

Bài 12A. Nested Class

1

1. Khái niệm

- Java cho phép định nghĩa 1 class trong class khác → Gọi là **nested class**
 - Nested class là 1 thành viên của lớp bao nó
 - Các loại từ chỉ định truy cập: public, private, protected, default

- Ví dụ:

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

2

2

2. Tại sao sử dụng nested class?

- Nhóm 1 cách logic các class được sử dụng chỉ ở 1 nơi
 - Nếu 1 class hữu ích chỉ cho 1 class khác → sẽ logic nếu nhúng class đó vào class kia
- Tăng tính đóng gói
 - Giả sử có class A và B
 - B cần truy cập tới các thành phần private của A
 - Đặt B là nested Class trong A
 - B có thể bị ẩn với bên ngoài (private, ...)
- Giúp dễ đọc code và dễ bảo trì

3

3

3. Phân loại

- Nested class chia làm 2 loại: static và non-static
 - Static nested class**: Nếu nested class được khai báo là static
 - Inner class**: ngược lại

- Ví dụ:

```
class OuterClass {  
    ...  
    static class StaticNestedClass {  
        ...  
    }  
    class InnerClass {  
        ...  
    }  
}
```

4

4

3.1. Static nested classes

- Được truy cập từ tên của class bao nó
 - Ví dụ: `OuterClass.StaticNestedClass`
- Để tạo 1 đối tượng của static nested class:
 - `OuterClass.StaticNestedClass nestedObject = new OuterClass.StaticNestedClass();`
- Chỉ được truy cập các thành viên static của class bao nó

```
public class Outer {
    private int id;
    public static class Inner {
        private int localId;
        public Inner() {
            localId=0000;
            id = 0; //Error
        }
    }
}
```

5

5

3.1. Static nested classes (2)

```
public class Outside {
    public static class Skinside {
        public Skinside()
        {
            System.out.println("Demo static");
        }
    }

    public class Inside {
    }

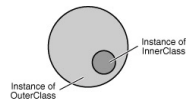
    public static void main(String[] arg) {
        Outside.Skinside example = new Outside.Skinside();
    }
}
```

6

6

3.2. Inner Class

- 1 thể hiện (instance) của inner class chỉ tồn tại được trong 1 thể hiện của outer class
 - Để khởi tạo đối tượng cho inner class, phải khởi tạo đối tượng của outer class trước
 - `OuterClass.InnerClass innerObject = outerObject.new InnerClass();`



7

7

3.2. Inner Class (2)

- Inner class có thể truy cập tới 1 member bất kỳ của outer class
- Inner class không được có thành phần static

```
public class Outer {
    private int id;
    private class Inner {
        private static int defaultId; //Error
        public Inner() {
            id = 00001; //Truy cập được id ngoài
        }
    }
}
```

8

8

```

public class DataStructure {
    private final static int SIZE = 15;
    private int[] arrayOfInts = new int[SIZE];
    public DataStructure() { //fill the array with ascending integer values
        for (int i = 0; i < SIZE; i++) {
            arrayOfInts[i] = i;
        }
    }
    public void printEven() { //In chỉ số lẻ trong mảng
        InnerEvenIterator iterator = this.new InnerEvenIterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.getNext() + " ");
        }
    }
    private class InnerEvenIterator { //inner class implements the Iterator pattern
        //start stepping through the array from the beginning
        private int next = 0;
        public boolean hasNext() {
            return (next <= SIZE - 1); //check if current element is the last in the array
        }
        public int getNext() {
            int retValue = arrayOfInts[next];
            next += 2; //get the next even element
            return retValue;
        }
    }
    public static void main(String s[]) {
        //fill the array with integer values and print out only values of even indices
        DataStructure ds = new DataStructure();
        ds.printEven();
    }
}

```

9

3.2. Inner Class (3)

- Inner Class lại chia làm 2 loại con:
 - local inner class: Khai báo 1 inner class trong 1 method
 - anonymous inner classes: Khai báo 1 inner Class trong 1 method nhưng không đặt tên

10

a. local inner class

```

class Outer {
    int outer_x = 100;

    void test() {
        for(int i=0; i<10; i++) {
            class Inner {
                void display() {
                    System.out.println("display: outer_x = " + outer_x);
                }
            }
            Inner inner = new Inner();
            inner.display();
        }
    }
}

class InnerClassDemo {
    public static void main(String args[]) {
        Outer outer = new Outer();
        outer.test();
    }
}

```

display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100

11

b. anonymous inner classes

```

interface Counter {
    int next();
}

public class Outer {
    private int count = 0;
    Counter getCounter(final String name) {
        return new Counter() {
            {
                System.out.println("Constructor Counter() ");
            }
            public int next() {
                System.out.print(name); // Access local final
                System.out.println(count);
                return count++;
            }
        };
    }

    public static void main(String[] args) {
        Outer out = new Outer();
        Counter c1 = out.getCounter("Local inner ");
        c1.next();
        c1.next();
    }
}

```

Constructor Counter()
Local inner 0
Local inner 1

Anonymous inner class cannot have a named constructor, only an instance initializer

12