Object Oriented Programming using Java

Prepared By:
Suyel, PhD
Assistant Professor
Dept. of CSE, NIT Patna

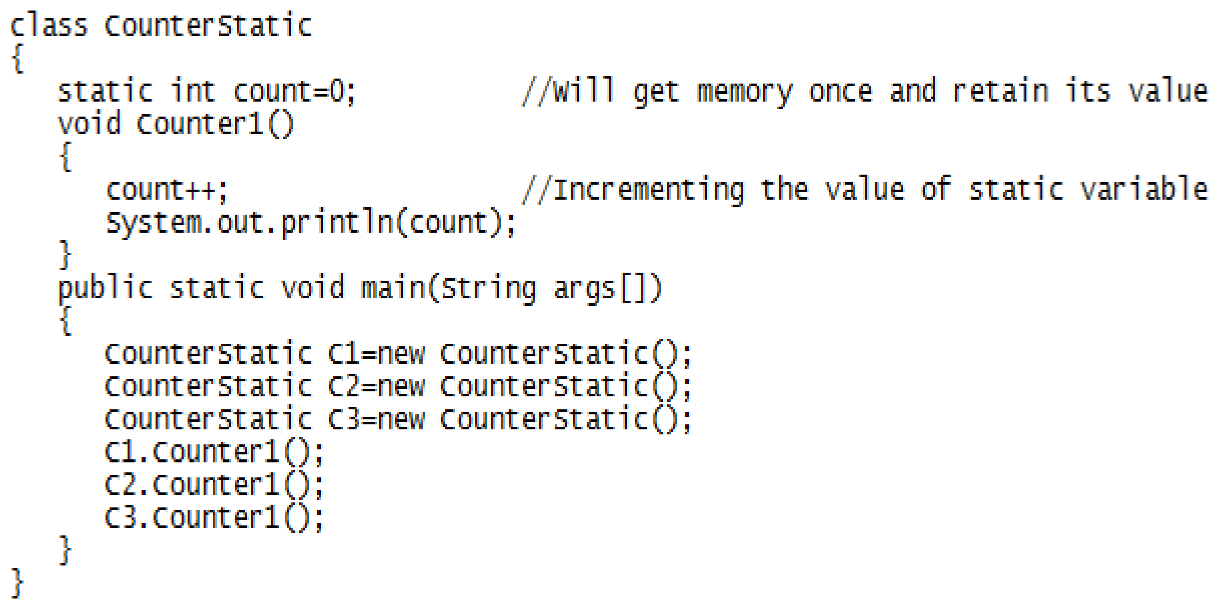
Outline

1. Static Variable
2. Static Method
3. Static Block
4. Static Class

Static Variable

- A static variable is common to all the instances (or objects) of the class because it is a class level variable.
- Only a single copy of static variable is created and shared among all the instances of the class.
- Memory allocation for such variables only happens once, when the class is loaded in the memory.
- Static variables are also known as **class variables**.
- Unlike non-static variables, static variables can be accessed directly in static and non-static methods by the class name and doesn’t need any object.

Static Variable (Cont…)



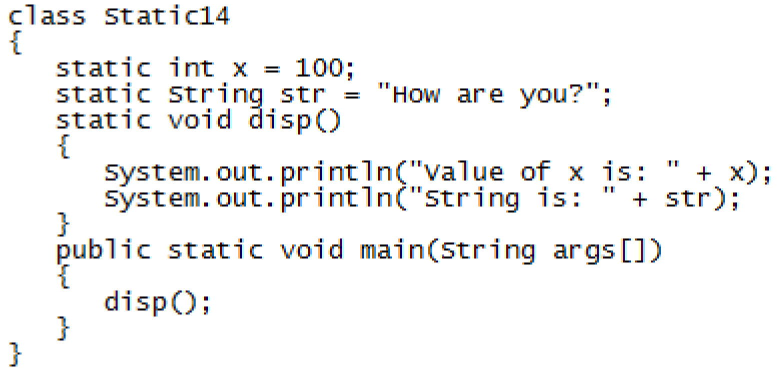
```
class CounterStatic
{
   static int count=0;          //Will get memory once and retain its value
   void Counter1()
   {
      count++;                  //Incrementing the value of static variable
      System.out.println(count);
   }
   public static void main(String args[])
   {
      CounterStatic C1=new CounterStatic();
      CounterStatic C2=new CounterStatic();
      CounterStatic C3=new CounterStatic();
      C1.Counter1();
      C2.Counter1();
      C3.Counter1();
   }
}
```

Static Variable (Cont…)

- Output

```
1
2
3
```

Static Variable (Cont…)



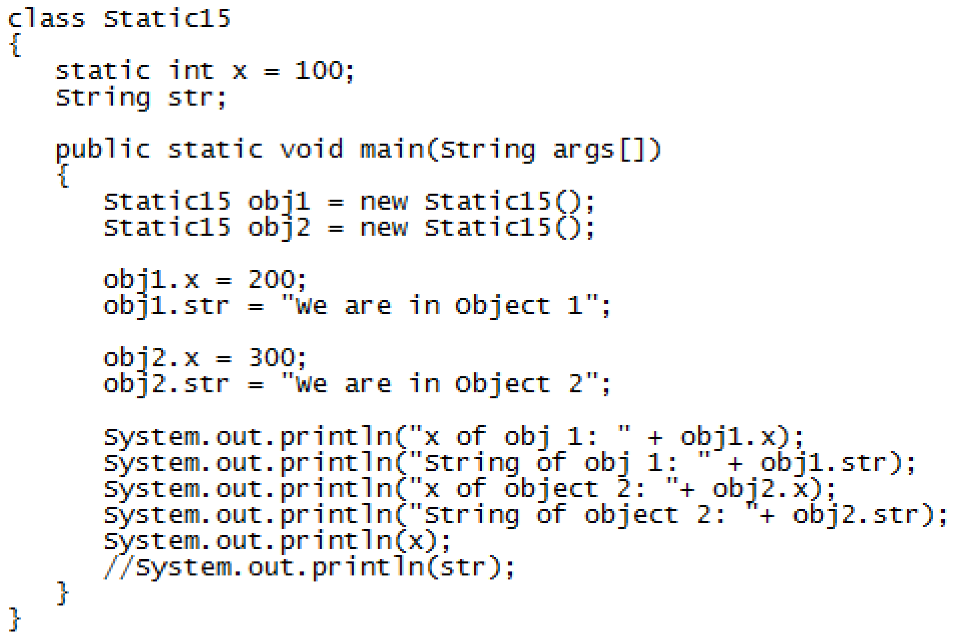
```
class Static14
{
   static int x = 100;
   static String str = "How are you?";
   static void disp()
   {
      System.out.println("Value of x is: " + x);
      System.out.println("String is: " + str);
   }
   public static void main(String args[])
   {
      disp();
   }
}
```

Static Variable (Cont…)

- Output

```
Value of x is: 100
String is: How are you?
```

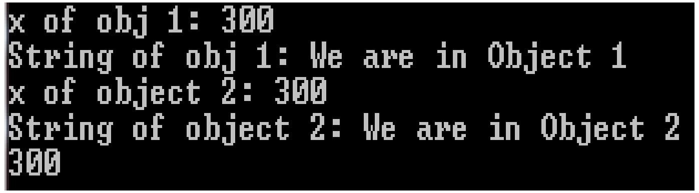
Static Variable (Cont…)



```
class Static15
{
   static int x = 100;
   String str;

   public static void main(String args[])
   {
      Static15 obj1 = new Static15();
      Static15 obj2 = new Static15();

      obj1.x = 200;
      obj1.str = "We are in Object 1";

      obj2.x = 300;
      obj2.str = "We are in Object 2";

      System.out.println("x of obj 1: " + obj1.x);
      System.out.println("String of obj 1: " + obj1.str);
      System.out.println("x of object 2: "+ obj2.x);
      System.out.println("String of object 2: "+ obj2.str);
      System.out.println(x);
      //System.out.println(str);
   }
}
```

Static Variable (Cont…)

- Output



```
x of obj 1: 300
String of obj 1: We are in Object 1
x of object 2: 300
String of object 2: We are in Object 2
300
```

Static Variable (Cont…)

❑ Combination of Variables

Student.java

```
class Student
{
    int rollno;                              //Instance variable
    String name;
    static String college = “ITS”;    //Static variable
    void  Student1(int r, String n)
    {
        rollno = r;
        name = n;
    }
    void display ()
    {
        System.out.println(rollno+" "+name+" "+college);
    }
}
```

Test.java

```
public class Test
{
    public static void main(String args[])
    {
        Student s1 = new Student();
        Student s2 = new Student();
        s1.Student1(111, “Karan”);
        s2.Student1(222, “Aryan”);
        s1.display();
        s2.display();
    }
}
```

Static Variable (Cont…)

- **Combination of Variables (Cont…)**
  - Output

```
111 Karan ITS
222 Aryan ITS
```

Static Variable (Cont…)

- **Combination of Variables (Cont…)**

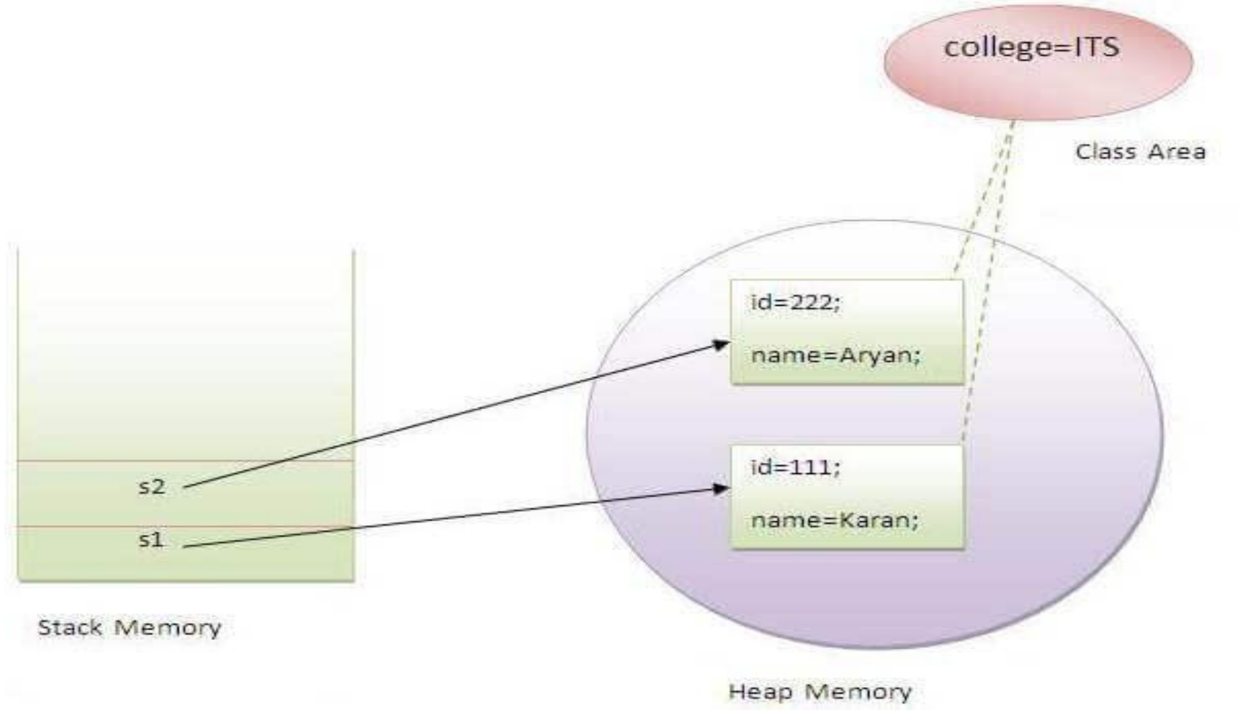


Fig. 1: Memory representation

Static Method

- If we apply static keyword with any method, it is known as static method.
- A static method belongs to the class rather than the object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- Static method can be called or accessed directly using class name.
- A static method can only access static data member inside the static method and can change the value of it.
- It cannot refer to **this** or **super** in any way.

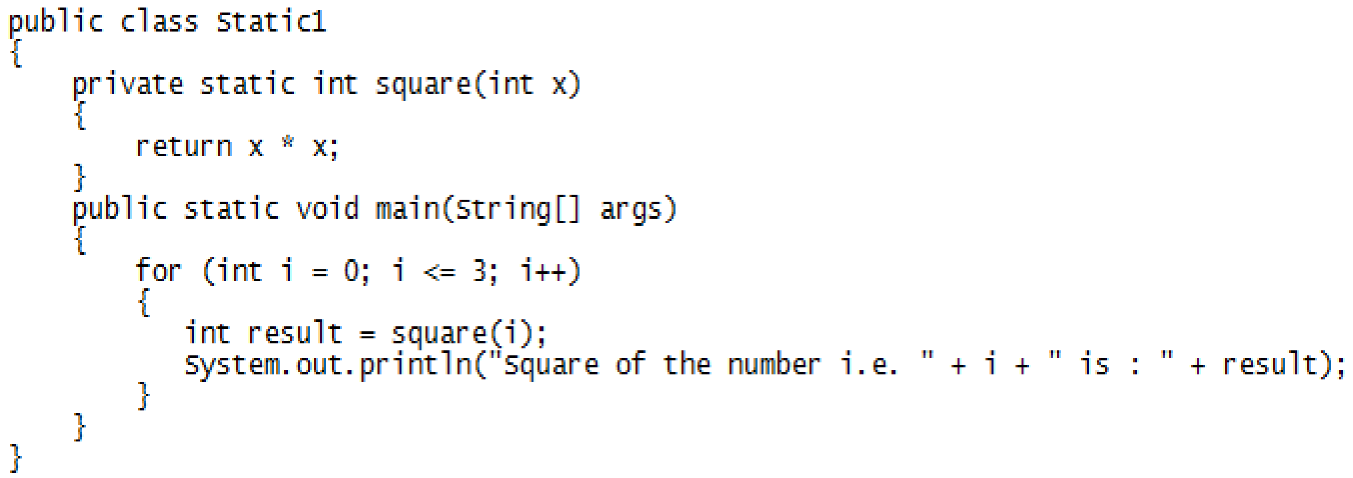
Static Method (Cont…)

- An instance method can call an instance or static method. It can also access instance or static data variable.
- A static method can call a static method only.
- A static method cannot invoke an instance method or access an instance variable.
- We cannot declare a static method and instance method with the same signature in the same class hierarchy.
- When we create a static method in the class, only one copy of the method is created in the memory and shared by all objects of the class.

Static Method (Cont…)

- A static method in java is also loaded into the memory before the object creation.
- Static methods can be overloaded in Java, but cannot be overridden because they are bound with class, not instance.

Static Method (Cont…)



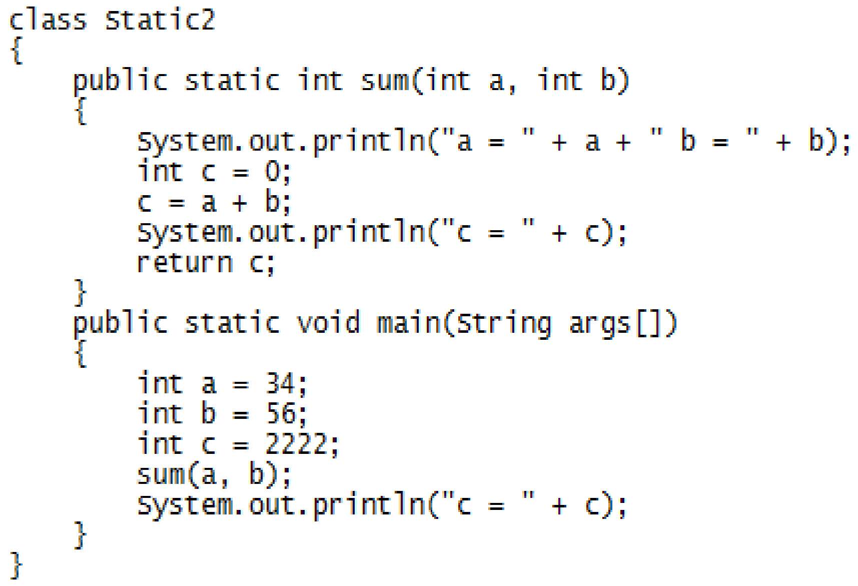
```java
public class Static1
{
    private static int square(int x)
    {
        return x * x;
    }
    public static void main(String[] args)
    {
        for (int i = 0; i <= 3; i++)
        {
            int result = square(i);
            System.out.println("Square of the number i.e. " + i + " is : " + result);
        }
    }
}
```

Static Method (Cont…)

- Output

```
Square of the number i.e. 0 is : 0
Square of the number i.e. 1 is : 1
Square of the number i.e. 2 is : 4
Square of the number i.e. 3 is : 9
```

Static Method (Cont…)



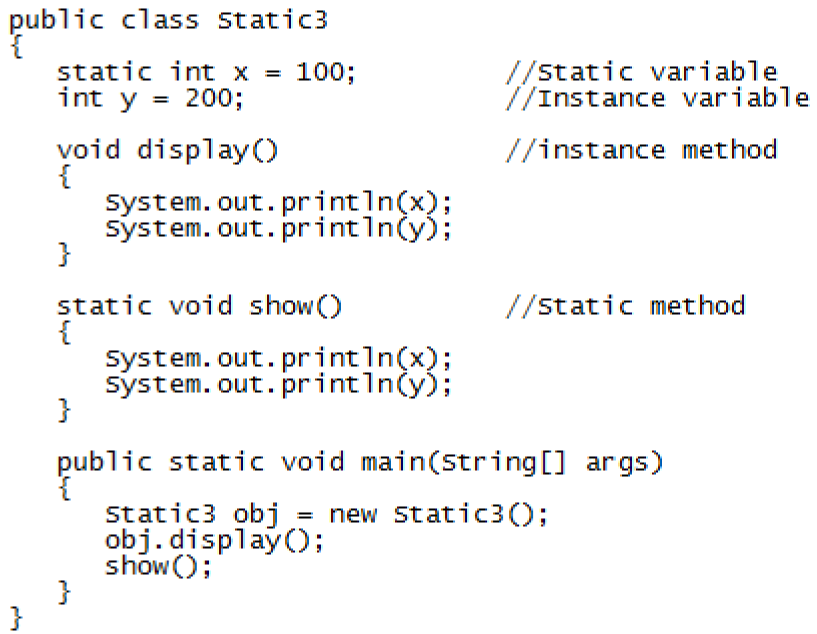
```
class Static2
{
    public static int sum(int a, int b)
    {
        System.out.println("a = " + a + " b = " + b);
        int c = 0;
        c = a + b;
        System.out.println("c = " + c);
        return c;
    }
    public static void main(String args[])
    {
        int a = 34;
        int b = 56;
        int c = 2222;
        sum(a, b);
        System.out.println("c = " + c);
    }
}
```

Static Method (Cont…)

- Output

```
a = 34 b = 56
c = 90
c = 2222
```

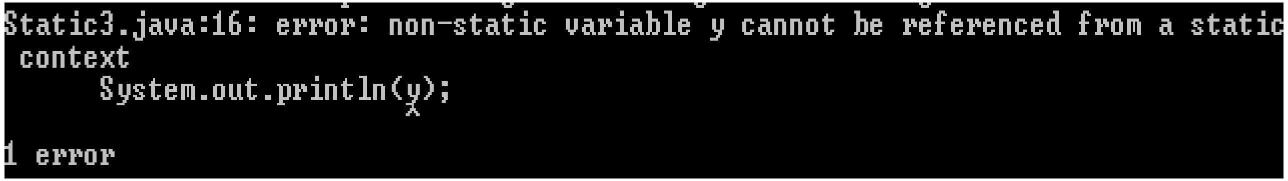
Static Method (Cont…)



```
public class Static3
{
   static int x = 100;         //Static variable
   int y = 200;                //Instance variable

   void display()              //instance method
   {
      System.out.println(x);
      System.out.println(y);
   }

   static void show()          //Static method
   {
      System.out.println(x);
      System.out.println(y);
   }

   public static void main(String[] args)
   {
      Static3 obj = new Static3();
      obj.display();
      show();
   }
}
```

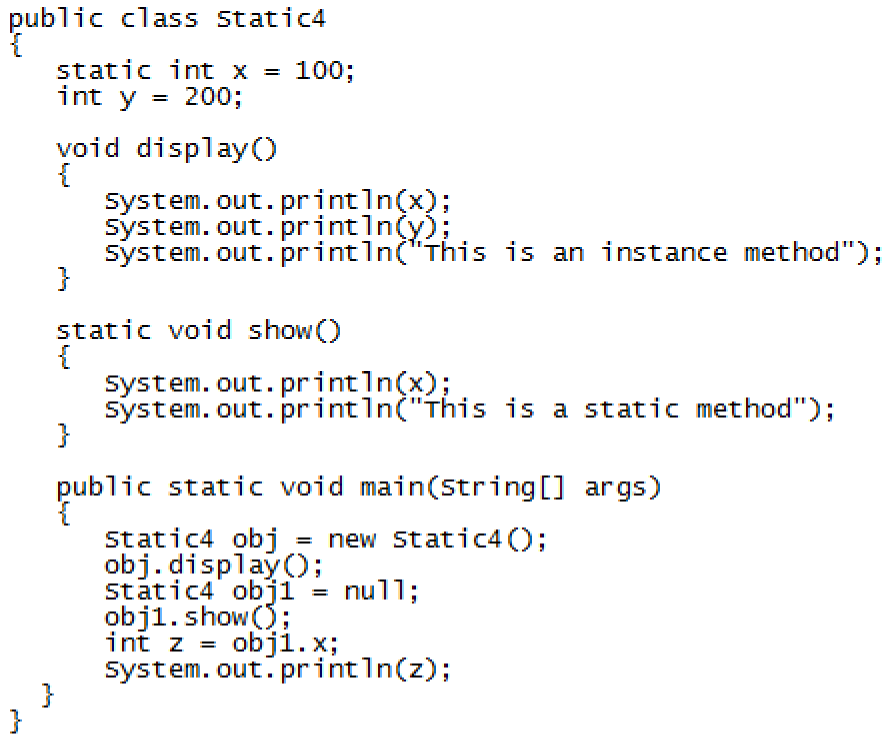
Static Method (Cont…)

- Output



```
Static3.java:16: error: non-static variable y cannot be referenced from a static
 context
        System.out.println(y);
                           ^
1 error
```

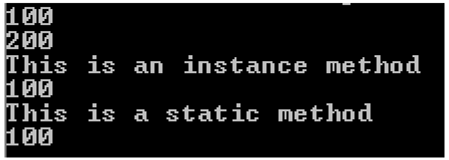
Static Method (Cont…)



```
public class Static4
{
   static int x = 100;
   int y = 200;

   void display()
   {
      System.out.println(x);
      System.out.println(y);
      System.out.println("This is an instance method");
   }

   static void show()
   {
      System.out.println(x);
      System.out.println("This is a static method");
   }

   public static void main(String[] args)
   {
      Static4 obj = new Static4();
      obj.display();
      Static4 obj1 = null;
      obj1.show();
      int z = obj1.x;
      System.out.println(z);
  }
}
```

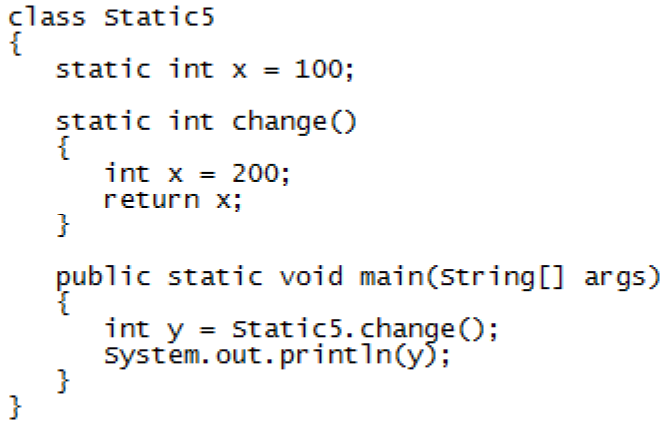
Static Method (Cont…)

- Output



```
100
200
This is an instance method
100
This is a static method
100
```

Static Method (Cont…)



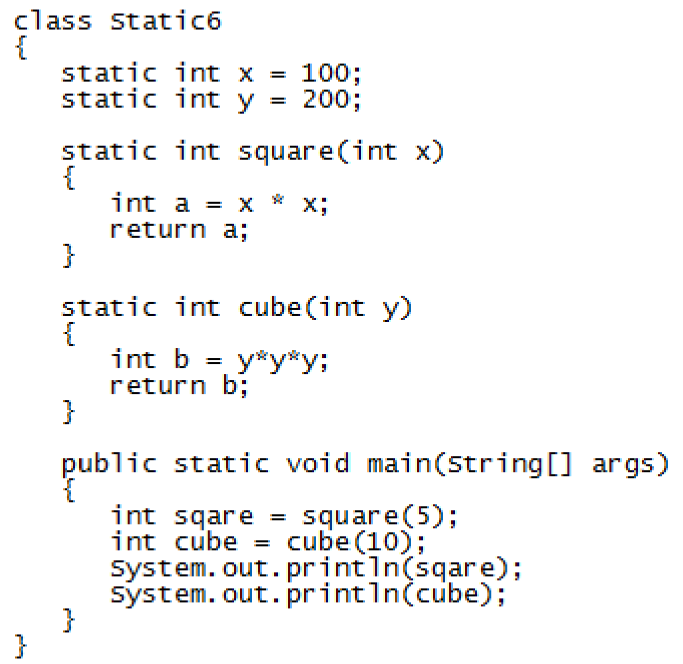
```
class Static5
{
    static int x = 100;

    static int change()
    {
        int x = 200;
        return x;
    }

    public static void main(String[] args)
    {
        int y = Static5.change();
        System.out.println(y);
    }
}
```

Static Method (Cont…)

- Output

```
200
```

Static Method (Cont…)



```
class Static6
{
   static int x = 100;
   static int y = 200;

   static int square(int x)
   {
      int a = x * x;
      return a;
   }

   static int cube(int y)
   {
      int b = y*y*y;
      return b;
   }

   public static void main(String[] args)
   {
      int sqare = square(5);
      int cube = cube(10);
      System.out.println(sqare);
      System.out.println(cube);
   }
}
```

Static Method (Cont…)

- Output

```
25
1000
```

Static Method (Cont…)

```
public class Static7
{
    public static void main(String[] args)
    {
        String nameStudent = Test.name("Vipin");
        int rollStudent = Test.rollNo(5);
        System.out.println("Name of Student: " + nameStudent);
        System.out.println("Roll no. of Student: " + rollStudent);
    }
}
```

```
public class Test
{
    static String name(String n)
    {
        return n;
    }

    static int rollNo(int r)
    {
        return r;
    }
}
```

Static Method (Cont…)

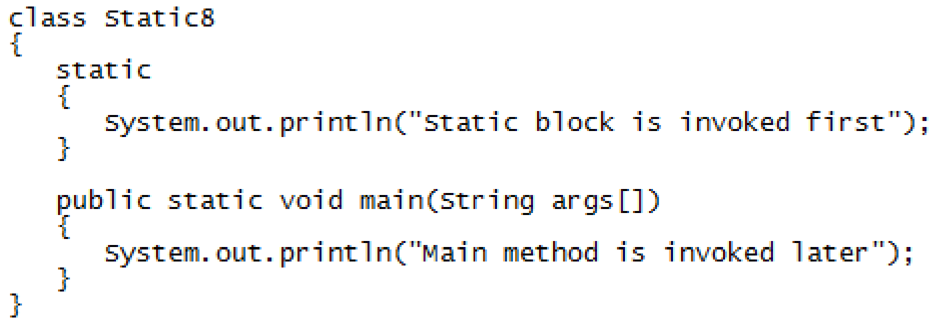
- Output

```
Name of Student: Vipin
Roll no. of Student: 5
```

Static Block

- The static block is a block of statement inside a Java class that is executed when a class is first loaded into the JVM.
- It is executed before the main method at the time of class loading.
- Static block is used for initializing the static variables.
- A class can have multiple Static blocks, which will execute in the same sequence in which they have been written.

Static Block (Cont…)



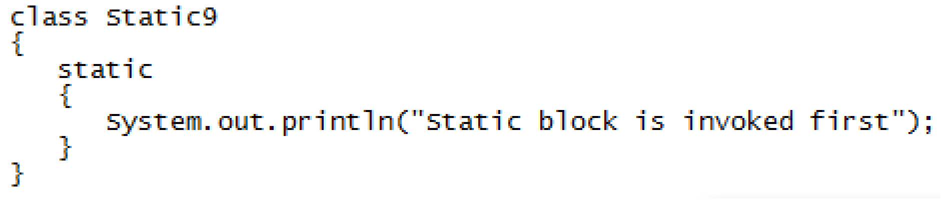
```
class Static8
{
    static
    {
        System.out.println("Static block is invoked first");
    }

    public static void main(String args[])
    {
        System.out.println("Main method is invoked later");
    }
}
```

Static Block (Cont…)

- Output

```
Static block is invoked first
Main method is invoked later
```

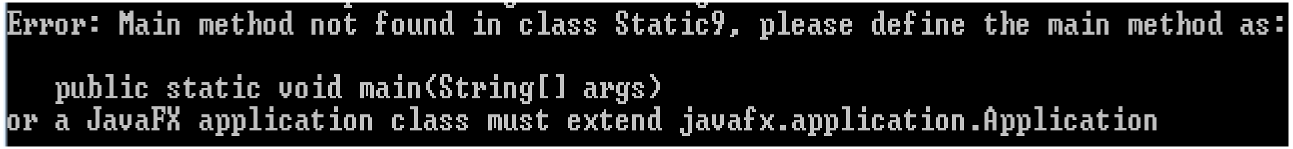
Static Block (Cont…)



```
class Static9
{
   static
   {
      System.out.println("Static block is invoked first");
   }
}
```

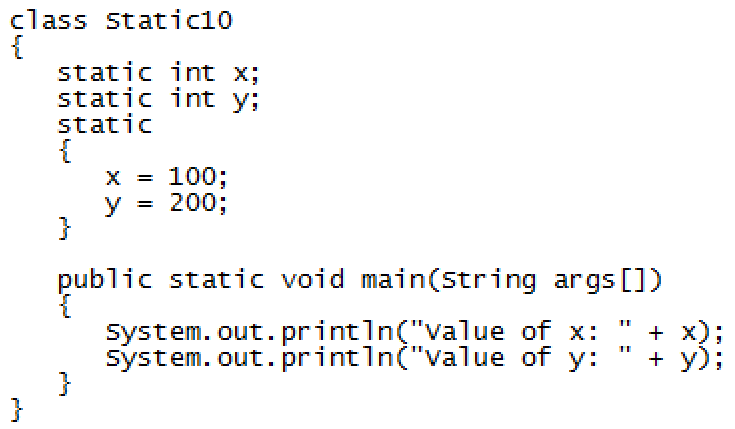
Static Block (Cont…)

- Output



```
Error: Main method not found in class Static9, please define the main method as:

   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
```

Static Block (Cont…)



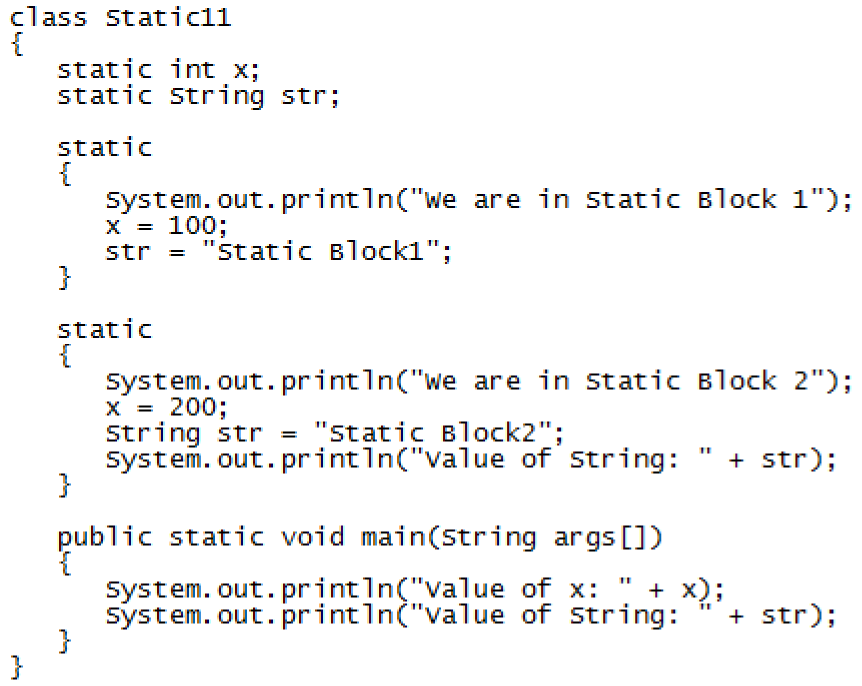
```
class Static10
{
    static int x;
    static int y;
    static
    {
        x = 100;
        y = 200;
    }

    public static void main(String args[])
    {
        System.out.println("Value of x: " + x);
        System.out.println("Value of y: " + y);
    }
}
```

Static Block (Cont…)

- Output

```
Value of x: 100
Value of y: 200
```

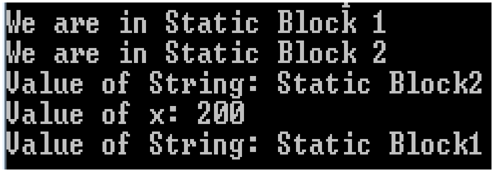
Static Block (Cont…)



```
class Static11
{
    static int x;
    static String str;

    static
    {
        System.out.println("We are in Static Block 1");
        x = 100;
        str = "Static Block1";
    }

    static
    {
        System.out.println("We are in Static Block 2");
        x = 200;
        String str = "Static Block2";
        System.out.println("Value of String: " + str);
    }

    public static void main(String args[])
    {
        System.out.println("Value of x: " + x);
        System.out.println("Value of String: " + str);
    }
}
```

Static Block (Cont…)

- Output



```
We are in Static Block 1
We are in Static Block 2
Value of String: Static Block2
Value of x: 200
Value of String: Static Block1
```

Static Class

- A class can be made static only if it is a nested class.
- Nested static class doesn’t need reference of outer/main class.
- A static class cannot access non-static members of the outer class.
- To create an instance of nested static class, we don't need the outer class instance. However, for a regular nested class, we need to create an instance of outer class first.
- **Nested classes** represent a special type of relationship that is it can access all the members of outer class including private.
- Nested classes are used to develop more readable and maintainable code because it logically group classes and interfaces in one place only.

Static Class (Cont…)

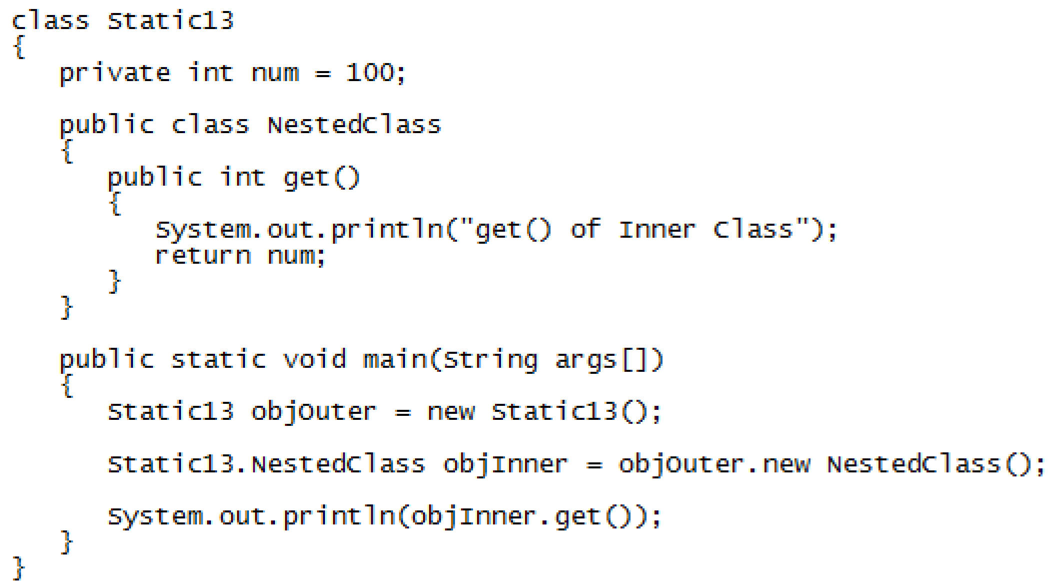
```
class Static12
{
   private static String str = "Print Nested Static Class";

   static class NestedStatic
   {
      public void disp()
      {
         System.out.println(str);
      }
   }

   public static void main(String args[])
   {
      Static12.NestedStatic obj = new Static12.NestedStatic();
      obj.disp();
   }
}
```

Static Class (Cont…)

- Output

```
Print Nested Static Class
```

Static Class (Cont…)



```
class Static13
{
   private int num = 100;

   public class NestedClass
   {
      public int get()
      {
         System.out.println("get() of Inner Class");
         return num;
      }
   }

   public static void main(String args[])
   {
      Static13 objOuter = new Static13();

      Static13.NestedClass objInner = objOuter.new NestedClass();

      System.out.println(objInner.get());
   }
}
```

Static Class (Cont…)

- Output

```
get() of Inner Class
100
```

thank
you . . .

Slides are prepared from various sources, such as Book, Internet Links and many more.