Bài 33: Các thành phần static

- ✓ Nested class
- ✓ Static nested class
- ✓ Inner class
- ✓ Local class
- ✓ Anonymous class

Nested class

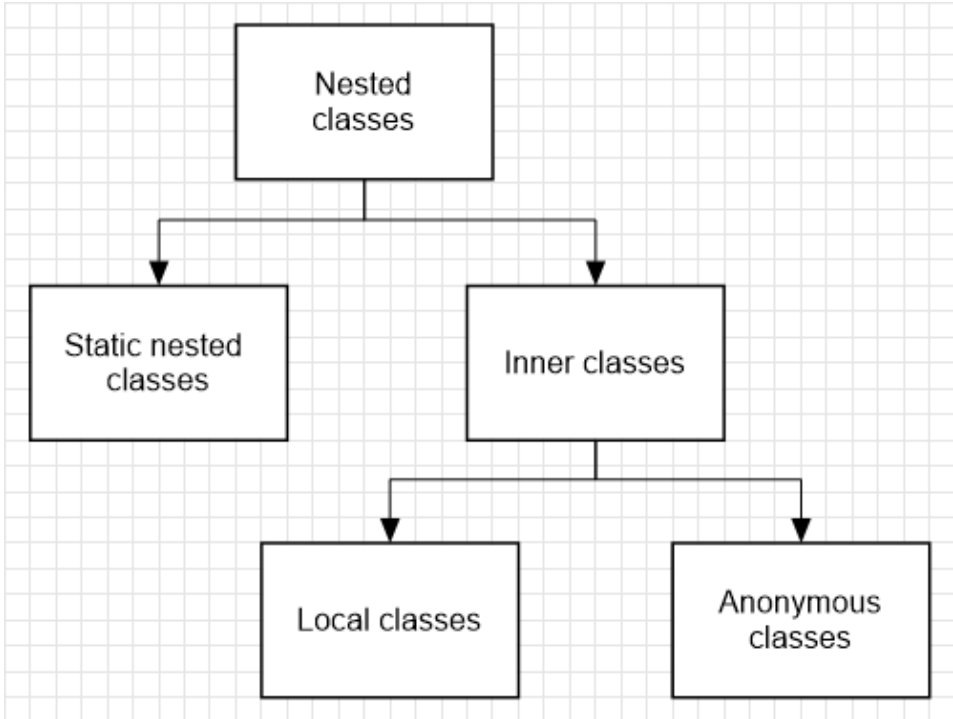
- Lớp được định nghĩa bên trong một lớp khác gọi là nested class
- Ví dụ:

```
public class OuterClass {
    //...
    class NestedClass {
        //...
    }
    //...
}
```

- Nested class lại chia làm hai loại:
  - Static nested class và
  - Inner class

Nested class

➢Phân loại các nested class:



Phân loại nested class

- Static nested class là các lớp nested có chứa keyword static trong khai báo lớp
- Các nested class không chứa keyword static gọi là các inner class hay non-static nested class
- Một nested class là thành phần của lớp chứa nó. Chính là thành phần thứ tư trong phần cú pháp tổng quan định nghĩa lớp
- Các inner class có thể truy cập tất cả các thành phần của lớp bên ngoài chứa nó kể cả thành phần private

Phân loại nested class

- Các lớp static nested thường không truy cập đến các thành phần khác của lớp chứa nó
- Các nested class có thể khai báo với keyword public, private, protected hoặc để trống
- Với outer class thì access modifier chỉ có thể là public hoặc để trống

Lý do sử dụng nested class

- Tăng tính đóng gói dữ liệu: các nested class thường được ẩn giấu khỏi thế giới bên ngoài và có thể truy cập cả những thành phần private của lớp outer
- Đây là một cách làm hợp lý để gom nhóm những lớp chỉ sử dụng ở một nơi nào đó. Nếu một lớp chỉ hữu ích cho một lớp khác, vậy tốt hơn hơn hết là nhúng nó vào lớp đó và giữ chúng tại một chỗ
- Code dễ đọc, dễ bảo trì hơn. Việc lồng các lớp nhỏ vào lớp sẵn có cho phép ta để các lớp này gần nơi nó được sử dụng

Static nested class

- Giống như các thành phần của lớp, static nested class được liên kết với lớp chứa nó
- Do là một thành phần static của lớp nên static nested class chỉ có thể truy cập các thành phần static của outer class
- Nếu muốn truy cập và sử dụng các thành phần non-static của outer class thì phải thông qua đối tượng của outer class
- Để truy cập đến lớp static nested ta dùng tên lớp *outer_class. static_nested_class*

Ví dụ

- Khai báo và sử dụng static nested class:

```
public class Other {
    public static void main(String[] args) {
        OuterClass.StaticNestedClass object = new
OuterClass.StaticNestedClass();
        //...
    }
}

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

Inner class

- Một inner class là một nested class không chứa keyword static trong khai báo lớp
- Là một thành phần của outer class nên inner class có thể truy cập mọi thành phần của lớp outer
- Trong inner class không thể định nghĩa bất kì thành phần static nào vì inner class liên kết với đối tượng của outer class
- Đối tượng của inner class tồn tại bên trong đối tượng outer class

Inner class

- Để tạo đối tượng của inner class, trước hết ta phải tạo đối tượng của outer class:

```
public class Other {
    public static void main(String[] args) {
        OuterClass outerObject = new OuterClass();
        OuterClass.InnerClass innerObject = outerObject.new InnerClass();
        //...
    }
}

class OuterClass {
    class InnerClass {
        //...
    }
}
```

- Có hai loại inner class: local class và anonymous class

Local class

- Local class là nested class mà không phải thành phần của bất kì lớp nào
- Local class có tên rõ ràng
- Tất cả các local class đều là inner class
- Các khai báo của local class thường trong 1 khối
- Local class không chứa access modifier, không static
- Sử dụng local class khi bạn cần tạo ra một kiểu mới có tên cụ thể, muốn tạo nhiều thể hiện của lớp, truy cập các constructor của chúng…

Local class

➢Ví dụ về local class:

```
public class OuterClass {
    public void doSomething() {
        class Local { // lớp cục bộ
        }
        class OtherLocal {
            void doSomething2() {
                class Local { // lớp cục bộ
                    // do something here
                }
            }
        }
        { // bao bởi 1 khối
            // lớp cục bộ
            class Local2 {
            }
        }
    }
}
```

Anonymous class

- Anonymous class cho phép bạn tạo ra các đoạn code ngắn gọn
- Anonymous class cho phép bạn khả năng tạo đồng thời lớp và đối tượng tại 1 thời điểm
- Anonymous class giống như local class nhưng không có tên
- Sử dụng anonymous class khi bạn cần sử dụng local class chỉ 1 lần duy nhất
- Cụ thể về anonymous sẽ trình bày trong phần các bài học liên quan đến tính trừu tượng

Nội dung tiếp theo

Lớp ArrayList