Pg1

Wednesday, May 27, 2020 1:15 PM

Non-static Members

Any member declared in the class, and it is not prefixed with static modifier is known as non-static member.

1. non static variables
2. non static methods
3. non static initializers
4. constructors

Note :

1. the non-static members get their memory allocation only inside the instance(object) of the class.
2. We cannot use non-static members without creating an object.

Pg2
Wednesday, May 27, 2020 1:23 PM

<u>**non static variable :**</u>

a variable declared in the class without prefixed with static modifier is known as non static variable.

example :

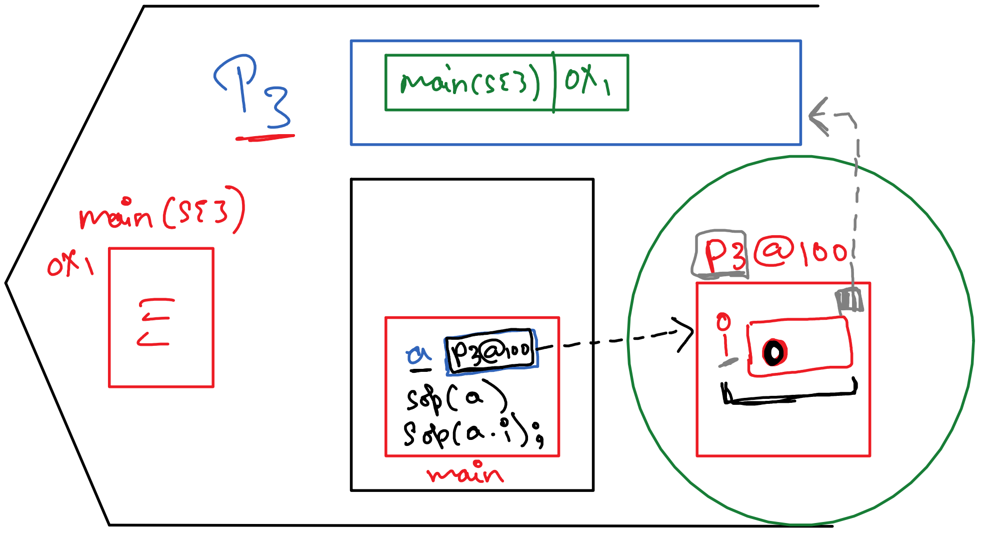
```
class Student
{
    int sid ; } -> non-static variable ,
}
```

Note :
1. it will be assigned with default values.
2. memory will be allocated inside the object of the class.
3. we can use non-static variable with the help of Object reference.
4. non static variable will have one copy in every object of the class.

Example : java P3

```
class P3
{
    int i ;

    public static void main(String[] args)
    {
        P3  a = new P3()  ;
        System.out.println( a );
        System.out.println(  a . i  ) ;
    }
}
```

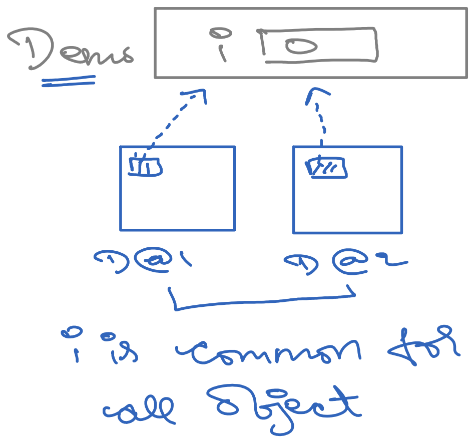
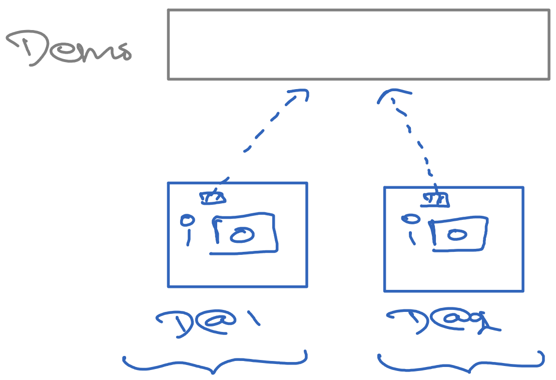


Note :

1. We can use static members with the help of object reference.
   example refer : **app9/non_static_var/P5.java**
2. We cannot use a non-static variable with the help of class name.

Difference between static variable and non-static variable :

static variable	non static variable
1. static variable should be prefixed with static keyword/modifier	non static variable should not be prefixed with static modifier
2. static variable will be allocated in the class static area	non static variable will be allocated in the instance of the class(Object)
3. it is also known as class variable	is also known as instance variable/ object variable
4. we can use a static variable with the help of class name	We cannot use a non-static variable with the help of class name

5. static variable is a shared memory for all the objects of the class. (it is common for all the objects)
 	
 ex:
 class Demo {
     static int i ;
 }
 	
 new Demo() ;
 new Demo() ;
 	
 
	non static variable is a independent (separate) memory for every object of the class. (it is not common)
	
	 ex:
	 class Demo {
	     int i ;
	 }
	
	 new Demo() ;
	 new Demo() ;
	
	 

6. We can use a static variable	we cannot use a non-static variable if

without creating an object	object is not created.

Pg3

Friday, May 29, 2020 1:09 PM

Non static methods :

1. the method which is not prefixed with a static modifier is a non static method.
2. the reference of the non static method will be stored in the object (instance) of the class.
3. We can call a non static method from a static context only with the help of Object reference.
4. The block of non-static method is considered as a non-static context.
5. Every non-static context will have a reference to its object area.
6. We can use a non-static member of a same class directly inside non static context.

this :

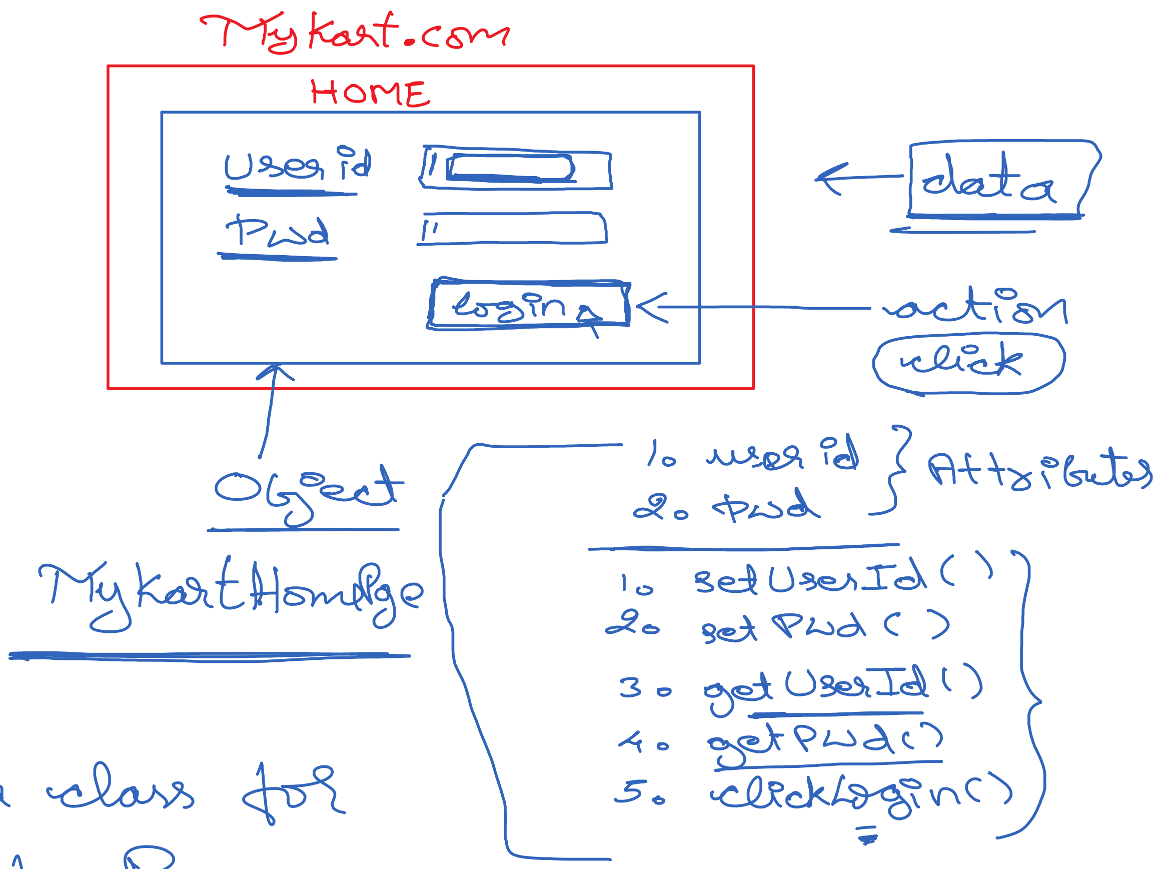
1. it is a keyword.
2. it is a non-static variable, which will have the address of the currently used object.
3. **this** can be used only in non-static context.

Why do we need **this** ?

1. if the local variable and the object variable is having same name, we can use the object variable only with the help of **this variable**.

```
class Demo2
{
	void printReference()
	{
		System.out.println("this : " + this ) ;
	}
	public static void main(String[] args)
	{
		Demo2 obj1  = new Demo2()  ;
		System.out.println("obj1 : " + obj1 ) ;
		obj1.printReference()  ;
	}
}
```

Assignment :



-> Design a class for MyKartHomePage

-> test the class in a driver class.

Pg4

Monday, June 1, 2020 1:30 PM

Non-static initializers :

1. declare and initialization statement :
2. non-static block (Instance Initializer Block) /IIB

1. declare and initialization statement :

syntax :

non-static initializer

```
class Name {
    datatype variable  =  value / expression ;
}
```

Declaration

2. non-static Block / IIB (Instance Initializer Block)

syntax:

```
class Name {
    {
        statements ;
    }c
}
```

→ Non-static Block

Note :

1. non-static block doesn't have a name.

2. non-static block doesn't have a return type.
3. it cannot accept any arguments.
4. non-static initializer gets executed implicitly during the loading process of an Object of the class.
5. non-static initializer gets executed only once for each object created.

Example : refer - **qcca3/jp/app11**

Why do we need Non-static initializers ? / When do we use Non-static initializers ?

Answer :

Any action to be performed mandatorily for every object created, such instructions can be written using non-static Initializer.

For example : refer, **qcca3/jp/app11/Baby.java, BabyDriver1.java**

Questions :

1. What is the difference between non-static method and non-static initializer ?
2. What is the difference between static block and non-static block? Justify with program.

Pg5

Tuesday, June 2, 2020 1:22 PM

Constructors :

1. it is a special member(special method) of a class.
2. it is non-static member. (it cannot be static)
3. The name of the constructor is always same as the class name.
4. they do not have a return type.
5. It is mandatory that, a class must have a constructor.
   (if the programmer fails to create a constructor, a constructor will be added automatically into the class by the compiler at the time of compilation)

syntax To Create a constructor :

```
[ modifiers ]  className( [ Formal arguments , ...]  )
{
        // Statements  ;
}
```

Constructor Signature

Constructor Definition

In java, we can classify constructors into 2 types :

1. no-argument constructor :

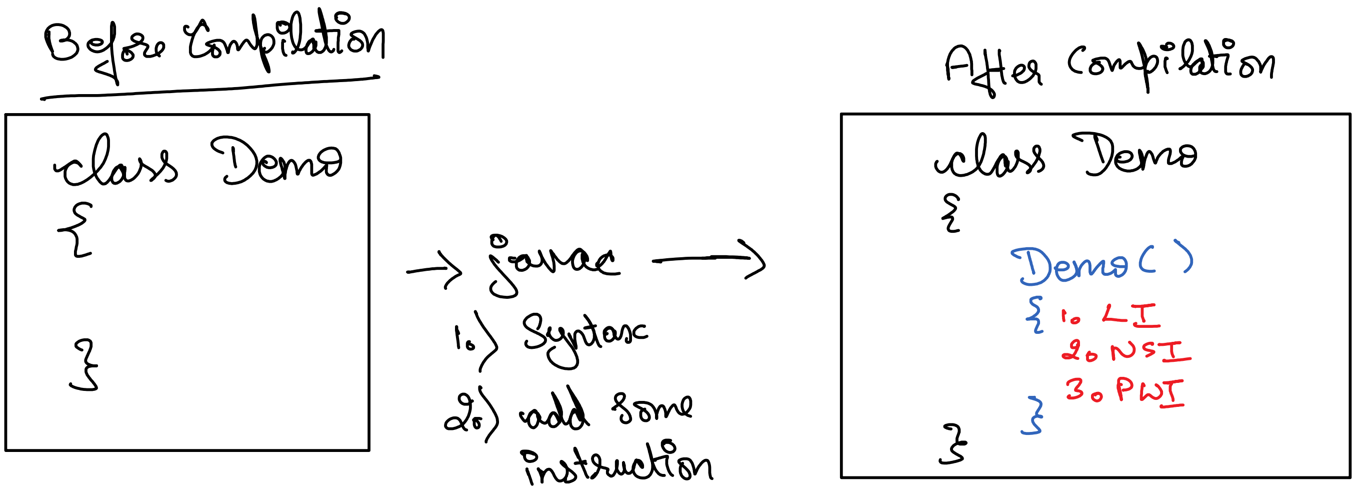
   Constructor which does not accept any arguments is known as no-argument constructor.

2. parameterized constructor

   Constructor which can accept arguments(data while call) is known as parameterized constructors.

Note :

It is mandatory that, a class must have a constructor.
if the **programmer fails to create a constructor**, a no-argument constructor will be added automatically into the class by the compiler at the time of compilation



Why do we need a Constructor ?
Ans.

1. It is used to load non-static members of that class into the instance of the class(Object of the class)
2. it also initializes the non-static members of the object.

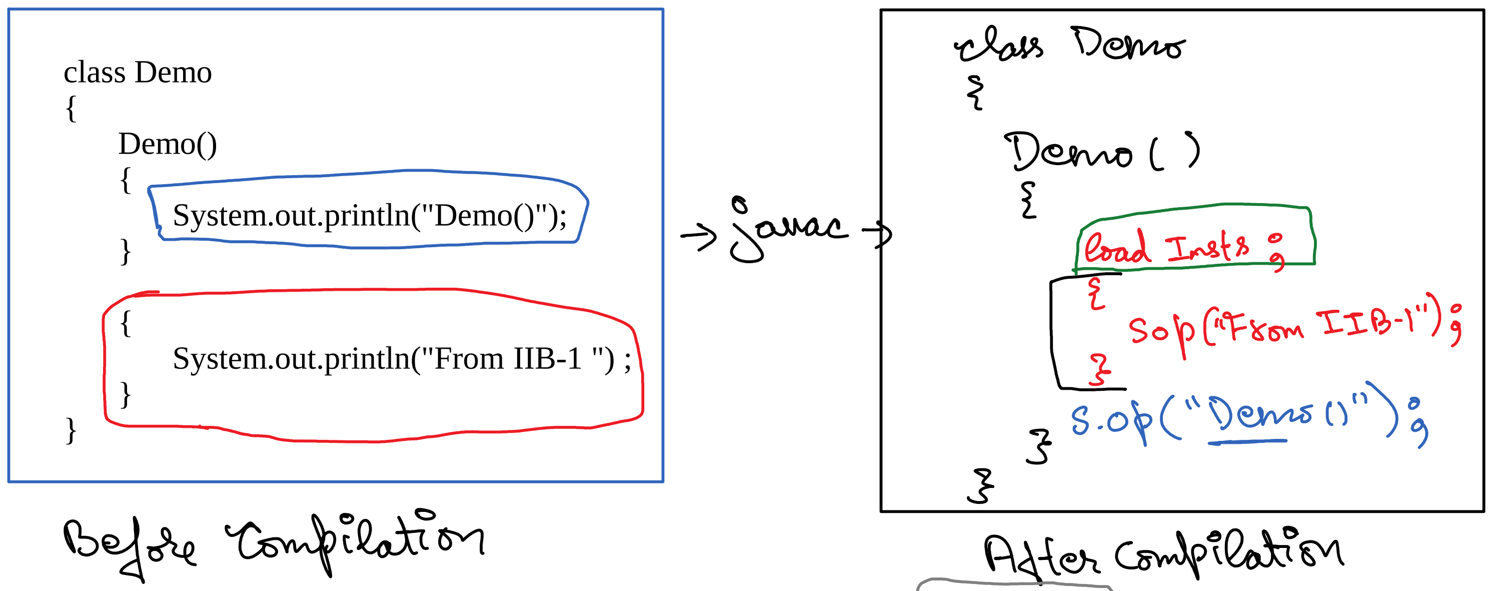
Note :

1. in every constructor body compiler adds few instructions :

```
[ modifiers ] className()
{
    //1. instructions which help to load non-static members
    // 2. Non-static initializers of the class will be added into the
    constructor body.
    // 3. Instructions written by the programmer

}
```

Example :

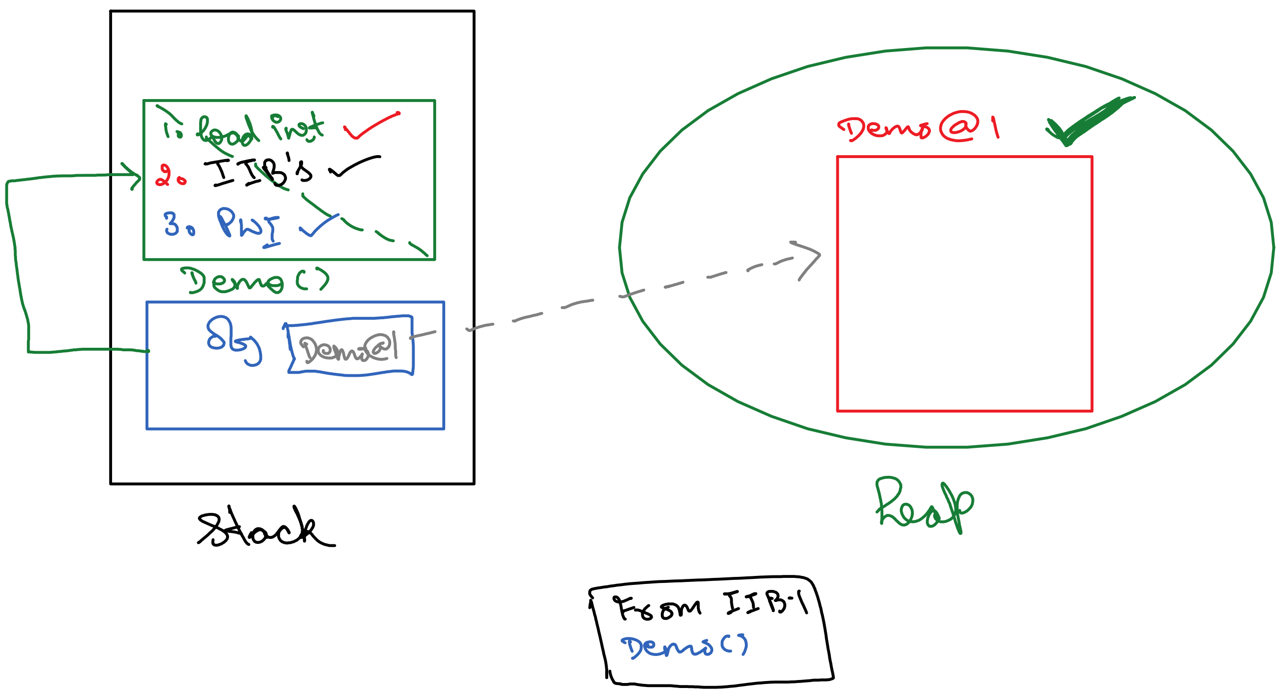


```
class Demo
{
	Demo()
	{
		System.out.println("Demo()");
	}

	{
		System.out.println("From IIB-1 ") ;
	}
}
```

```
class Demo
{
   Demo ( )
   {
      load Insts ;
      {
         Sop ("From IIB-1");
      }
      S.op ("Demo()");
   }
}
```

Demo obj = new Demo();



Object (Instance) loading process :

1. new creates object area in the heap
2. Constructor is called and executed :
    1. Load instructions :
        a. if non static variable is present, it will be allocated with default values assigned.
        b. if non-static method is present, it's reference along with

signature is stored in the object.

   2. if Non-static initializers present, they get executed.
   3. if programmer written instructions of constructer present, they get executed.

3. The loading process of the Object is completed, new operator gives the reference of the Object created.

Pg6

Wednesday, June 3, 2020 1:12 PM

Parameterized constructor :

Constructor created with formal arguments declared is known as parameterized constructors.

syntax :

```
className( datatype var1 , … )
{
        statements ;
}
```

Why do we need a parameterized constructor ?
Ans: it is used to initialize the object variables, at the time of object creation, where the values are passed in the object creation statement.

example refer : **app12/cons2/Laptop.java, Driver1.java**

Constructor Overloading :

A class having more than one constructor it is known as constructor overloading.

Note :

1. The signature of the constructor must be different either by length or type.

Example refer : **app12/cons2/Laptop.java**

Task1 :

Design a class to store the details of the Employee data.
attributes of the employee :

1. name
2. employee ID
3. salary

Note : Design a class such that the user should be able to create object by :

1. assigning no data
2. assigning all the data
3. assigning only name
4. assigning only Employee ID

Note :

1. For one object creation all the non-static initializers of the class get executed.
2. For one object creation only one constructor gets executed.

Pg7

Thursday, June 4, 2020 1:31 PM

Can we execute more than one constructor for one object ?

Yes, we can execute more than one constructor of a class for one object creation, with the help of constructor chaining.

constructor chaining:

one constructor calling another constructor, is known as constructor chaining.

In java we can achieve constructor chaining in two ways :

1. using **this()** ------- > this call statement
2. using **super()** ----> super call statement

this() :

It is used to call another constructor of the same class.

Rules :

1. it should be used only in a constructor.
2. this() should be first statement of the constructor.
3. the constructor should not call itself. (recursion is not allowed)
4. If a class has N constructors, we can use this() call statement only in N-1 number of constructors.

Note :

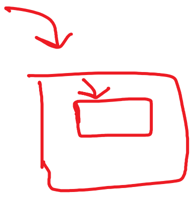
1. if a constructor has this() statement used, for that constructor compiler does not add load instructions and non-static initializers.

case 1 : Before After

Before

```
class A
{
   A()
   { sop("A");
   }

   A(int i)
   {
     sop("A(int)");
   }
}
```



=> javac

new A(10)

After

```
class A
{
    A()
    {
      //load inst
      // NI/IIB's
      sop("A");
    }
    A(int i)  this();
    {
      //load inst
      // IIB's
      sop("A(int)");
    }
}
```

Case 2 :

```
class A
{
    A()
    {
      sop("A()");
    }
    A(int i)
    {
      this();
      sop("A(int)");
    }
}
```

javac

```
class A
{
    A()
    {
      //load instr
      // IIB's
      sop("A()");
    }
    A(int i)
    {
      this();
      sop("A(int)");
    }
}
```

Assignment1

Thursday, June 4, 2020 2:25 PM

```
class A
{
    {
        System.out.println("IIB-1 ") ;
    }
    A()
    {
        this( 10 , 20  )  ;
        System.out.println("From A()  ") ;
    }
    A(int j)
    {
        System.out.println("From A(int )  ") ;
    }
    A(int j , double k )
    {
       this(  10 )  ;
        System.out.println("From A(int , double )  ") ;
    }
}
```

1. Number of constructors executed for the object creation ?
2. Write the flow of constructor call ?
3. Order of execution of constructor
4. Output when the object is created

```
        > new A()  ;
        > new A(10)  ;
        > new A( 10 , 20.3 ) ;
```

==

1. Loading process of a class
2. Loading process of an object

3. Clear knowledge on static members
4. Clear knowledge of non-static members

members	**static context**	**non-static context**
static variables	1. directly (within same class) 2. Class Name 3. Object reference	1. directly (within same class) 2. Class Name 3. this
static methods	1. directly (within same class) 2. Class Name 3. Object reference	1. directly (within same class) 2. Class Name 3. this
non-static variable	1. Object reference	1. directly (within same class) 2. this
non-static methods	1. Object reference	1. directly (within same class) 2. this