Вкладка 1

Отчет о лабораторной работе №6

Выполнил:
Петров Евгений Станиславович
Группа: 6204-010302D

Вкладка 2

ВВЕДЕНИЕ

Цель: Разработать приложение, формирующее в одном потоке вычислений набор заданий для интегрирования, а во втором потоке – вычисляющее значения интегралов функций.

В ходе выполнения работы необходимо расширить библиотеку функций численным интегратором, реализовать последовательную и многопоточную генерацию и обработку заданий, обеспечить корректный обмен данными между потоками, корректную обработку прерываний, а также подготовить демонстрационный запуск реализованных решений.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1. *Метод трапеций*
   Метод трапеций основан на разбиении отрезка интегрирования на подотрезки длиной *h*. Площадь под графиком функции аппроксимируется суммой площадей трапеций и вычисляется по формуле:
   $\sum (f(x_i) + f(x_{i+1})) / 2 \cdot \Delta x$.
   Если длина последнего подотрезка меньше шага *h*, он интегрируется с использованием собственной ширины.
2. *Область определения функции*
   Численное интегрирование корректно только внутри области определения функции. Попытка интегрирования за пределами домена должна приводить к ошибке или исключению.
3. *Потоки и разделяемое состояние*
   Одновременный доступ нескольких потоков к общему объекту без синхронизации может приводить к гонкам данных, несогласованным чтениям и исключениям времени выполнения, таким как `NullPointerException`. Использование синхронизированных блоков или семафоров позволяет исключить одновременную запись и чтение разделяемых данных.
4. *Одноместный семафор*
   Одноместный семафор позволяет различать стадии «запись задания» и «чтение задания». Это предотвращает потерю данных и перемешивание параметров задания при передаче между потоками генерации и интегрирования.

МЕТОДИКА И РЕАЛИЗАЦИЯ

Среда выполнения и структура проекта

Язык реализации — Java. Рекомендуемая версия JDK — 17 и выше.

Структура проекта включает следующие элементы:

- пакет `functions` — базовые и табулированные функции, а также численный интегратор `Functions.integrate`;
- пакет `functions.basic` — элементарные функции (экспонента, логарифм, тригонометрические функции);
- пакет `functions.meta` — композиции и арифметические операции над функциями;
- пакет `threads` — классы обмена заданиями между потоками (`Task`, генераторы, интеграторы, семафор);
- класс `Main` — сценарии проверки заданий 1–4.

Реализация по этапам

1. *Численное интегрирование (Задание 1)*
   Реализован метод `Functions.integrate(Function f, double left, double right, double step)`, который проверяет корректность шага и принадлежность границ области определения функции. Интегрирование выполняется методом трапеций с учётом неполного последнего шага. В методе `Main.integrationCheck` вычисляется интеграл функции exp(x) на отрезке [0; 1] с подбором шага, обеспечивающего точность не хуже $1 \cdot 10^{-7}$.
2. *Последовательная версия (Задание 2)*
   Метод `Main.nonThread` создаёт объект `Task` с заданным количеством задач. Генерируются логарифмические функции со случайным основанием от 1 до 10, случайные границы интегрирования из диапазонов [0; 100] и [100; 200], а также шаг интегрирования из интервала (0; 1]. Для каждой задачи выводятся строки вида `Source ...` и `Result ...`, после чего интеграл вычисляется численным методом.
3. *Простая потоковая модель (Задание 3)*
   Поток `SimpleGenerator` реализует интерфейс `Runnable`, заполняет объект `Task` и выводит исходные данные задачи. Поток `SimpleIntegrator` считывает снимки состояния `Task`, ожидая появления корректных данных, и вычисляет результат

интегрирования.
Для предотвращения смешивания полей используется синхронизация внутри методов `Task.update` и `Task.snapshot`.

4. *Семафорная модель и обработка прерываний (Задание 4)*
Реализован одноместный семафор `OnePlaceSemaphore`, обеспечивающий корректное чередование операций записи и чтения.
Потоки `Generator` и `Integrator` наследуются от `Thread`, используют семафор для передачи задания, корректно обрабатывают вызов `interrupt()` и выполняют короткие паузы (`sleep(1)`) для наглядной демонстрации чередования выполнения.
Метод `Main.complicatedThreads` запускает пару потоков, ожидает 50 мс и затем завершает их с помощью прерывания.

РЕЗУЛЬТАТЫ

В результате выполнения лабораторной работы:

- реализован численный интегратор методом трапеций с проверкой области определения функции;
- последовательный сценарий корректно генерирует и решает не менее 100 логарифмических интегралов;
- потоковая версия с синхронизацией устраняет `NullPointerException` и несогласованные состояния данных;
- семафорная модель предотвращает потерю заданий и корректно обрабатывает прерывания потоков;
- продемонстрирована сходимость метода трапеций на примере интеграла exp(x) на отрезке [0; 1] с точностью лучше $1 \cdot 10^{-7}$.

ВЫВОДЫ

Поставленные цели лабораторной работы достигнуты. Реализован численный интегратор и исследованы три модели взаимодействия потоков: без синхронизации, с использованием синхронизации и с применением семафора. Использование синхронизации и семафоров позволило устранить гонки данных и потери заданий. Метод трапеций

продемонстрировал ожидаемую сходимость при уменьшении шага интегрирования.

ТЕСТИРОВАНИЕ И ПРОВЕРКА

Функциональная проверка выполняется с помощью сценариев `integrationCheck`, `nonThread`, `simpleThreads` и `complicatedThreads`, реализованных в классе `Main`.

В коде предусмотрены проверки граничных условий: положительность шага интегрирования, корректность границ области определения функции и защита от передачи `null`-функций.

Рекомендуется после установки JDK выполнить компиляцию и запуск проекта и убедиться, что консольный вывод содержит ожидаемые пары строк `Source` и `Result`.

КЛЮЧЕВЫЕ КЛАССЫ

- `functions.Functions.integrate` — численное интегрирование методом трапеций;
- `threads.Task` — контейнер параметров задания;
- `threads.SimpleGenerator`, `threads.SimpleIntegrator` — базовая потоковая модель;
- `threads.OnePlaceSemaphore`, `threads.Generator`, `threads.Integrator` — семафорная модель с корректным завершением по прерыванию;
- `Main` — точка входа и сценарии демонстрации заданий 1–4.

СПИСОК ЛИТЕРАТУРЫ

1. Документация Oracle по многопоточности в Java. Java Concurrency Tutorial.