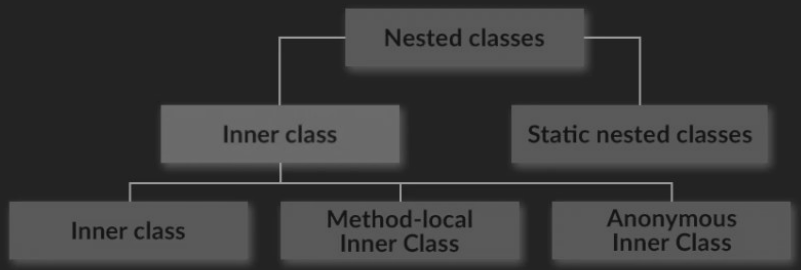
Вложенные и внутренние классы

Определение



В Java вложенные и внутренние классы предоставляют способ логически сгруппировать классы, которые используются только в одном месте, увеличивая инкапсуляцию и читаемость кода.

Вложенные классы делятся на два типа: статические вложенные классы и нестатические внутренние классы.

Вторые разделяются еще на три вида.

Контрольная точка

- Понятно ли что такое вложенные и внутренние классы?

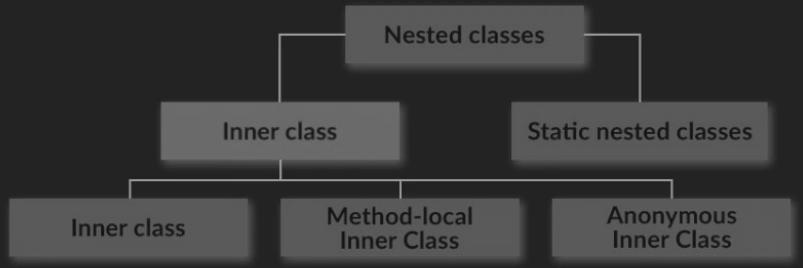
Если все ясно, ставим плюсы, иначе - задаем вопросы.

Доступность членов их классов

```java
package l28;


class OuterClass {
   private static String staticMsg = "Стат. сообщение";
   private String instanceMsg = "Сообщение экземпляра";

   // Статический вложенный класс
   static class StaticNestedClass {
       void display() {
           // Доступ к статическому члену внешнего класса
           System.out.println(staticMsg);
       }
   }

   // Внутренний класс
   class InnerClass {
       void display() {
           // Доступ к статическому члену
           System.out.println(staticMsg);
           // Доступ к нестатическому члену
           System.out.println(instanceMsg);
       }
   }
}
```

Внутренние классы имеют доступ ко всем членам (включая приватные) внешнего класса. Статические вложенные классы имеют доступ только к статическим членам внешнего класса.

Преимущества использования



Использование вложенных и внутренних классов помогает сделать ваш код более организованным и модульным.

Они позволяют близко расположить код, который тесно связан функционально, но который не должен быть доступен извне родительского класса.

Контрольная точка

- Понятно ли что доступно из вложенных классов?

Если все ясно, ставим плюсы, иначе - задаем вопросы.

Создание экземпляров

```
package l28;


class Main {
   public static void main(String[] args) {
     // Создание экземпляра
     // статического вложенного класса
     OuterClass.StaticNestedClass nestedObject;
     nestedObject = new OuterClass.StaticNestedClass();
     nestedObject.display();

     // Создание экземпляра внутреннего класса
     OuterClass outerObject = new OuterClass();
     OuterClass.InnerClass innerObject;
     innerObject = outerObject.new InnerClass();
     innerObject.display();
   }
}
```

Для создания экземпляров внутренних и вложенных классов необходимо использовать экземпляр внешнего класса (для внутренних классов) или обращаться к вложенному классу через имя внешнего класса (для статических вложенных классов).

Контрольная точка

- Понятно ли как создавать экземпляры?

Если все ясно, ставим плюсы, иначе - задаем вопросы.

Домашнее задание

Задачи минимум:

- прочитать статью https://javarush.com/groups/posts/2181-vlozhennihe-vnutrennie-klassih
- см. следующий слайд (задание можно делать два дня)
  итератор это библиотекарь который проходится по всей полке и при каждом вашем запросе называет книгу на которую сейчас смотрит (должен хранить индекс последней названной книги, начинает с индекса 0, при достижении конца - возвращается к началу)

Домашнее задание: "Книжная полка"

```java
import java.util.ArrayList;
import java.util.List;

class BookShelf {
   private List<Book> books = new ArrayList<>();

   static class Book {
       String title;
       String author;
       int year;

       Book(String title, String author,  int year) {
           this.title = title;
           this.author = author;
           this.year = year;
       }

       @Override
       public String toString() {
           return title + " by " + author + ", " + year;
       }
   }

   void addBook(String title, String author,  int year) {
       books.add(new Book(title, author, year));
   }

   // Методы removeBook и toString опущены для краткости
}
```

Задача:

Разработайте систему для управления коллекцией книг, используя вложенные и внутренние классы в Java. Система должна позволять добавлять книги на виртуальную книжную полку и удалять их оттуда.

Шаги:

1. Класс BookShelf: Создайте внешний класс BookShelf, который будет представлять книжную полку. В этом классе должен быть ArrayList для хранения книг.
2. Класс Book: Определите статический вложенный класс Book внутри BookShelf. Класс Book должен содержать базовую информацию о книге, такую как название, автор и год издания.
3. Управление книгами: В классе BookShelf реализуйте методы для добавления книг (addBook) и удаления книг (removeBook) с полки. Также реализуйте метод для вывода информации о всех книгах на полке.
4. Внутренний класс Iterator: Создайте нестатический внутренний класс Iterator в BookShelf, который позволит итерироваться по книгам на полке.
5. Демонстрация в main методе: В методе main создайте экземпляр BookShelf, добавьте несколько книг и затем удалите одну. Используйте итератор для вывода списка оставшихся книг.

The end