engineering.vips.edu

VIPS
Technical Campus
योग: कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

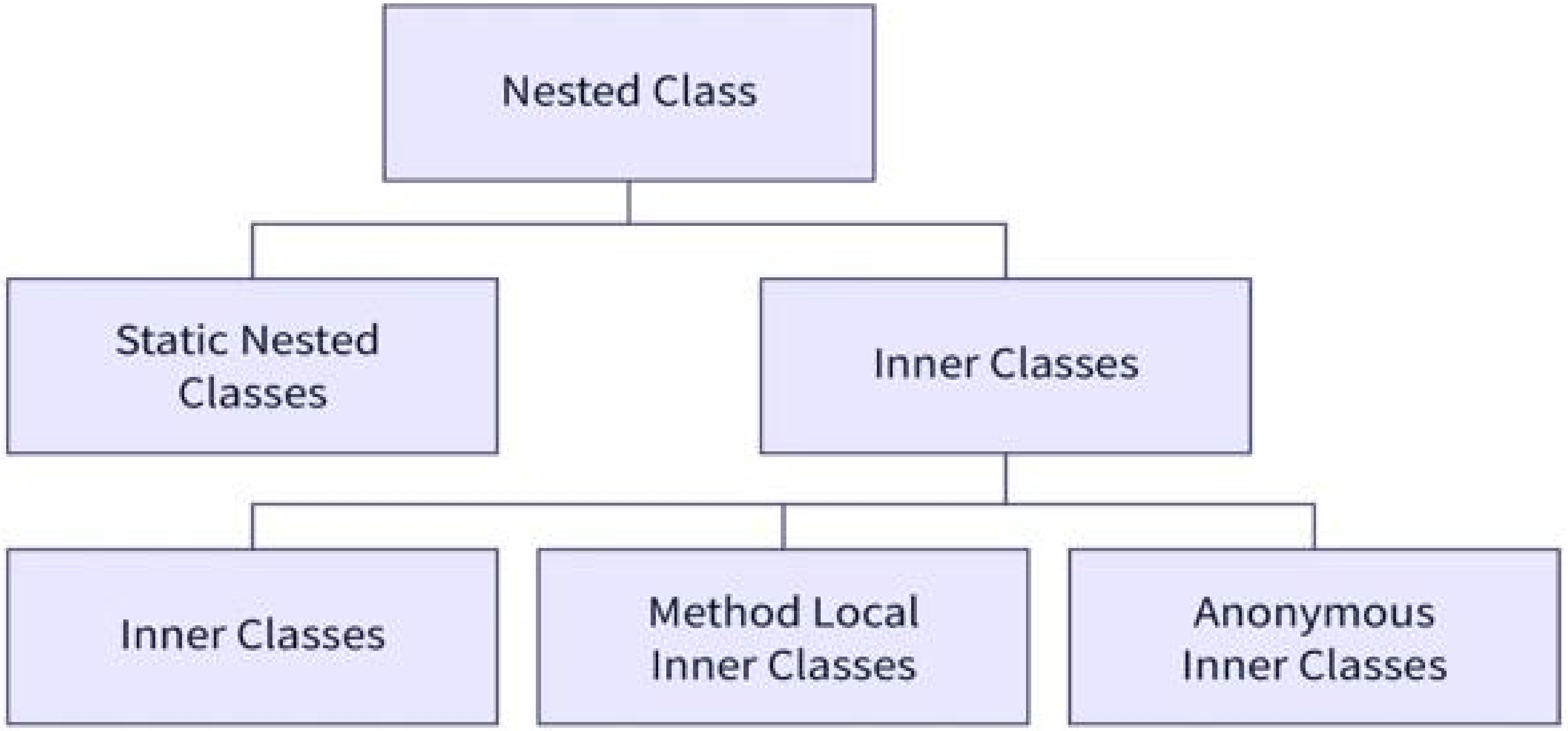
SCHOOL OF ENGINEERING AND TECHNOLOGY

Course : OBJECT ORIENTED PROGRAMMING

Paper Code: AIDS-202

Faculty : Dr. Shivanka
Assistant Professor
VIPS

Anonymous Inner Class



Anonymous Inner Class

- An **anonymous inner class** is one that has **no name** and produces only **one object.**

- **Anonymous classes** can be useful for providing quick implementations of interfaces or abstract classes in situations where creating a named class would be overly verbose or cumbersome.
- In simple words, a class that has no name is known as an **anonymous inner class in Java**. It should be used if you have to **override a method** of class or interface. Java Anonymous inner class can be created in two ways:
- 1.Class (may be abstract or concrete).
- 2.Interface

Note:- An anonymous class must be defined inside another class. Hence, it is also known as an anonymous inner class. Its syntax is:

```
class outerClass {


   // defining anonymous class
   object1 = new Type(parameterList) {
       // body of the anonymous class
   };
}
```

Anonymous classes usually extend subclasses or implement interfaces.

Here, Type can be a superclass that an anonymous class extends an interface that an anonymous class implements the above code creates an object, **object1**, of an anonymous class at runtime.

Types of Anonymous Class in Java

Anonymous classes are typically used to extend subclasses or implement interfaces. So, there can be 2 types of anonymous classes in Java:

- ❖ An anonymous class that extends a superclass.
- ❖ An anonymous class that implements an interface.

Example: Anonymous Class Extending a Class

Let's see an example of an anonymous class that extends a class.

In the example below, we have created a class Demo. It has a single method view(). We then created an anonymous class that extends the class Demo and overrides the view() method.

When we run the program, an object d1 of the anonymous class is created. The object then calls the view() method of the anonymous class.

Example 1: Anonymous Class Extending a Class

```
class Polygon {
   public void display() {
      System.out.println("Inside the Polygon class");  }}
class AnonymousDemo {
   public void createClass() {
      // creation of anonymous class extending class
Polygon
      Polygon p1 = new Polygon() {
         public void display() {
      System.out.println("Inside an anonymous
class.");
         }
};
  p1.display();  }}
```

```
class Main {
   public static void main(String[] args) {
      AnonymousDemo an = new
AnonymousDemo();
      an.createClass();
   }
}
```

Output

```
Inside an anonymous class.
```

Example 2: Anonymous Class Implementing an Interface

```
interface Polygon {
  public void display();}
class AnonymousDemo {
  public void createClass() {
    // anonymous class implementing interface
    Polygon p1 = new Polygon() {
      public void display() {
        System.out.println("Inside an anonymous
class.");
      }};//anonymous inner claass ends
    p1.display();
  }
}
```

```
class Main {
  public static void main(String[] args) {
    AnonymousDemo an = new AnonymousDemo();
    an.createClass();
  }
}
```

Output

```
Inside an anonymous class.
```

In the above example, we have created an anonymous class that implements the Polygon interface.

Example: Anonymous Class Extending a Class

```
class Demo {
public void view() {
    System.out.println("Inside the Demo class");
  }}
class MyClass {
public void myMethod() {// creation of anonymous
class extending class the class Demo
    Demo d1 = new Demo() {//Anonymous Class
      public void view() {
        System.out.println("Inside the Anonymous
Class");
      }
    };
  d1.view();
  }
}
class Main {
  public static void main(String[] args) {
    MyClass cls = new MyClass();
    cls.myMethod();
  }}
```

Output

Inside an anonymous class.

Advantages of Anonymous Classes

- In anonymous classes, objects are created whenever they are required. That is, objects are created to perform some specific tasks. For example,

```
Object = new Example() {
  public void display() {
    System.out.println("Anonymous class overrides the method display().");
  }
};
```

Here, an object of the anonymous class is created dynamically when we need to override the display() method.

- Anonymous classes also help us to make our code concise.

Thank You