Java Technologies
Inner Class

Prepared for: DDU CE Semester 4

Prepared by: Prof. Niyati J. Buch

Inner class in Java

- Inner class means one class which is a member of another class.
- There are basically four types of inner classes in Java.
  1. Nested Inner class
  2. Method Local inner classes
  3. Anonymous inner classes
  4. Static nested classes

https://www.geeksforgeeks.org/inner-class-java/

Nested Inner Class

- **Nested Inner class** can access any private instance variable of outer class. Like any other instance variable, we can have access modifier private, protected, public and default modifier.
- Like class, interface can also be nested and can have access specifiers.
- There cannot be a static method in a nested inner class because an inner class is implicitly associated with an object of its outer class so it cannot define any static method for itself.

NestedInnerClass.java

```
package myclasses;

class OuterClass {

   class InnerClass {

      void method() {
         System.out.println("method in inner class");
      }

   }
}

public class NestedInnerClass {
   public static void main(String args[]) {
      OuterClass.InnerClass in = new OuterClass().new InnerClass();
      in.method();
   }
}
```

OUTPUT

```
method in inner class
```

NestedInnerClass.java

```
package myclasses;
class OuterClass {
   private class InnerClass {
      void method() {
         System.out.println("method in inner class");
      }
   }
   class Test {
      void methodTest() {
         InnerClass iobj = new InnerClass();
         iobj.method();
      }
   }
}
public class NestedInnerClass {
   public static void main(String args[]) {
      OuterClass.Test in = new OuterClass().new Test();
      in.methodTest();
   }
}
```

OUTPUT

```
method in inner class
```

Method Local inner classes

- Inner class can be declared within a method of an outer class.
- Method local inner class can’t be marked as private, protected, static and transient but can be marked as abstract and final, but not both at the same time.

MethodLocalInnerClass.java

```
package myclasses;
class OuterClass {
   void methodOuter() {
      int x =10;
      System.out.println("in methodOuter");
      class InnerClass {
         void methodInner() {
            System.out.println("in methodInner");
            System.out.println(x);
         }
      }
      InnerClass iobj = new InnerClass();
      iobj.methodInner();
   }
}
public class MethodLocalInnerClass {
   public static void main(String args[]) {
      OuterClass obj = new OuterClass();
      obj.methodOuter();
   }
}
```

OUTPUT

```
in methodOuter
in methodInner
10
```

Static nested classes

- Static nested classes are not technically an inner class.
- They are like a static member of outer class.

StaticInnerClass.java

```
package myclasses;
class OuterClass {

   static void method1() {
      System.out.println("in method of outer class");
   }

    static class InnerClass {
      public static void main(String args[]) {
         method();
         method1();
      }
      static void method() {
         System.out.println("in method of static inner class");
      }
   }
}
```

```
>javac -d . StaticInnerClass.java
>java myclasses.OuterClass$InnerClass
in method of static inner class
in method of outer class
```

Anonymous inner classes

- Anonymous inner classes are declared without any name at all.
- They are created in two way:
  - As subclass of specified type
  - As implementor of specified interface

AnonymousInnerClass1.java

```
package myclasses;

class Demo {
  void show() {
    System.out.println("show method of super class");
  }
}
public class AnonymousInnerClass1 {

  //  An anonymous class with Demo as base class
  static Demo d = new Demo() {
     @Override
     void show() {
        super.show();
        System.out.println("overridden show method");
     }
  };
  public static void main(String[] args){
     d.show();
  }
}
```

OUTPUT

```
show method of super class
overridden show method
```

AnonymousInnerClass2.java

```
package myclasses;

interface Demo {
   void show();
}

public class AnonymousInnerClass2 {

   //  An anonymous class that implements Demo interface
   static Demo d = new Demo() {
      @Override
      public void show() {
         System.out.println("overridden show method");
      }
   };

   public static void main(String[] args) {
      d.show();
   }
}
```

OUTPUT
overridden show method

Anonymous Object

- Anonymous simply means nameless.
- An object which has no reference is known as an anonymous object.
- It can be used at the time of object creation only.
- If you have to use an object only once, an anonymous object is a good approach

AnonymousObject.java

```
package myclasses;
class A {

   int i = 10;

   void method() {
      System.out.println("in method of A class");
   }
}

public class AnonymousObject {

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

OUTPUT

```
10
in method of A class
```

Topics covered so far

- Inner Class
- Anonymous object