មជ្ឈមណ្ឌលកូរ៉ែ សហ្វវែរ អេច អ ឌី

Korea Software HRD Center

ការស្វែងយល់ពី Nested Class ក្នុង Java

Java Nested Class

ណែនាំដោយ : Dr. Kim Tae Kyung

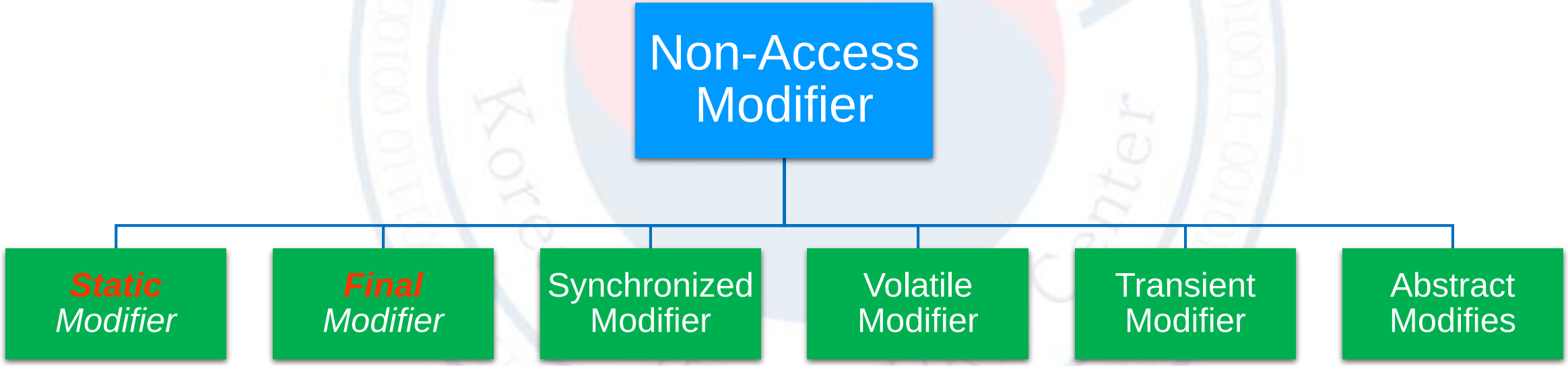
Webcash

KOICA
Korea International Cooperation Agency

មាតិកា

១. ការស្វែងយល់អំពី Static និង Final Keyword

- ក្នុងភាសា Java គេចែក Non access modifier ជា ៦ ប្រភេទ៖



១.១. ការស្វែងយល់អំពី Static Keyword

- អ្វីជា Static Keyword?
  - Static Keyword គឺជា Keyword ប្រើក្នុងការគ្រប់គ្រង memory។ ហើយ static member មានទំនាក់ទំនងផ្ទាល់ជាមួយ class តែមួយ ជាជាង instance ជាក់លាក់ណាមួយ។
  - វាធ្វើអោយកម្មវិធីដំណើរការល្អព្រោះវាប្រើប្រាស់ memory តិច។

Static Keyword អាចប្រើជាមួយនឹង៖

- Variable (class variable)
- Method (class method)
- Blocks
- Nested class

១.១. ការស្វែងយល់អំពី Static Keyword (ឥ)

- Syntax នៃការប្រកាស static variable:

```
access-modifier static dataType variableName;
```

- ការប្រើប្រាស់ Static Keyword ជាមួយ variable
  - យើងប្រើ Static Keyword ជាមួយ variable នៅពេលដែលគ្រប់ object ទាំងអស់ត្រូវប្រើ field មួយរួមគ្នា។

១.១. ការស្វែងយល់អំពី Static Keyword (ត)

- ការប្រើប្រាស់ Static Keyword ជាមួយ method
  - យើងប្រើ Static Keyword ជាមួយ method ចំពោះ method ណាដែលមិនមែនជារបស់ object ណាទេ។

- កំណត់សំគាល់ចំពោះ static method
  - មិនអាចប្រើ data member ដែលមិនមែនជា static បានទេ
  - មិនអាចប្រើ keyword this និង super
  - មិនអាចប្រើ static keyword ជាមួយ constructor

១.១. ការស្វែងយល់អំពី Static Keyword (ត)

- Syntax នៃការប្រកាស static method:

```
public static void methodName(){
    //statement
}
```

១.១. ការស្វែងយល់អំពី Static Keyword (ត)

- ហេតុអ្វីបានជា main method របស់ java ប្រើប្រាស់ keyword static
  - ដោយសារតែ non-static method ត្រូវការ object នៅពេលដំណើរការ ។ ប្រសិនបើ main method ជាប្រភេទ non-static នោះ jvm ត្រូវការបង្កើត object មុនពេលហៅ main method ដែលជាហេតុធ្វើអោយប្រើប្រាស់ memory ច្រើន។

- ការប្រើប្រាស់ static block ដើម្បី៖
  - ផ្តល់តំលៃដំបូងទៅអោយ data member ដែលជាប្រភេទ static
  - វាដំណើរការមុន main method គឺនៅពេលដែល class ត្រូវបាន load

១.១. ការស្វែងយល់អំពី Static Keyword (ត)

- **ឧទាហរណ៍:** ការប្រើប្រាស់ static block

```
class Student {

    static {
        String school = "KSHRD Center";
    }
}
```

១.១. ការស្វែងយល់អំពី Static Keyword (ឥ)

- Syntax នៃការប្រើប្រាស់ static methods and static variable

```
ClassName.variableName="Value";
ClassName.methodName();
```

១.១. ការស្វែងយល់អំពី Static Keyword (ត)

- **ឧទាហរណ៍:** ការបង្ហាញពីសិស្សនៅក្នុងសាលាមួយប្រើ ឈ្មោះសាលាដូចគ្នា

```
public class Student {
        String name;
        static String collegeName = "KSHRD Center";

        public static void main(String[] args){
                Student stu1 = new Student();
                Student stu2 = new Student();
                stu1.name= "Kokpheng";
                stu2.name= "Pirang";

                System.out.println(stu1.name);
                System.out.println(stu1.collegeName);
                System.out.println(stu2.name);
                System.out.println(stu2.collegeName);
        }
}
```

Output:

```
Kokpheng
KSHRD Center
Pirang
KSHRD Center
```

១.១. ការសិក្សាយល់អំពី Static Keyword (ត)

- **ឧទាហរណ៍:** Static method និង Static Block

```
public class TestStatic {
    // Static Block is executed before main method
    // at the time of classloading
     static {
        System.out.println("Static Block 1 is involked"); }

     static void staticMethod() {
        System.out.println("Static Method is involked"); }

     static {
       System.out.println("Static Block 2 is involked"); }

     public static void main(String[] args) {
        staticMethod(); // Call Static Method
     }
}
```

Output:

```
Static Block 1 is invoked
Static Block 2 is invoked
Static Method is invoked
```

១.២. ការស្វែងយល់អំពី Final Keyword

- អ្វីជា Final Modifier?
  - Final Modifier គឺជាប្រភេទ Keyword ដែលប្រើសំរាប់ restrict user និងបញ្ជាប់ការ Implement ទៅលើ *class variable* ឬ *method* ។

Final Keyword

* **Restrict changing value of variable**
* **Restrict method overriding**
* **Restrict inheritance**

- ✓ Final at variable level
- ✓ Final at method level
- ✓ Final at class level

១.២. ការស្វែងយល់អំពី Final Keyword (ត)

- **Final at variable level**
  - Final Keyword ប្រើសំរាប់បង្កើតនូវអថេរ(Variable) ជាលក្ខណៈ Constant
  - អថេរ (variable) ដែលបានប្រកាសជាមួយនឹង Keyword Final គឺមិនអាចធ្វើការកែប្រែតម្លៃបានឡើយ នៅពេល runtime បានទេ ។
  - ការដាក់ឈ្មោះឲ្យ Final Variable
    - ✓ ត្រូវសរសេរជាអក្សរធំទាំងអស់
    - ✓ ប្រើសញ្ញា (_) បើមានចាប់ពី ២ ពាក្យឡើង។

```
public class Circle {

        public static final double PI = 3.14159;

        public static void main(String[] args) {
                System.out.println(PI);
        }
}
```

១.២. ការស្វែងយល់អំពី Final Keyword (ត)

- Final at variable level
  - Final variable ដែលមិនមានតំលៃ ត្រូវបានគេហៅថា blank final variable រឺ uninitialized variable។ យើងអាច initialize តំលៃអោយវាបានតែនៅក្នុង constructor តែប៉ុណ្ណោះ។
  - ឧទាហរណ៍: Blank final variable

```
public class Student {
	final String SCHOOL_NAME;

	public Student() {
		SCHOOL_NAME = "KSHRD Center";
	}
}
```

១.២. ការស្វែងយល់អំពី Final Keyword (ត)

- Final at variable level
  - ចំពោះ variable ដែលបានប្រកាសជា static final ហើយមិនមានតំលៃ យើងអាច initialize តំលៃអោយវាបានតែនៅក្នុង static block តែប៉ុណ្ណោះ។
  - ឧទាហរណ៍: Static final variable

```
public class School {

	static final String SCHOOL_NAME;
	static {
		SCHOOL_NAME = "KSHRD Center";
	}
}
```

១.២. ការស្វែងយល់អំពី Final Keyword (ត)

- **Final at method level**
  - Final Keyword ដែលប្រើជាមួយនឹង Method មានន័យថា subclass មិនអាចធ្វើការ Override ទៅលើ method នោះឡើយ។
  - ប៉ុន្តែយើងអាចបង្កើត overloading method បាន

១.២. ការស្វែងយល់អំពី Final Keyword (ត)

- រូបភាព៖ បង្ហាញអំពីការប្រើប្រាស់ Final Keyword ទៅលើ method.

```
public class Employee{

        final void display() {
                System.out.println("Good Morning");
        }
}

class Developer extends Employee{
        void display() { // This line give an error because
                         // we can not override final method
                System.out.println("How are you?");
        }

        void display(String para){ // Overloading method is ok
        }
}
```

Multiple markers at this line
- Cannot override the final method from Employee
- overrides Employee.display

១.២. ការសិក្សាយល់អំពី Final Keyword (ត)

- **Final at class level**
  - Final Keyword ប្រើជាមួយនឹង Class មានន័យថា មិនអាចអោយ class ផ្សេងទៀតធ្វើការ extends/inherit បាន។

ឧទាហរណ៍:

```
public final class Employee {
        // Statements
}

public class Developer extends Employee {
        // it gives an error, because we
        // can not inherit final class

        // Statements
}
```

The type Developer cannot subclass the final class Employee

២. ការស្វែងយល់យល់អំពី Nested Class

អ្វីទៅជា Inner class / Nested Class ?

- Nested class ជា class ដែលបង្កើត នៅក្នុង Class ឬ Interface មួយទៀត។
- Nested class អាច access រាល់ member របស់ class ខាងក្រៅ ដោយរួមបញ្ចូលនូវ private data members ហើយនិង methods។ ប៉ុន្តែ class ខាងក្រៅ access វិញ គឺត្រូវឆ្លងការ instance នៃ inner class។

Syntax:

```
class OuterClass {
        ...
      class NestedClass {
              ...
      }
}
```

២. ការស្វែងយល់យល់អំពី Nested Class (ត)

ហេតុអ្វីប្រើ Nested Class ? មានប្រយោជន៍ 3:

1. ប្រើបណ្តុំ Class នៅកន្លែងតែមួយ:
   - បើ class A មានប្រយោជន៍តែសំរាប់ class B គឺត្រូវតែបញ្ចូល class A ទៅក្នុង class B នោះ ឲ្យជាបណ្តុំជាមួយគ្នា។

2. បង្កើន encapsulation:
   - Class Top-Level ពីរ គឺ class A និង class B។ ដោយបញ្ចូល class B ទៅក្នុង class A, class B អាច access members របស់ class A បើទោះជា members នោះ ប្រកាសជា private ក៏ដោយ។ ហើយយើងក៏អាចលាក់ Class B បាន ។

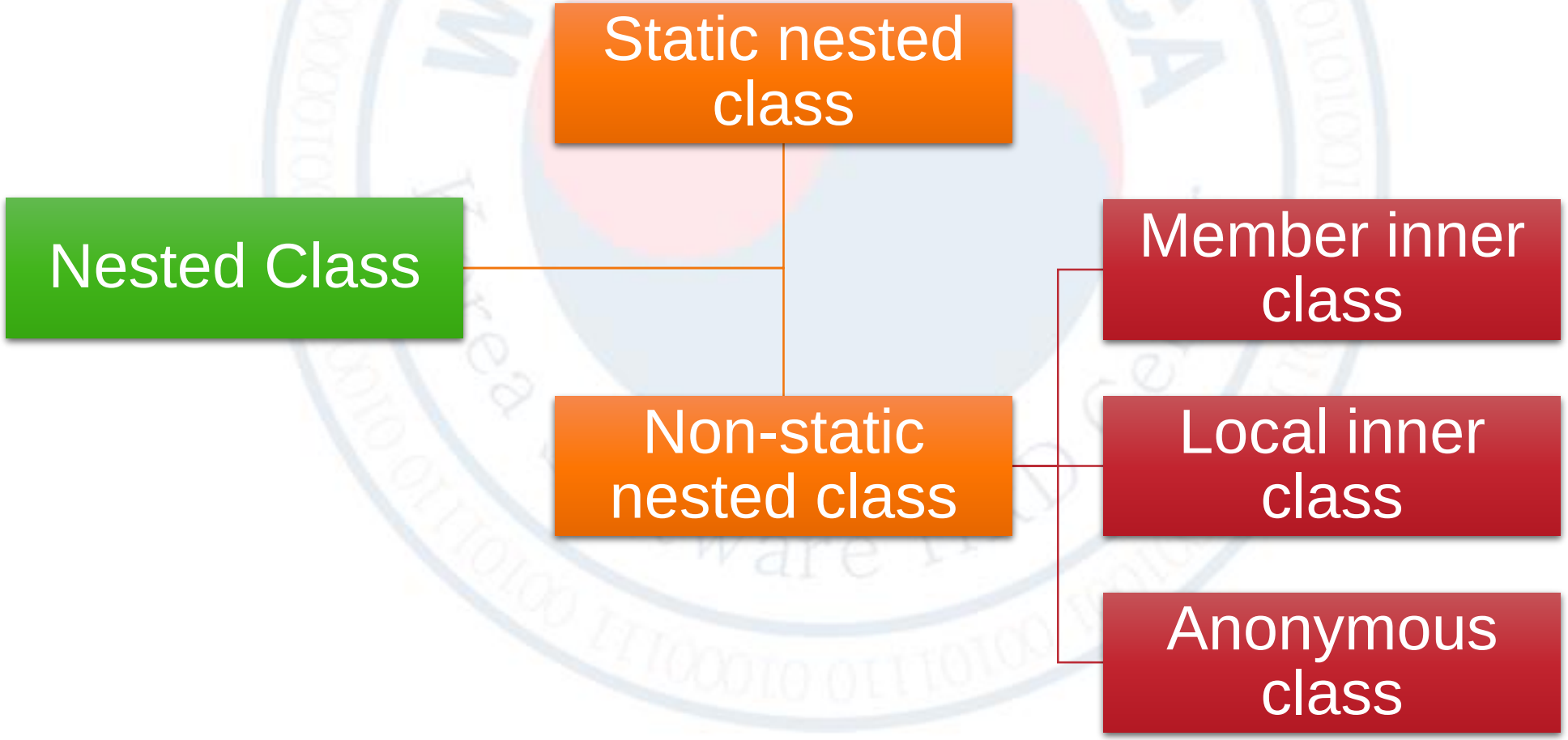
២. ការស្វែងយល់យល់អំពី Nested Class (ត)

ហេតុអ្វីប្រើ Nested Class ? មានប្រយោជន៍ 3:

3. ធ្វើឲ្យកូដស្រួលអាន និង ស្រួលការពារ កែប្រែកូដ more readable and maintainable:
   - ដាក់ class តូចៗ ក្នុង top-level class ជាការធ្វើកូដស្ថិតនៅជិតកន្លែងប្រើប្រាស់។

២. ការស្វែងយល់យល់អំពី Nested Class (ត)

Java មាន Nested Class ២ ប្រភេទគឺ:



២.១. សិក្ខាអំពី Static Inner Class

- Static Nested Class គឺជា class នៅជាប់នឹង outer class ដូចជា class methods និង Variables ដែរ។
- Static Nested Class ស្រដៀងនឹង Static Class Method ដែរ មិនអាចហៅ instance variables ឬ methods ដែលបង្កើតក្នុង enclosing class បានទេ។ វាអាចធ្វើ បានតាមរយៈ Object reference ។
- វាត្រូវបានបង្កើតនៅក្នុង Class ហើយវាមាន Keyword static នៅពីមុខវា។ Static nested class គឺមិនអាច access non-static data member និង method របស់ outer class បានទេ។ តែវាអាច access private static member បាន។

២.១. សិក្សាអំពី Static Inner Class (ត)

- Static nested class អាចទំនាក់ទំនងជាមួយ instance members របស់ outer class និង top-level class ដទៃដែរ ។ ហេតុនេះ static nested class ក៏ជា top-level class ដែលត្រូវគេដាក់ក្នុងមួយទៀត សំរាប់ភាពងាយស្រួលនៃ packaging។

- យើង access static nested class ដោយប្រើ enclosing class name:

```
OuterClass.StaticNestedClass
```

២.១. សិក្ខាអំពី Static Inner Class (ត)

- របៀបប្រកាស Static nested Class

```
public class Outer {
        public static class StaticInner {
        }
}
```

- Syntax: ដើម្បីបង្កើត object របស់ static nested class:

```
Outer.StaticInner innerObj = new Outer.StaticInner();
```

២.១. សិក្ខារំពី Static Inner Class (ត)

- ឧទាហរណ៍ of Static inner class:

```
public class OuterClass {
        // Outer class member
        static int age = 20;
        private static String name = "Bobo";

        // Static inner class
        static class InnerClass {
                void show() {
                        System.out.println("My name is " + name + " and I'm " + age + " years old.");
                }
        }

        // Outer class main method
        public static void main(String[] args) {
                OuterClass.InnerClass inner = new OuterClass.InnerClass();
                inner.show();
        }
}
```

២.២. សិក្ខាអំពី Non-Static Inner Class

- Member inner class គឺជា non-static class ដែលប្រកាសនៅក្នុង class ប៉ុន្តែក្រៅ method។

  - Syntax:

```
public class OuterClass {

    // member inner class
    class InnerClass {
    // Statements
    }
}
```

២.២.១. Member inner class

- ការបង្កើត object របស់ member inner class

```
public class StaticKeyword {

    // outer class main method
    public static void main(String[] args) {
        // outer class object
        Outer oc = new Outer();

        // inner class object
        Outer.Inner in = oc.new Inner();

        //Outer.Inner in = new Outer().new Inner();

        // invoke inner class method
        in.print();
    }
}
```

```
class Outer {

    // member inner class
    class Inner {
        void print() {
            System.out.println("member");
        }
    }
}
```

២.២.២. Local inner class

- អ្វីទៅជា Local Inner Class ?
  - Local Inner Class ជា class មួយដែលត្រូវបានបង្កើតនៅក្នុង method។ បើសិនជាយើងចង់ហៅ method នៃ local inner class យើងត្រូវតែធ្វើការ instantiate object នៃ local inner class នោះនៅក្នុង method។

```
// Outer Class Method
void outerClassMethod() {
    // Local inner Class create inside method
     class LocalClass {
          void LocalMethod() {
             // Statements
           }
      }
     // To use LocalMethod, we must instantiate Local Class inside the Outer Class Method
     LocalClass local = new LocalClass();
     local.LocalMethod();
 }
```

២.២.២. Local inner class (៥)

- របៀបប្រើប្រាស់ Local Inner Class:
  - Local variable មិនអាចប្រកាសជា private, public ឬ protected ទេ
  - មិនអាចត្រូវបានហៅពី method ខាងក្រៅទេ
  - មិនអាចធ្វើការ access non-final local variable នៅក្នុង Method បានទេរហូតដល់ JDK 1.7។ តែ JDK 1.8 អាច access បាន
  - មិនអនុញ្ញាតឲ្យមាន member ជា static ទេ ដូចជា Interface

២.២.២. Local inner class (៥)

- **<u>ឧទាហរណ៍:</u>**

```
class Test_Local_Inner {
      private String room = "Phnom Penh"; // instance variable
      public void getName() {
             final String group = "3"; // Local variable must be final
                                       // for JDK version below 1.7
             class LocalInner {
                    void msg() {
                          System.out.println("Room : " + room);
                          System.out.print("Group : " + group); }
             }
             LocalInner li = new LocalInner();
             li.msg();
          }
       public static void main(String[] args) {
             Test_Local_Inner name = new Test_Local_Inner();
             name.getName();
       }
}
```

<u>Output:</u>

Room : Phnom Penh
Group : 3

២.២.៣. Anonymous inner class

- អ្វីទៅជា **Anonymous inner class** ?
  - Inner Class មួយដែល **គ្មានឈ្មោះ** ត្រូវបានចាត់ទុកថាជា anonymous inner class នៅក្នុង java។ វាត្រូវបានបង្កើតនៅពេលដែល instance ចាប់ផ្តើម **បង្កើត** និងនៅក្នុង class ដទៃ។
  - ជាធម្មតា statements របស់វាត្រូវបានប្រកាសនៅក្នុង method ឬ code block ដែលស្ថិតនៅក្នុង curly braces ហើយបញ្ចប់ដោយសញ្ញា semicolon (;)
  - Anonymous មិនអាចជា Static ឡើយ។

២.២.៣. Anonymous inner class (៥)

- Anonymous inner class អាចប្រើបានតែមួយដងទេ នៅទីតាំងដែលយើងសរសេរកូដ មានន័យថាបើសិនយើងចង់បង្កើតតែ sub-classed object នៃ class មួយនោះ យើងមិនចាំបាច់ផ្តល់ ឈ្មោះ class ទេ ហើយយើងអាចប្រើ anonymous inner class ក្នុងករណីនោះបាន។ គេប្រើវាដើម្បី override method របស់ Class ឬ Interface។
- Anonymous inner class មិនអាចមានការប្រកាស constructors ពីព្រោះវា មិនមានឈ្មោះ ដែលសម្រាប់ផ្តល់ឲ្យ constructor
- Anonymous inner class អាចបង្កើតបាន ២ របៀប ដែលវាខុសពី class ធម្មតា៖
  - Class (អាចជា abstract ឬ concrete)
  - Interface

២.២.៣. Anonymous inner class (ត)

នេះគឺជាឧទាហរណ៍ដែលបង្ហាញពីការបង្កើត anonymous inner class។ reference variable dog នៃ Dog មិនមែនសំដៅទៅលើ instance នៃ Dog ទេ ប៉ុន្តែវាសំដៅលើ instance នៃ anonymous inner subclass មួយរបស់ Dog។

```
class Dog {
    public void someDog() {
      System.out.println("Classic Dog");
   }
}
```

Output:

Anonymous Dog

```
public class AnonymousClass {
    public static void main(String[] args) {
        Dog dog = new Dog() {
            public void someDog() {
                System.out.println("Anonymous Dog");
            }
    }; // anonymous class body closes here
// dog contains an object of anonymous subclass of Dog.
        dog.someDog();
    }
}
```

២.២.៣. Anonymous inner class (ត)

ឧទាហរណ៍: anonymous inner class ជាមួយ interface

```
/* AnonymousInterface */
interface Manageable {
        public void manage();
}
```

```
public class AnonymousClass {
  public static void main(String[] args) {
      Manageable m = new Manageable() {
          public void manage() {
              System.out.println("It is manageable");
           }
      }; // anonymous interface implementer closes here

 // m contains an object of anonymous interface
 // implementer of Manageable.
      m.manage();
  }
}
```