aboba

Содержание

Введение

Современные реалии требуют обработки большого количества данных. При этом мощности компьютеров постепенно выхрдят на плато и нарушается закон Мура[1]. Что ставит перед человеком вопрос: "Как можно обработать гиганские массивы данных в относительно небольшой интервал времени?".

Решением этой проблемы является распараллеливание вычислений. Такой метод позволяет распределить вычисления на множество ЭВМ, что, при правильной настройке, позволяет сократить время выполнения программ в разы.

Но это только верхушка айзберга. Для достижения этой цели требуется пройти путь с самого низа.

Рассмотрим ситуацию: имеется фреймворк для матмоделирования. Каждая матмодель задаётся своим набором параметров и системой уравнений. Для обработки такого поведения требуется механизм, который сможет принять метод, его параметры и произвести необходимыевычисления.

И он есть - делегаты.

Цель лабораторной работы: изучить возможности применения делегатов в языке C# / Java. Задачи лабораторной работы:

— освоить принципы работы с делегатами;

— освоить основные направления применения делегатов;

— изучить способы использования делегатов совместно с потоками.

1 Выполнение работы

1.1 Код программы

Для выполнения лабораторной работы был выбран ЯП Java [2].

```
1  package ru.happines;
2
3  import java.util.function.Function;
4
5  @FunctionalInterface
6  interface Delegate {
7      void invoke(Function<Double, Double> f, double a, double b);
8  }
9
10 public class aboba {
11     public static void sum(Function<Double, Double> f, double a,
           double b) {
12         System.out.println("Sum = " + (f.apply(a) + b));
13     }
14
15     public static void multiply(Function<Double, Double> f, double a,
           double b) {
16         System.out.println("Multiply = " + (f.apply(a) * b));
17     }
18
19     public static void main(String[] args) {
20         Delegate del;
21
22         del = aboba::sum;
23         del.invoke(x -> x * x, 3, 4);
24
25         del = aboba::multiply;
26         del.invoke(x -> x + 2, 3, 4);
27     }
28 }
```

1.2 Описание работы программы

— Создан функциональный интерфейс `Delegate` с методом `invoke`, принимающим функцию и два параметра типа `double`.

— Методы `sum` и `multiply` принимают функцию и два числа, выполняют соответствующую операцию и выводят результат.

— В методе `main` создается делегат `del`, которому присваиваются ссылки на методы `sum` и `multiply`.

— Вызов `del.invoke` выполняет соответствующую операцию с функцией, переданной в качестве аргумента.

1.3 Результаты работы программы

При запуске программы на консоли выводится:

```
Sum = 13.0
Multiply = 20.0
```

Выводы

В ходе лабораторной работы были изучены следующие аспекты работы с делегатами в Java:

— Делегаты позволяют ссылаться на методы и вызывать их динамически.

— Функциональные интерфейсы в Java предоставляют удобный способ реализации делегатов.

— Применение делегатов повышает гибкость и расширяемость кода.

Список использованных источников

1. *Зубкова, В. В.* Анализ актуальности закона Мура / В. В. Зубкова // *Перспективы развития информационных технологий*. — 2014. — no. 21. — Дата обращения: 06.10.2025. URL: https://cyberleninka.ru/article/n/analiz-aktualnosti-zakona-mura.

2. *Сьерра, К.* Изучаем Java: [пер. с англ.] / К. Сьерра, Б. Бейтс. Мировой компьютерный бестселлер. — Эксмо, 2012. URL: https://books.google.ru/books?id=qEZ1kQEACAAJ.