Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение высшего образования

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ

Кафедра «Математической кибернетики и информационных технологий»

ИТиП

Лабораторная работа №7

Выполнил:

студент группы БВТ2203

Петухов Г. В.

Москва 2023

**Задание 1.**

Реализация многопоточной программы для вычисления суммы элементов массива.

**Вариант 1.**

Создать два потока, которые будут вычислять сумму элементов массива по половинкам, после чего результаты будут складываться в главном потоке.

**Задание 2.**

Реализация многопоточной программы для поиска наибольшего элемента в матрице.

**Вариант 1**.

Создать несколько потоков, каждый из которых будет обрабатывать свою строку матрицы. После завершения работы всех потоков результаты будут сравниваться в главном потоке для нахождения наибольшего элемента.

**Задание 3:**

У вас есть склад с товарами, которые нужно перенести на другой склад. У каждого товара есть свой вес. На складе работают 3 грузчика. Грузчики могут переносить товары одновременно, но суммарный вес товаров, которые они переносят, не может превышать 150 кг. Как только грузчики соберут 150 кг товаров, они отправятся на другой склад и начнут разгружать товары. Напишите программу на Java, используя многопоточность, которая реализует использование Lock и Condition: Используйте блокировки и условия для синхронизации работы грузчиков.

**Реализация**

**1)** Для реализации первой задачи создадим класс ArraySum(Рис. 1) наследующий класс Threads. Добавим поля с массивом, начальным и конечным индексом а также перепишем метод run так, чтобы он вычислял сумму элементов от начального до конечного индекса. Для демонстрации работы создадим отдельный метод (Рис. 2), результат работы (Рис. 3).



Рисунок 1 – Листинг класса

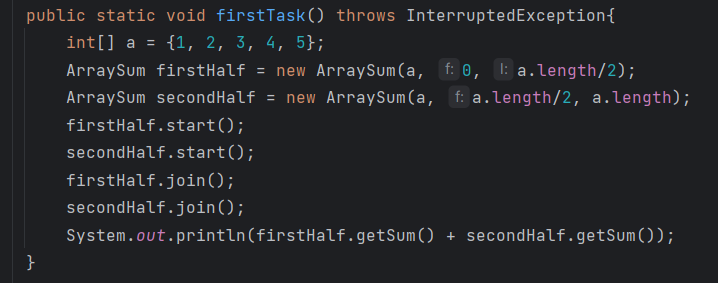


Рисунок 2

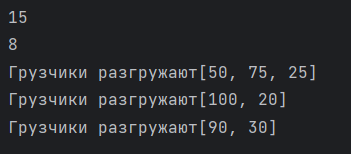


Рисунок 3 – Результат работы

2) Для реализации второй задачи реализуем класс ArrayMax(Рис. 4) в котором переопределим метод run так, чтобы он находил максимальный элемент в массиве. Далее создадим метод (Рис. 5), который для каждой строки матрицы будет создавать поток в котором будет находится максимум для каждой строки. Далее будет ожидаться пока все эти потоки завершат свою работу и найденные максимумы будут записаны в отдельный массив, для которого создастся еще один поток, в котором найдут максимум из максимумов. Результат работы рис. 3.



Рисунок 4 – Листинг класса ArrayMax

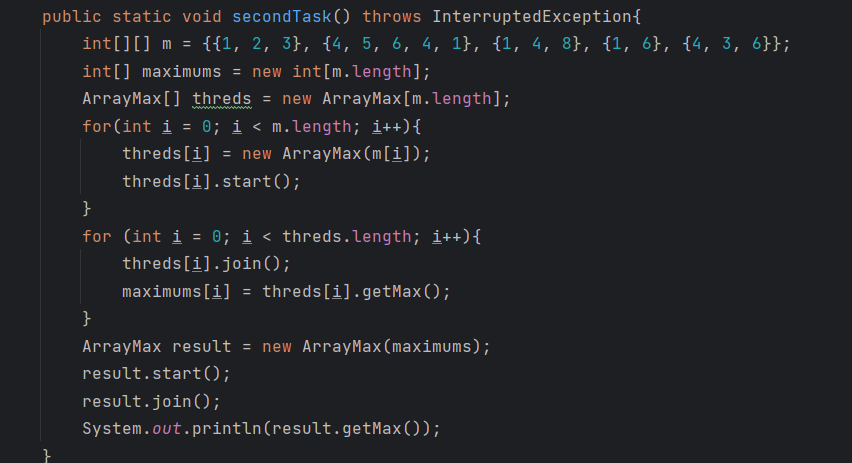


Рисунок 5 – Листинг метода

3) Для реализации третей задачи создадим класс Worker, который наследует Thread. Перепишем метод run так, чтобы он ожидал пока общий вес не станет больше 150 и только после этого сигнализировал другим что нужно разгрузить товар на другой склад. Для демонстрации работы был создан метод (Рис. 7), в котором созданы три рабочих, которые разгружают груз и получают товары по очереди пока не наберется 150 кг. Результат работы рис. 3.

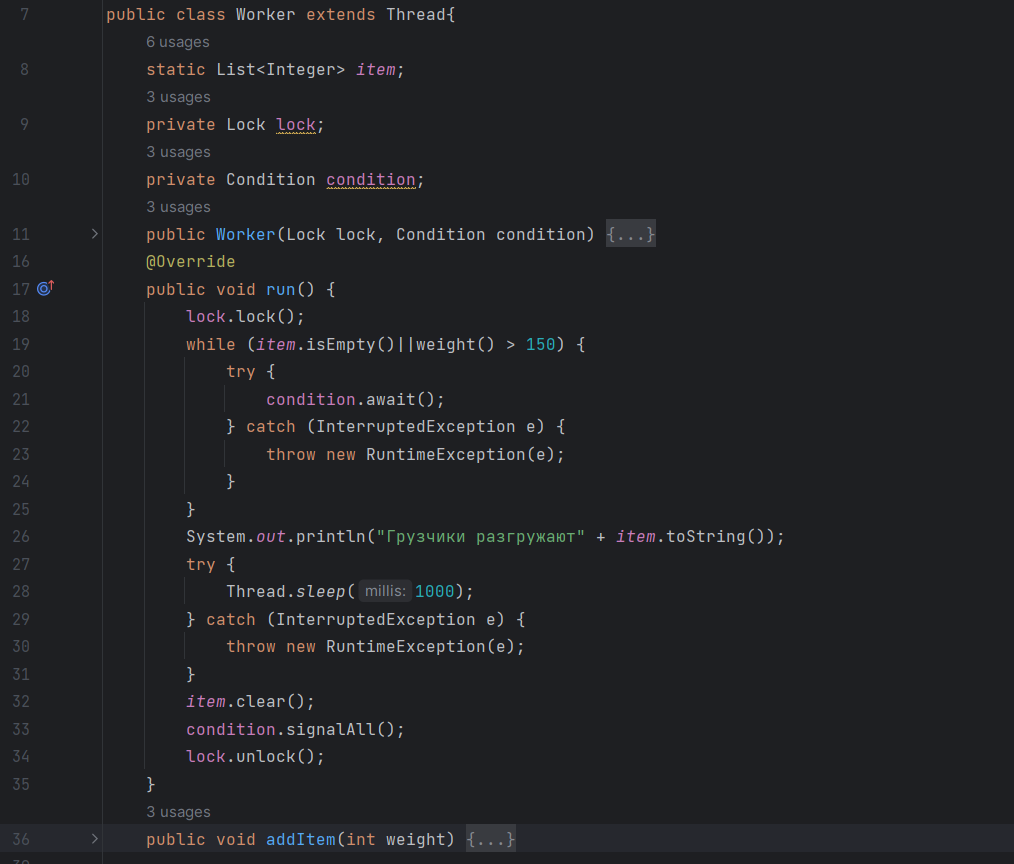


Рисунок 6 – Листинг Worker

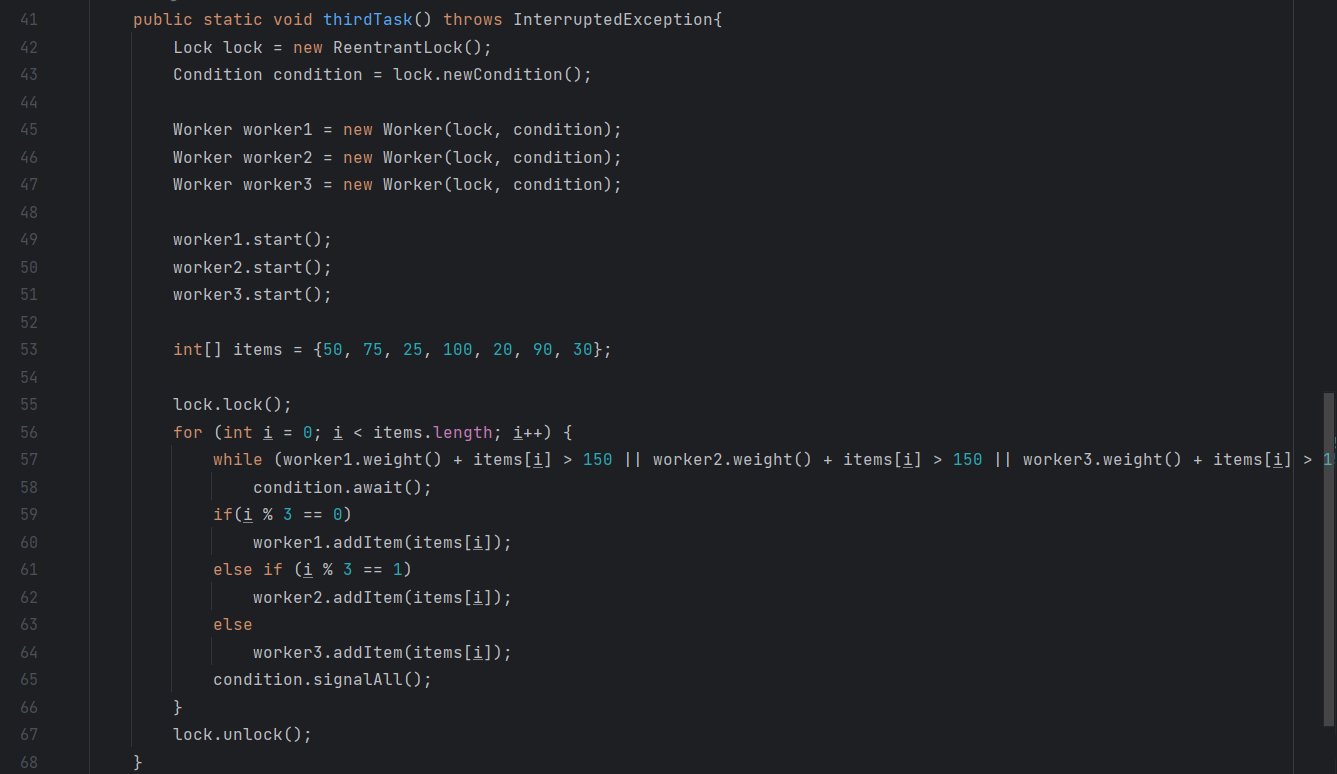


Рисунок 7