DAY-30

26 July 2023 15:37

NON-PARAMETERISED CONSTRUCTOR

```
class Test1
{
	int x;
	double y;

	public Test1()
	{
		x = 10;
		y = 20.3;
	}

	public static void main(String[] args)
	{

		Test1 a1 = new Test1();

		System.out.println(a1.x);
		System.out.println(a1.y);

	}

}
```

ARGUMENTED CONSTRUCTOR/PARAMETERIZED CONSTRUCTORS

```
class Test1
{
	int x;

	public Test1(int a)
	{
		x = a;

	}

	public static void main(String[] args)
	{

		Test1 a1 = new Test1(10);

		System.out.println(a1.x);

	}
```

```
}



class Test1
{
	int x;
	double y;

	public Test1(int a, double b)
	{
		x = a;
		y = b;

	}

	public static void main(String[] args)
	{

		Test1 a1 = new Test1(10, 5.3);

		System.out.println(a1.x);
		System.out.println(a1.y);

		Test1 a2 = new Test1(20, 23.14);
		System.out.println(a2.x);
		System.out.println(a2.y);


	}

}
```

- **TASK**

WRITE A PROGRAM ON EMPLOYEE DATABASE (4 RECORDS) USING THE CONSTRUCTOR

CONSTRUCTOR LOADING /OVERLOADING

- HAVING MULTIPLE CONSTRUCTORS WITHIN THE SAME CLASS WHICH DIFFERES IN THE ARGUMENTS IS CALLED CONSTRUCTOR OVERLOADING.
- THE ARGUMENTS SHOULD DIFFER IN EITHER OF 3 WAYS
    - LENGTH OF ARGUMENTS
    - TYPE OF ARGUMENTS
    - ORDER OF OCCURANCE OF ARGUMENTS
- CONSTRUCTOR OVERLOADING HELP US IN CREATING OBEJCTS WITH DIFFERENT NUMBER OF ARGUMENTS, TYPES OF ARGUMENTS AND DIFFERENCT ORDER OF ARGUMENTS.

```
class Test1
{
```

```
	int x;
	double y;

	public Test1()
	{
		x = 5;
		y = 10.4;
	}

	public Test1(int a, double b)
	{
		x = a;
		y = b;

	}

	public Test1(double b, int a)
	{
		x = a;
		y = b;

	}

	public Test1(double b)
	{
		y = b;

	}

	public static void main(String[] args)
	{

		Test1 a1 = new Test1();
		System.out.println(a1.x);
		System.out.println(a1.y);

		Test1 a2 = new Test1(7, 3.14);
		System.out.println(a2.x);
		System.out.println(a2.y);

		Test1 a3 = new Test1(9.18, 911);
		System.out.println(a3.x);
		System.out.println(a3.y);


		Test1 a4 = new Test1(11.22);
		System.out.println(a4.x);
		System.out.println(a4.y);


	}

}
```

INTERVIEW QUESTION :

1. WRITE A PROGRAM ON STUDENT DATABASE USING CONSTRUCTOR

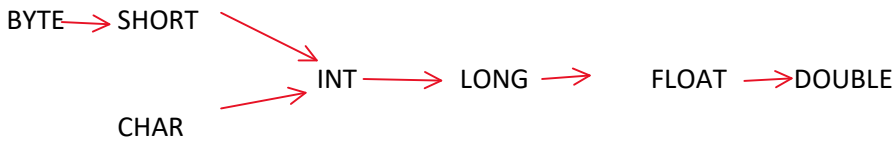
1. NAME, ID, MARKS, NUMBER
2. NAME, ID , MARKS
3. NAME, ID
4. NAME

HAVING MULTIPLE CONSTRUCTORS WHICH CAN DIFFERES IN THE ARGUMENTS/PARAMETERS WHICH DIFFERE IN LENGTH OF ARGUMENT , IN ORDER OF ARGUMENTS, IN TYPE OF ARGUMENTS

2. WRITE A PROGRAM ON EMPLOYEE DATABASE USING CONSTRUCTOR
    a. EMPLOYEE NAME
    b. ID
    c. DESIGNATION
    d. SALARY

TYPE OF CONVERSION

- TYPE CONVERSION IS POSSIBLE FROM LOWER DATA TYPE TO HIGHER DATA TYPE BUT VICE-VERSA IS NOT POSSIBLE



METHOD OVERLOADING

WHAT IS METHOD OVERLOADING ?

- WRITING MULTIPLE METHOD WITH SAME NAME BUT, DIFFERENCE IN EITHER LENGTH OF ARGUMENT/TYPE OF ARGUMENT/ ORDER OF OCCURRENCE OF ARGUMENTS IS CALLED AS **METHOD OVERLOADING**
- WE CAN OVERLOAD BOTH STATIC AS WELL AS NON STATIC METHODS
- COMPILER WILL BIND METHOD DECLARATION WITH METHOD DEFENITION BY LOOKING AT ONE OF THE FOLLOWING CRITERIA :
    - BASED OF NUMBER OF ARGUMENTS
    - BASED ON TYPE OF ARGUMENTS
    - BASED ON ORDER OF OCCURRENCE OF ARGUMENTS

```
class Test1
{
```

```
	public static void add()
	{
		int x = 5, y = 4;

		int c = x+y;

		System.out.println(c);
	}

	public static void add(int a, int b)
	{
		int c = a+b;

		System.out.println(c);
	}

	public static void add(int a, float b)
	{
		float c = a+b;

		System.out.println(c);
	}
	public static void main(String[] args)
	{
		add();
		add(7, 4);
		add(9, 11.3f);

	}

}
```

- WE CAN ALSO DO METHOD OVERLOADING FOR NON STAITC METHODS
- WE HAVE DIFFERENT RETURN TYPE FOR EACH METHOD IN METHOD OVERLOADING

EXAMPLE OF MAIN METHOD OVERLOADING

```
class Test1
{


	public static void main(String[] args)
	{
			System.out.println("BitsQ");
			main("java");
			main(1);
	}

	public static void main(String args)
	{
```

```
        System.out.println("in String method "+ args);

    }

    public static void main(int a)
    {
        System.out.println("in int method "+ a);
    }



}
```

ADVANTAGES OF METHOD OVERLOADING

- REDUCES COMPLEXITY FOR USER AS USER DON'T HAVE TO REMEMBER DIFFERENT METHOD NAME
- INCREASES READABILITY OF PROGRAM

CLASS DIAGRAM

IT IS A PICTORIAL REPRESENTATION OF OBEJCTS INSIDE THE CLASS

```
class Test1
{
    String name;
    int id;
    String department;

    public Test1(String a, int b, String c)
    {
        name =a ;
        id = b;
        department = c;
    }

    Public String getName()
    {

    }

    public int getId()
    {

    }



}
```