor "+a);
	}

	public static void main(String[] args)
	{
		Test2 a1 = new Test2(10);
	}

}
```

OUTPUT :

```
inside parent constructor
inside child constructor 10
```

EXPLICIT CHAINING

```
class Test1
{
```

```
    int x;

    public Test1(int a)
    {
        x = a;
        System.out.println("The value of x is "+x);
        System.out.println("inside parent constructor");
    }

}

class Test2 extends Test1
{

    double b;

    public Test2(double c)
    {
        super(10);
        b = c;
        System.out.println("The value of b is "+b);
        System.out.println("inside child constructor ");
    }

    public static void main(String[] args)
    {
        Test2 a1 = new Test2(100.42);
    }

}
```

WHY MULTIPLE INHERITANCE IS NOT SUPPORTED IN JAVA AT CLASS LEVEL ?

- WHENEVER WE CREATE AN OBJECT OF CHILD CLASS, SUPER CLASS CONSTRUCTOR HAS TO BE CALLED WITHER EXPLICITY OR IMPLICITLY BY JVM TO COMPLETE CONSTRUCTOR CHAINING PROCESS.
- IN MULTIPLE INHERITANCE, ONE CHILD CLASS WILL HAVE MULTIPLE SUPER CLASS.
- WHENVER WE CREATE AN OBJECT OF CHILD CLASS, IT WILL LEAD TO CONFUSION THAT WHICH SUPER CONSTRCUTOR TO BE CALLED FIRST AND EXECUTE FIRST AND HENCE, CONSTRCUTOR CHAINING PROCESS WILL REMAIN INCOMPLETE .
- SINCE, CONSTRCUTOR CHAINING PROCESS IS MANDATORY IN **IS-A RELATIONSHIP** EITHER IMPLICITLY OR EXPLICITLY AND SINCE, CONSTRCUTOR CHAINING REMAINS INCOMPLETE IN MULTIPLE INHERITANCE , SO MULTIPLE INHERITANCE IS NOT SUPPORTED IN JAVA

METHOD OVERRIDING

WHAT IS METHOD OVERRIDING ?

- WHENVER THE SUBCLASS WANTS TO CHANGE THE IMPLMENTATION OF SUPER CLASS METHOD, IT CAN CHANGE THE IMPLEMENTATION OF THAT PARTICULAT METHOD FOR ITSLEF
- THIS PROPERTY OF CHANGING THE SUPER CLASS METHOD IMPLEMENTATION IN SUBCLASS IS KNOWN AS METHOD OVERRIDING
- FOR METHOD OVERRIDING TO TAKE PLACE, THE CLASSES MUST HAVE **IS-A RELATION** AMONG THEM

```
class Test1
{
	public void dream()
	{
		System.out.println("i want my son should be musician");
	}


}

class Test2 extends Test1
{
	public void dream()
	{
		System.out.println("i want to become world class palyer");
	}


	public static void main(String[] args)
	{
		Test2 a1 = new Test2();
		a1.dream();

		Test1 a2 = new Test1();
		a2.dream();
	}

}
```

NOTE : 1

1. IF WE DECLARE A METHOD AS A FINAL , WE CAN NOT OVERRIDE IT, i.e, IF A METHOD DECLARED AS FINAL, WE CAN NOT CHANGE ITS IMPLEMENTATION.

```
class Test1
{
	final public void m1()
	{
		System.out.println("Hi");
	}


}

class Test2 extends Test1
{
	public void m1()
	{
		System.out.println("Bye");
	}


	public static void main(String[] args)
```

```
	{
		Test2 a1 = new Test2();
		a1.m1();

		Test1 a2 = new Test1();
		a2.m1();
	}

}
```

//cannot change implementation of final method

NOTE : 2

WHILE OVERRIDING , WE CAN NOT CHANGE RETURN TYPE(WITH RESPECT TO PRIMITIVE DATATYPE)

```
class Test1
{
	 public void vehicle()
	{
		System.out.println("car");
	}


}

class Test2 extends Test1
{
	public int vehicle()
	{
		System.out.println("bike");
	}


	public static void main(String[] args)
	{
		Test2 a1 = new Test2();
		a1.vehicle();


	}

}
```

CONCLUSIONS

- WE CAN NOT OVERRIDE STATIC MEMBER FUNCTIONS BECAUSE IN JAVA OVERRIDING CONCEPT IS SUPPORTED ONLY FOR OBEJCT LEVEL MEMBERS(NON STATIC MEMBER FUNCTIONS) NOT CLASS LEVEL MEMBERS
- OVERRIDING IS ONLY FOR MEMBER FUNCTIONS OF A CLASS NOT THE DATA MEMBERS

***** DIFFERENCE BETWEEN METHOD OVERLOADING AND METHOD OVERRIDING

METHOD OVERLOADING	METHOD OVERRIDING

TAKES PLACE WITHIN SAME CLASS	HAPPENS BETWEEN PARENT AND CHILD CLASS (**IS-A** RELATIONSHIP IS MANDATORY)
METHOD NAME SHOULD BE SAME BUT WITH DIFFERENT ARGUMENTS	BOTH METHOD NAME AND ARGUMENTS SHOULD BE SAME, IMPLEMENTATION SHOULD BE DIFFERENT
IS - A RELATIONSHIP IS NOT MANDATORY	IS-A RELATIONSHIP IS MANDATORY
STATIC METHODS CAN OVERLOADED	WE CAN NOT OVERRIDE STATIC METHODS
RETRUNG TYPE CAN BE CHANGED	RETURN TYPE CAN NOT BE CHANGED

QUESTIONS :

- WRITE A PROGRAM ON EMTHOD OVERRIDING IMPLEMENTING SEAONS AND CLIMATES
- WRITE A PROGRAM ON FETCHING THE ACCOUNT HOLDER NAME USING THE OVERRIDING CONCEPT