Лабораторная работа №7

Задание на лабораторную работу

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

Official disclaimer (официальное уведомление). Некоторые элементы дизайна, навязанные в задании, являются не очень корректными (например, вместо одного объекта задания, используемого разными нитями, в данном случае разумно использовать очередь заданий, и т.д.). Однако корректная реализация и её исследование по своей сложности выходят за рамки одной лабораторной работы.

Задание 1 *(0,4 балла)*

Добавьте в класс `Functions` метод, возвращающий значение интеграла функции, вычисленное с помощью численного метода.

В качестве параметров метод должен получать ссылку типа `Function` на объект функции, значения левой и правой границы области интегрирования, а также шаг дискретизации.

Если интервал интегрирования выходит за границы области определения функции, метод должен выбрасывать исключение.

Вычисление значения интеграла должно выполняться по методу трапеций. Для этого вся область интегрирования разбивается на участки, длина которых (кроме одного) равна шагу дискретизации. На каждом таком участке площадь под кривой, описывающейся заданной функцией, приближается площадью трапеции, две вершины которой расположены на оси абсцисс на границах участка, а ещё две – на кривой в точках границ участка. Обратите внимание на то, что в область интегрирования необязательно укладывается целое количество шагов дискретизации.

В методе `main()` проверьте работу метода интегрирования. Для этого вычислите интеграл для экспоненты на отрезке от 0 до 1. Определите также, какой шаг дискретизации нужен, чтобы рассчитанное значение отличалось от теоретического в 7 знаке после запятой.

Задание 2 *(1 балл)*

Далее в приложении один поток инструкций будет генерировать задачи для интегрирования, а второй поток – решать их. Для этого потребуется объект задания, через который эти потоки будут взаимодействовать. В объекте будут храниться параметры задания.

Создайте пакет `threads`, в котором будут размещены классы, связанные с потоками.

В пакете `threads` опишите класс `Task`, объект которого должен хранить ссылку на объект интегрируемой функции, границы области интегрирования, шаг дискретизации, а также целочисленное поле, хранящее количество выполняемых заданий. Т.е. объект описывает одно задание.

В главном классе программы опишите метод `nonThread()`, реализующий последовательную (без применения потоков инструкций) версию программы. В методе необходимо создать объект класса `Task` и установить в нём количество выполняемых заданий (минимум 100). После этого в цикле (до количества заданий) нужно выполнить следующие действия:

1) создать и поместить в объект задания объект логарифмической функции, основание которой является случайной величиной, распределённой равномерно на отрезке от 1 до 10;
2) указать в объекте задания левую границу области интегрирования (случайно распределена на отрезке от 0 до 100);

3) указать в объекте задания правую границу области интегрирования (случайно распределена на отрезке от 100 до 200);
4) указать в объекте задания шаг дискретизации (случайно распределён на отрезке от 0 до 1);
5) вывести в консоль сообщение вида "`Source <левая граница> <правая граница> <шаг дискретизации>`";
6) вычислить значение интеграла для параметров из объекта задания;
7) вывести в консоль сообщение вида "`Result <левая граница> <правая граница> <шаг дискретизации> <результат интегрирования>`".

Проверьте работу метода, вызвав его в методе `main()`.

Задание 3 *(1,8 балла)*

В пакете `threads` создайте два следующих класса. При их реализации воспользуйтесь фрагментами последовательной версии программы из предыдущего задания.

Класс `SimpleGenerator` должен реализовывать интерфейс `Runnable`, получать в конструкторе и сохранять в своё поле ссылку на объект типа `Task`, а в методе `run()` в цикле должны формироваться задачи и заноситься в полученный объект задания, а также выводиться сообщения в консоль.

Класс `SimpleIntegrator` должен реализовывать интерфейс `Runnable`, получать в конструкторе и сохранять в своё поле ссылку на объект типа `Task`, а в методе `run()` в цикле должны решаться задачи, данные для которых берутся из полученного объекта задания, а также выводиться сообщения в консоль.

В главном классе программы создайте метод `simpleThreads()`. В нём создайте объект задания, укажите количество выполняемых заданий (минимум 100), создайте и запустите два потока вычислений, основанных на описанных классах `SimpleGenerator` и `SimpleIntegrator`. Проверьте работу метода, вызвав его в методе `main()`.

Попробуйте запускать программу (несколько раз).

Попробуйте запускать программу, изменяя приоритеты потоков перед их запуском.

Убедитесь, что в некоторых случаях интегрирующий поток может прекращать своё выполнение из-за исключения `NullPointerException`. Определите причину возникновения этого исключения и сделайте так, чтобы оно не возникало (используйте какое-нибудь простое решение, без синхронизации и нотификаций).

Попробуйте запускать программу (несколько раз).

Убедитесь, что в некоторых случаях интегрирующий поток выводит сообщение с набором данных, который не встречается в сообщениях генерирующего потока (например, левая граница области интегрирования оказывается взята из одного задания, а правая граница и шаг дискретизации – из другого). Определите причину возникновения таких ситуаций и устраните её, используя блоки синхронизации в методах `run()` генерирующего и интегрирующего потоков. Убедитесь, что в коде нигде не возникает необоснованно длительной блокировки объектов.

Задание 4 *(1,8 балла)*

Определите причину того, что не все сгенерированные задания оказываются выполнены интегрирующим потоком.

Для устранения этой проблемы потребуется дополнительный объект, представляющий собой одноместный семафор, различающий операции чтения и записи в защищаемый объект (пример класса таких объектов был приведён в лекции).

В пакете `threads` создайте два следующих класса. При их реализации воспользуйтесь фрагментами классов из предыдущего задания.

Класс `Generator` должен расширять класс `Thread`, получать в конструкторе и сохранять в свои поля ссылки на объект типа `Task` и на объект семафора, а в методе `run()` должны выполняться те же действия, что и в предыдущей версии генерирующего класса.

Класс `Integrator` должен расширять класс `Thread`, получать в конструкторе и сохранять в свои поля ссылки на объект типа `Task` и на объект семафора, а в методе `run()` должны выполняться те же действия, что и в предыдущей версии интегрирующего класса.

Отличие этих классов от предыдущих версий должно заключаться в том, что вместо средств синхронизации в методах `run()` должны использоваться возможности семафора.

В главном классе программы создайте метод `complicatedThreads()`. В нём создайте объект задания, укажите количество выполняемых заданий (минимум 100), создайте и запустите два потока вычислений классов `Generator` и `Integrator`. Проверьте работу метода, вызвав его в методе `main()`.

Попробуйте запускать программу (несколько раз).

Попробуйте запускать программу, изменяя приоритеты потоков перед их запуском.

Сделайте так, чтобы после создания ваших потоков основной поток программы выжидал 50 миллисекунд, после чего прерывал работу ваших потов путём вызова метода `interrupt()`. Измените код ваших классов потоков так, чтобы прерывание происходило и происходило корректно.