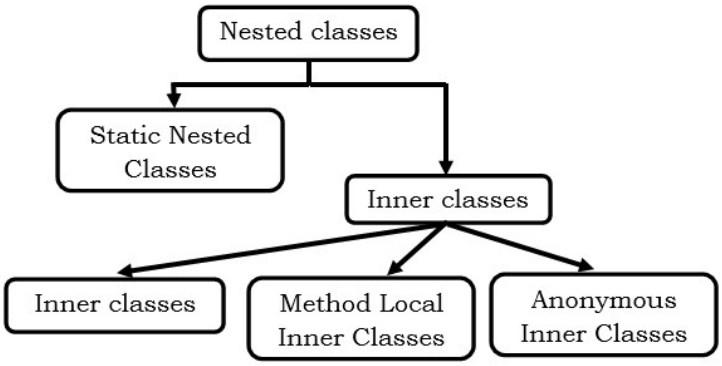
Вопросы по теме Вложенный класс:

1. Что такое вложенный класс?
2. Что такое локальные класс?
3. Что такое анонимный класс?
4. Как вложенные классы решают проблему множественного наследования в Java?
5. Чем отличаются анонимные классы, созданные на основе интерфейса и на основе класса?

Вложенный класс.

Вложенный класс – класс, определенный внутри другого класса.



Вложенный класс делят на две категории:

1. статический вложенный класс;
2. нестатический (внутренний) класс.

Нестатический вложенный класс (внутренний класс) делится на:

1. локальный класс;
2. анонимный класс.

Вложенный класс используется:

1. как способ логической группировки классов, которые используются только в одном месте (если класс полезен только для одного другого класса, то логично встроить его в этот класс и сохранить их вместе);
2. для увеличения инкапсуляции (скрывает вложенный класс от пользователей);
3. делает код более читаемым и удобным (код с разных классов расположен в одном месте);
4. позволяет частично решить вопрос множественного наследования (можно наследовать вложенный класс от нужного нам, а не основной).

Особенности статического вложенного класса:

1. объект статического вложенного класса может существовать без объекта внешнего класса;
2. статический вложенный класс может обращаться только к статическим полям внешнего класса;
3. может создать сколько угодно объектов статического вложенного класса.

Особенности внутреннего класса (нестатический):

1. объект внутреннего класса не может существовать без объекта внешнего класса;
2. объект внутреннего класса нельзя создать в статическом методе внешнего класса;
3. внутренний класс имеет доступ к ко всем членам внешнего класс, даже если они private;
4. начиная с Java 16, внутренние классы могут содержать статические поля и методы.

Локальный класс – это класс, определенный в блоке кода, обычно в теле метода. Он принадлежит не внешнему классу, а блоку кода, в котором определен. Применяется, если нужно написать класс, который будет использоваться внутри одного метода. Особенности:

1. область видимости локального класса – блок кода, в котором он определен;
2. объект локального класса не может создаваться за пределами метода или блока, в котором его объявили;
3. могут иметь только модификатор final;
4. локальный класс, объявленный в статическом блоке, имеет доступ только к статическим переменным внешнего класса, а объявленный в нестатическом – ко всем переменным.

Анонимный класс – позволяет объявить и создать экземпляр класса одновременно. Похож на локальный, только не имеет имени. Применяется, если нужно использовать локальный класс всего один раз (тогда вместо локального применяем анонимный).

В то время как локальные классы являются объявлениями классов, анонимные классы являются выражениями, что означает, что класс определяется в другом выражении. Синтаксис выражения анонимного класса аналогичен вызову конструктора, за исключением того, что определение класса содержится в блоке кода.

Выражение анонимного класса состоит из следующего:

1. оператор new;
2. имя интерфейса для реализации или класса для расширения;
3. круглые скобки, содержащие аргументы конструктора;
4. тело, которое является телом объявления класса (в конце “;”).

```
public class Main{

    interface AnonymousInnerClass {}

    public static void main(String[] args) {
        new AnonymousInnerClass() { }; // Анонимный класс

        class LocalInnerClass { } // Локальный класс
    }

    static class StaticNestedClass { } // Статический вложенный класс

    class InnerClass { } // Нестатический (внутренний) класс

}
```

Особенности анонимного класса:

1. не может иметь явно объявленного конструктора;
2. может существовать ровно один экземпляр анонимного класса;
3. не может иметь статических членов, кроме final static;
4. имеет доступ к локальным переменным, которые находятся в области блока кода, в котором объявлен анонимный класс, за пределами блока есть доступ только к static переменным;
5. анонимный класс, созданный на основе интерфейса, требует реализации всех методов этого интерфейса, созданный на основе класса не требует такого.