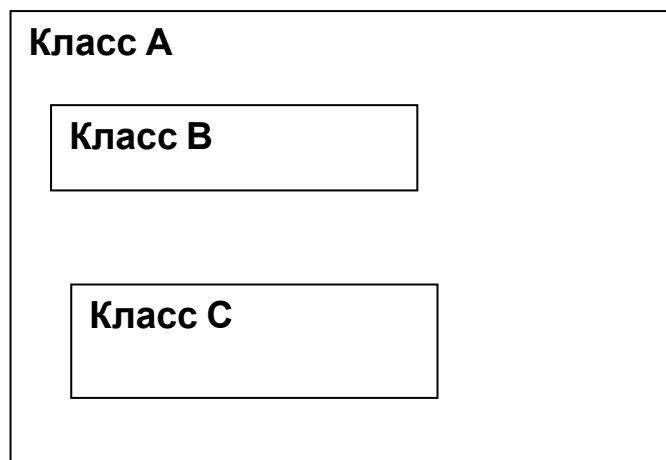


Вложенные (внутренние) классы

- Если требуется, чтобы класс А использовал все доступные методы, включая и `protected`-методы, двух классов (класса В и С), то реализовать такую схему можно через вложенный класс.



- Это хороший способ группировки классов, которые используются только в одном месте.
- Для инкапсуляции.
- Улучшение читаемости кода

Типы внутренних классов

```
class A { // внешний класс
class C { } // нестатический внутренний класс
static class B { } // статический внутренний
класс
    void f() {
        class D { } // локальный внутренний
класс
    }

    void g() { // анонимный внутренний
класс
        Base bref = new Base() {
            void method1() { }
        };
    }
}
```

Внутренние статические и нестатические классы

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

Нестатический объект вложенного класса можно создать только как экземпляр внешнего класса

```
OuterClass MyOuterClass = new OuterClass();  
OuterClass.InnerClass MyInnerClass = new MyOuterClass.InnerClass();
```

Статический объект вложенного класса можно создавать отдельно

```
OuterClass.StaticNestedClass MyStaticNestedClass =  
new OuterClass.StaticNestedClass()
```

Пример нестатического внутреннего класса

Нестатический внутренний класс имеет доступ ко всем полям и методам обрамляющего класса

```
public class OuterClass {  
    public void method() { ... }  
  
    public class InnerClass {  
        public InnerClass () { ... }  
        public void method() { ... }  
  
        public void anotherMethod() {  
            method(); // ВЫЗОВ method InnerClass  
            OuterClass.this.method() // ВЫЗОВ method OuterClass  
        }  
    }  
}
```

Создание экземпляра вложенного класса (объект компонентного класса привязан к объекту внешнего класса)

```
OuterClass oclass = new OuterClass();  
    OuterClass.InnerClass iclass = oclass.new InnerClass();  
iclass.anotherMethod();
```

Пример статического внутреннего класса

Статические внутренние классы, не имеют доступа к нестатическим полям и методам обрамляющего класса

```
class Outer3 {  
String name;  
...  
static class Inner3 {  
...  
public void f(Outer3 obj) { System.out.println(obj.name); // Здесь без obj  
нельзя  
}}  
...  
public static Inner3 createInner() { return new Inner3(); }  
...  
}  
// Объект статического класса не привязан к объекту внешнего класса  
Outer3.Inner3 obj1 = new Outer3.Inner3(); // явное порождение  
// порождение через метод createInner()  
Outer3.Inner3 obj2 = Outer3.createInner();
```

Локальные внутренние классы

Локальные классы определяются в блоке Java кода (между фигурными скобками).

У локальных классов следующие ограничения:

- они видны только в пределах блока, в котором объявлены;
- они не могут быть объявлены как `private`, `public`, `protected` или `static`;
- они не могут иметь внутри себя статических объявлений (полей, методов, классов); исключением являются константы (`static final`);

```
public class Handler {  
    public void handle(String requestPath) {  
        class LocalClass {  
            LocalClass () {...};  
            ...  
        }  
        LocalClass lc = new LocalClass();  
        ...  
    }  
}
```

Анонимные классы

Анонимный класс - это локальный класс без имени.

Использование анонимных классов :

- тело класса является очень коротким;
- нужен только один экземпляр класса;
- нельзя объявлять конструктор;
- класс используется в месте его создания или сразу после него;
- анонимные классы никогда не могут быть статическими, либо абстрактными;
- имя класса не важно и не облегчает понимание кода.

```
new Thread(new Runnable() { // анонимный класс
public void run() { ... }
}
).start();
```

Анонимные объекты

Анонимный объект – это объект, к которому нельзя обратиться извне (нет переменной, которая хранит ссылку на объект).

```
class MyClass {  
    void show(String msg){  
        System.out.println(msg);}  
}  
  
class Demo {  
    public static void main (String args[]){  
        new MyClass().show("Этот объект не имеет  
        имени");}  
}
```