Topics

- Use of static keyword in Java
  - ❑ static fields / class variables
  - ❑ static methods / class methods
  - ❑ static classes

Object vs Class Variables and Methods

- Attributes/Fields of a class that are declared using ‘static’ keyword of Java are known Class Variables
- Methods of a class that are declared using ‘static’ keyword of Java are known Class Methods

```
class     XYZ
{
          private   int      x;
          private   double   y;

          private   static   int      z;
          public    static   int      a;

          public    void     doS() { … }
          public    static   int      doS1()   { …}

}// End of class
```

Object Variables / Instance Fields

class Fields/ Variables

Object Method

Class Method

Object Variable vs Class Variable

- Object Variables (or Instance Fields) belongs to Objects of the class. Each instance field is allocated a memory space for each object creation of that class.
- Class Variables (or static variables) belongs to the whole class. All objects of the class share a common copy. Static variables are allocated space only once.
- Instance Fields are accessed only via object-references. On the other hand 'static' variables are primarily accessed through class name via syntax <class-name>.<variable-name>. However, static variables can also be accessed through object-reference, but it is rarely done.

Note : Visibility (public, protected etc.) of the object variables and class variables decides the places where they are visible.

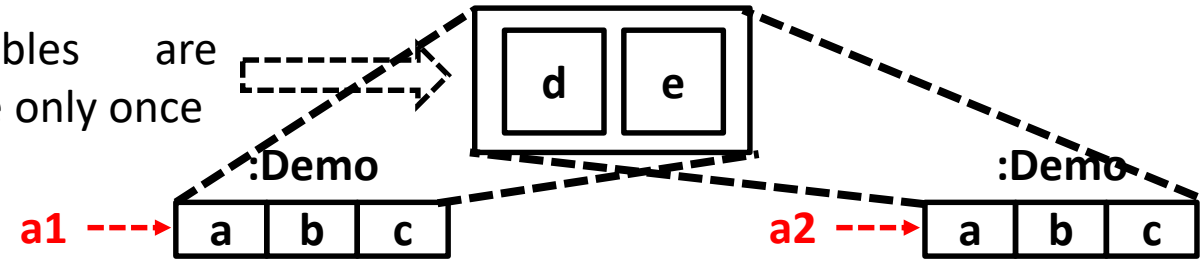
Object vs. Class Variables : Example

- Object Variables are allocated space each time an object is created. Class Variables are allocated space only once and this space is being shared by all the objects of that class.

```
// File Name : Demo.java
class Demo
{
        double      a;                  << Object Variable, Access Modifier : package-private >>
        double      b;                  << Object Variable, Access Modifier : package-private >>
        double      c;                  << Object Variable, Access Modifier : package-private >>
        static      double      d;      << Object Variable, Access Modifier : package-private >>
        static      double      e;      << Object Variable, Access Modifier : package-private >>
}// End of class Demo
// Driver Class
class Test
{
        public      static void main(String args[])
        {
                Demo    a1      =       new Demo();
                Demo    a2      =       new Demo();

        } // End of Method
} // End of class Test
```

static variables are allocate space only once



Accessing Object and Class Variables : Example

- Object Variables are accessed through object-references whereas class variables can be accessed via class name as well as object-reference

```
// File Name : Demo.java
class Demo
{
        double      a;                  << Object Variable, Access Modifier : package-private >>
        double      b;                  << Object Variable, Access Modifier : package-private >>
        double      c;                  << Object Variable, Access Modifier : package-private >>
        static      double      d;      << Object Variable, Access Modifier : package-private >>
        static      double      e;      << Object Variable, Access Modifier : package-private >>
}// End of class Demo
// Driver Class
class Test
{
        public      static void main(String args[])
        {
                Demo    a1      =       new Demo();
                Demo    a2      =       new Demo();

                System.out.println(a1.a);

                System.out.println(Demo.d);

                System.out.println(Demo.e);

                a1.d = 45.67;
                System.out.println(a2.d);

        } // End of Method
} // End of class Test
```

Accessing instance field via ‘a1’

Accessing static fields via class name ‘Demo’

Accessing static field via ‘a1’ and ‘a2’

Instance Methods vs Class Methods

innovate achieve lead

Object Methods (Non-static Methods	Class Methods (static Methods)

Instance Methods vs Class Methods

Object Methods (Non-static Methods	Class Methods (static Methods)
1. Accessible only via Object-Reference Variables	1. Accessible via both Object-Reference Variable as well class name

Instance Methods vs Class Methods

Object Methods (Non-static Methods	Class Methods (static Methods)
1. Accessible only via Object-Reference Variables	1. Accessible via both Object-Reference Variable as well class name
2. Access Syntax	2. Access Syntax
<object-reference>.Method-Name(<parameters>)	<class-name>.Method-Name(<parameters>); or <object-reference>.Method-Name(<parameters>)

Instance Methods vs Class Methods

Object Methods (Non-static Methods	Class Methods (static Methods)
1. Accessible only via Object-Reference Variables	1. Accessible via both Object-Reference Variable as well class name
2. Access Syntax <object-reference>.Method-Name(<parameters>)	2. Access Syntax <class-name>.Method-Name(<parameters>); or <object-reference>.Method-Name(<parameters>)
3. Object Methods can access both static as well as non-static fields	3. Class Methods can only access static fields. Object Fields in a static method are accessible only by creating objects of the class

Accessing Object and Class Methods : Example 1

```
// File Name : Demo.java
class Demo
{
        double      a;
        double      b;
        double      c;
        static      double      d;
        static      double      e;
        public      void        doS()
        {
                    this.a = 30;
                    this.b = 40;
                    c = 50;

                    d = 89.67;
                    e = 67.89;

                    display();
        }// End of Method
        public static void doS1()
        {
                    this.a = 30;
                    this.b = 40;
                    c = 50;

                    d = 89.67;
                    e = 67.89;
        }// End of Method
        public      void        display()
        {
                    System.out.println("a= "+this.a+" b= "+this.b+" c= "+this.c +" d= "+d+" e= "+e);
        } // End of Method
}// End of class Demo
```

Object Method

class Method

Instance-fields accessed in a object-method

class-fields (static) accessed in a object-method

object-method invoked from other object-method

instance-fields accessed in a class-method → Error

class-fields accessed in a class-method → No Error

Accessing Object and Class Methods : Example 1 …

innovate achieve lead

```
// File Name : Demo.java
class Demo
{
        double      a;
        double      b;
        double      c;
        static      double      d;
        static      double      e;
        public      void        doS()
        {
                    this.a = 30;
                    this.b = 40;
                    c = 50;

                    d = 89.67;
                    e = 67.89;

                    display();
        }// End of Method
        public static void doS1()
        {
                    this.a = 30;
                    this.b = 40;
                    c = 50;

                    d = 89.67;
                    e = 67.89;
        }// End of Method
        public      void        display()
        {
                    System.out.println("a= "+this.a+" b= "+this.b+" c= "+this.c +" d= "+d+" e= "+e);
        } // End of Method
}// End of class Demo
```

Object Method

class Method

```
F:\>javac Demo.java
Demo.java:22: non-static variable this cannot be
referenced from a static context
                    this.a = 30;
                    ^
Demo.java:23: non-static variable this cannot be
referenced from a static context
                    this.b = 40;
                    ^
Demo.java:24: non-static variable c cannot be
referenced from a static context
                    c = 50;
                    ^
3 errors
```

Accessing Object and Class Methods : Example 2

innovate achieve lead

```
// File Name : Demo.java
class Demo
{
        double      a;
        double      b;
        double      c;
        static      double      d;
        static      double      e;
        // Object Method-1 doS
        public      void        doS()
        {
                    this.a = 30;
                    this.b = 40;
                    c = 50;
                    d = 89.67;
                    e = 67.89;
                    display();
        }// End of Method
        // Class Method doS1
        public static void doS1()
        {
                    d = 189.67;
                    e = 167.89;
        }// End of Method
```

```
// Object Method-2 display
public      void        display()
{
            System.out.println("a= "+this.a);
            System.out.println("b= "+this.b);
            System.out.println("c= "+this.c);
            System.out.pritnln("d= "+d);
            System.out.println("e= "+e);
}// End of Method

}// End of class Demo

//          Driver Class
class Test
{
  public    static void main(String args[])
  {
    Demo    a1          =           new Demo();

            a1.doS();

            Demo.doS1();

            a1.display();

  }// End of Methods
}// End of Test class
```

Accessing Object and Class Methods : Example 2 …..

- **Output of the code Shown in the Previous Slide**

```
F:\>java Test
a= 30.0
b= 40.0
c= 50.0
d= 89.67
e= 67.89
a= 30.0
b= 40.0
c= 50.0
d= 189.67
e= 167.89
```

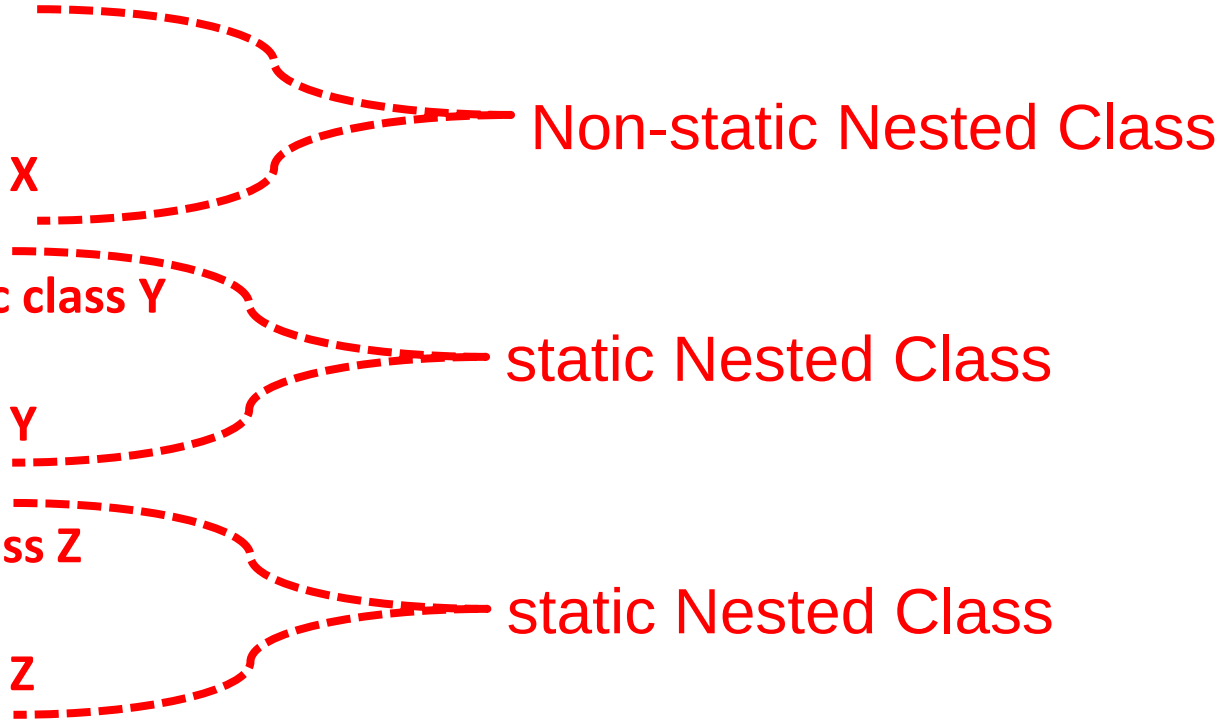
static keyword for classes

- 'static' keyword for classes is used only for nested classes (class inside another class)

```
class XYZ
 {
        private class X
        {
        }// End of class X

        protected static class Y
        {
        }// End of class Y

        public static class Z
        {
        }// End of class Z
}// End of class XYZ
```



Thank You