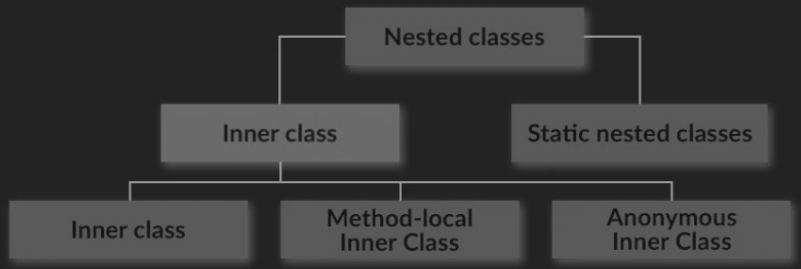
# Анонимные классы и лямбда выражения

# Анонимный класс




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.

# Анонимный класс

```java
package l31.s1;

interface Greeting {
    void sayHello();
}

class Main {
    public static void main(String[] args) {
        Greeting greeting = new Greeting() {
            @Override
            public void sayHello() {
              System.out.println("Привет!");
            }
        };
        greeting.sayHello();
    }
}
```

Анонимный класс - это класс без названия, объявленный и инстанциированный в одном выражении.

Они обычно используются для расширения существующих классов или реализации интерфейсов на месте.

Анонимные классы и лямбда-выражения в Java позволяют упростить код, особенно когда речь идет о создании экземпляров классов с минимальным количеством методов или реализации интерфейсов на лету.

# Контрольная точка

- Понятно ли что такое анонимные классы?
- Понятно ли что как их писать?
- Понятно ли что они могут быть создагны на основе интерфейса, абстрактного и обычного классов?
- Понятно ли что анонимный класс может быть использован для создания объекта только в том месте где он задан?

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

# Применение анонимного класса

```java
package l31.s1;

public class Main2 {
   public static void main(String[] args) {
       // Пример использования анонимного класса
       // для обработки нажатия кнопки в программе
       // с графическим интерфейсом
       button.addActionListener(new ActionListener() {
           public void actionPerformed(ActionEvent e) {
               System.out.println(
                 "Кнопка нажата, привет вам из 11:58"
               );
           }
       });
   }
}
```

Создаем и сразу пеередаем в аргументы

# Функциональные интерфейсы

```java
package l31.s2;

@FunctionalInterface
interface Calculator {
    int operate(int a, int b);
}
```

Функциональный интерфейс в Java - это интерфейс с одним абстрактным методом.

Они предназначены для использования с лямбда-выражениями и анонимными классами.

Этот интерфейс содержит один абстрактный метод operate, который принимает два аргумента и возвращает результат.

Аннотация @FunctionalInterface не обязательна, но она помогает гарантировать, что интерфейс соответствует требованиям функционального интерфейса.

# Лямбда-выражения

```
package l31.s2;

@FunctionalInterface
interface Calculator {
   int operate(int a, int b);
}

public class Main {
   public static void main(String[] args) {
       Calculator add = (a, b) -> a + b;
       Calculator subtract = (a, b) -> a - b;

       System.out.println(
         "Сумма 5 и 3 равна "+
         add.operate(5, 3)
       );
       System.out.println(
         "Разность 5 и 3 равна "+
         subtract.operate(5, 3)
       );
   }
}
```

Лямбда-выражения предоставляют синтаксически компактный способ реализации функциональных интерфейсов, позволяя пропускать объявление класса и имя метода.

В этом примере мы создаем два экземпляра функционального интерфейса Calculator с использованием лямбда-выражений для реализации операций сложения и вычитания.

Лямбда-выражения (a, b) -> a + b и (a, b) -> a - b представляют собой реализации метода operate.

# Использование

```java
package l31.s3;

import java.util.Arrays;
import java.util.List;

class Main {
 public static void main(String[] args) {
     // Пример использования лямбда-выражений
     // с потоками данных
     List<String> strings = Arrays.asList(
       "one", "two", "three"
     );
     // Создание потока данных из списка строк
     strings.stream()
         // Фильтрация строк, начинающихся с "t"
         .filter(s -> s.startsWith("t"))
         // Вывод отфильтрованных строк
         .forEach(System.out::println);

   }
}
```

Использование в потоках данных, ссылки на методы и составные лямбда-выражения.

# Контрольная точка

- Понятно ли что такое лямбда выражения?

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

# Домашнее задание

Материалы:

- https://metanit.com/java/tutorial/9.1.php
- https://javarush.com/groups/posts/845-lambda-vihrazhenija-na-primerakh
- https://www.w3schools.com/java/java_lambda.asp

Задачи минимум:

- Написать функциональный интерфейс (l31.Calculator)
- Написать анонимный класс на основе этого интерфейса (l31.Main30)
- Написать лямбда-выражение на основе этого интерфейса (см l31.Main3)


# The end