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»

Информационные технологии и программирование

Лабораторная работа №7

Выполнил: студент группы

БВТ2306

Кесслер Алексей Сергеевич

Москва, 2024 г.

Цель работы: Изучение многопоточности в Java, основы использования классов, связанных с многопоточностью.

Задача работы: Реализовать свои классы потоков.

Выполнение

Задание 1 : В первом задании необходимо реализовать многопоточную программу для вычисления суммы элементов массива при помощи двух потоков (1 вариант).

Для начала создадим свой класс-поток MyThread. В нем реализуем такие поля, как: сумма, начальный индекс массива, конечный индекс массива, исходный массив. В методе run пропишем механизм сложения всех значений массива с индексов от старта, до финиша.

```
class MyThread extends Thread {
    2 usages
    private Sum sum;
    4 usages
    private int start;
    2 usages
    private int finish;
    2 usages
    private int[] array;
    2 usages  M1NT
    public MyThread(Sum sum, int start, int finish, int[] array) {
        this.sum = sum;
        this.start = start;
        this.finish = finish;
        this.array = array;
    }
    M1NT
    public void run() {
        while (start != finish) {
            sum.addNumber(array[start]);
            start++;
        }
    }
}
```

Рисунок 1 Класс MyThread

Также для данного класса понадобится другой класс, в котором будет храниться текущая сумма.

```
class Sum {
    2 usages
    private int nowSum;
    1 usage   M1NT
    public void addNumber(int number) { nowSum += number; }

    1 usage   M1NT
    public int getSum() { return nowSum; }
}
```

Рисунок 2 Класс Sum

После необходимо создать два потока, раздели массив пополам с помощью индексов.

```
public class ArraySum {
    2 usages   M1NT
    public static int arraySum(int[] numbers) {
        Sum sum = new Sum();
        int middle = numbers.length / 2;
        MyThread thread1 = new MyThread(sum, start: 0, middle, numbers);
        MyThread thread2 = new MyThread(sum, middle, numbers.length, numbers);
        thread1.start();
        thread2.start();
        try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return sum.getSum();
    }
}
```

Рисунок 3 - Метод ArraySum

Задание 2: Во втором задании необходимо реализовать многопоточную программу для поиска наибольшего элемента в матрице с помощью пула потоков (2 вариант, класс ExecutorService)

Для данного задания создаем пул потоков с помощью Executors.newFixedThreadPool, где количество потоков совпадает с количеством строк. Далее в каждом потоке необходимо найти максимальный элемент строки и добавить его в новый список. После выполнения всех потоков находим максимальное число из всех полученных результатов.

```
public class MaxNumber {
    1 usage
    public static int maxNumber(int[][] array) throws InterruptedException {
        int rows = array.length;
        ExecutorService service = Executors.newFixedThreadPool(rows);
        List<Integer> list = new ArrayList<>();
        for (int i = 0; i < rows; i++) {
            int index = i;
            service.execute(new Runnable() {
                @Override
                public void run() {
                    int maxNum = array[index][0];
                    for (int j = 1; j < array[index].length; j++) {
                        maxNum = Math.max(maxNum, array[index][j]);
                    }
//                    System.out.println("Thread " + index);
                    list.add(maxNum);
                }
            });
        }
        service.shutdown();
        service.awaitTermination( timeout: 4, TimeUnit.SECONDS);
//        System.out.println(Arrays.toString(list.toArray()));
        int answer = list.getFirst();
        for (int i = 1; i < list.size(); i++) {
            answer = Math.max(answer, list.get(i));
        }
        return answer;
    }
}
```

Рисунок 6 - Второе задание

3 задание: В третьем задании необходимо создать программу по перевозке груза из одного склада на другой с учетом ограничения на 150 кг веса и 3-х грузчиков, используя класс Thread, классы для товаров, складов и грузчиков (1 вариант).

В данном задании создадим 4 класса – продукты, склады, грузчики и машина.

В классе продуктов есть только 1 поле – вес продукта.

```
public class Good {
    6 usages
    int weight;

    9 usages
    public Good(int weight) {
        this.weight = weight;
    }

}
```

Рисунок 9 - Класс Good

В классе склада находится поле - список из товаров и методы для взятия товара со склада, доставки товара на склад и проверки на пустоту склада.

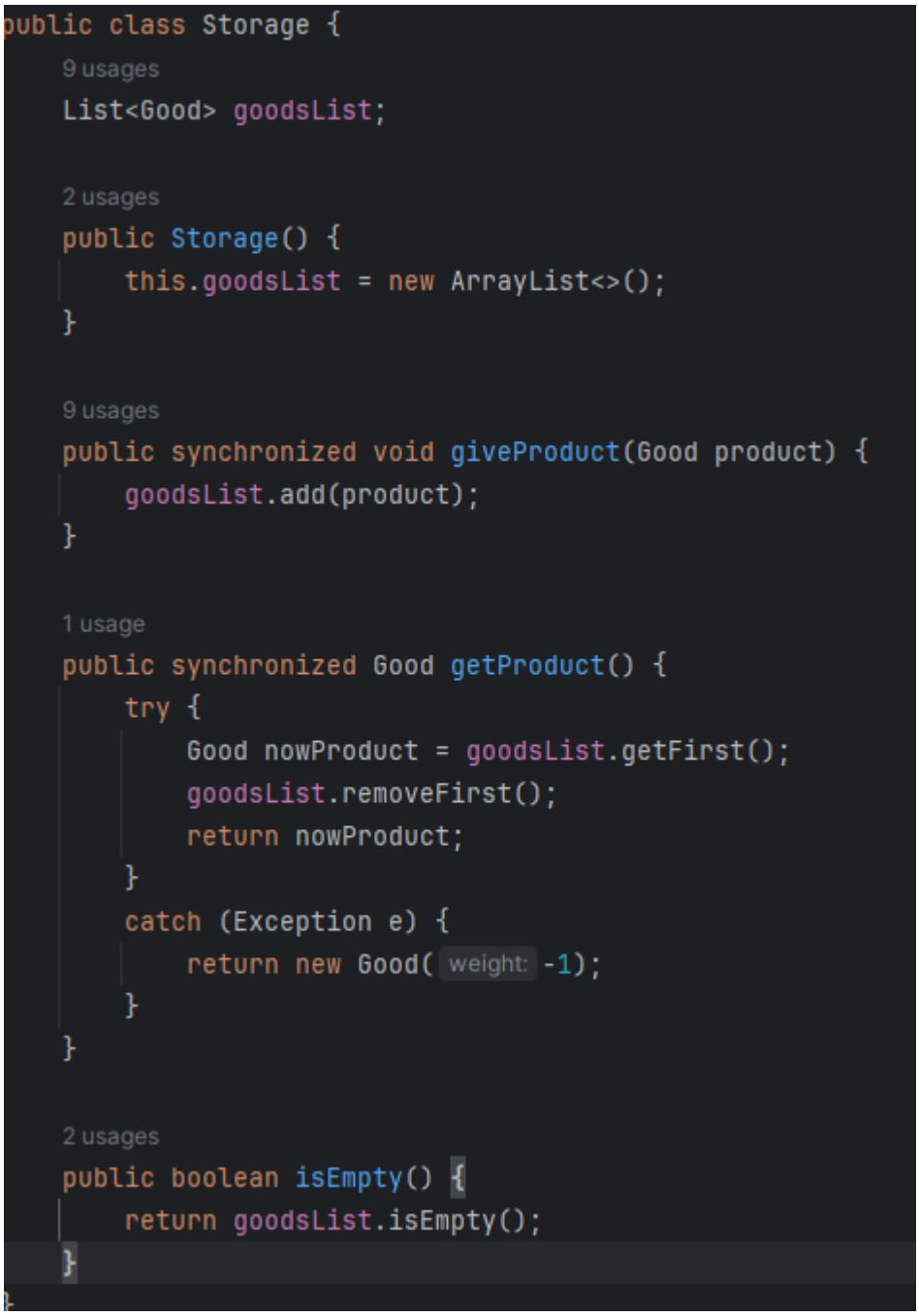

Рисунок 10 - Класс Storage

Класс Car похож на класс склада, но для него также необходимо учитывать вес хранимого товара.

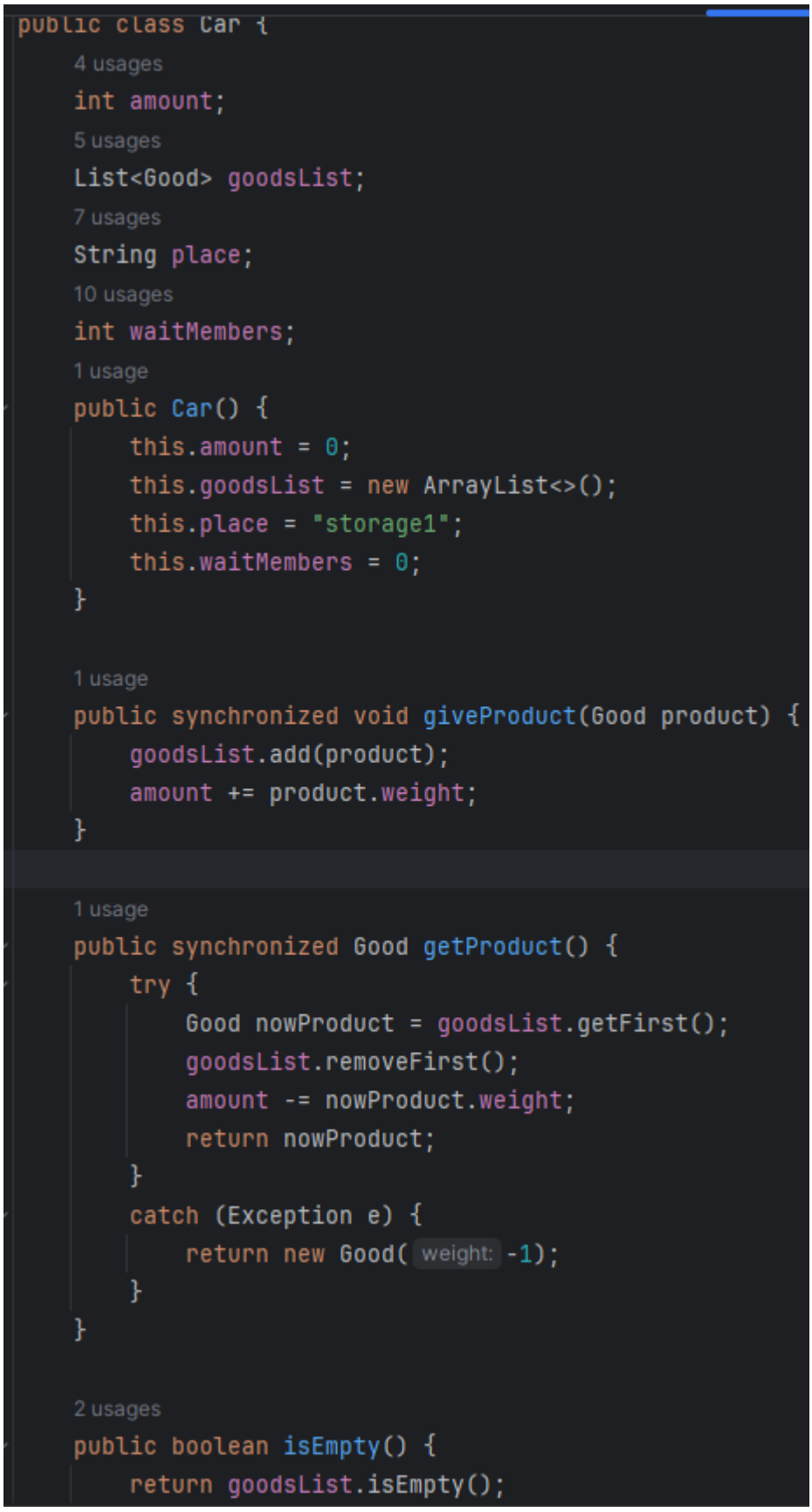

Рисунок 11 - Класс Car

Класс грузчиков мы наследуем от Thread и реализуем логику перетаскивания товара.

```
public class Mover extends Thread {
    5 usages
    Storage storage1;
    2 usages
    Storage storage2;
    21 usages
    Car car;
    1 usage
    int index;
    5 usages
    boolean waiting = false;
    3 usages
    String waitPlace;
    3 usages
    public Mover(Storage storage1, Storage storage2, Car car, int index) {
        this.storage1 = storage1;
        this.storage2 = storage2;
        this.car = car;
        this.index = index;
        this.waitPlace = "storage1";
    }
    public void run() {
        while (!(storage1.isEmpty() && car.isEmpty())) {
            if (!waiting) {
                Good hand;
                if (car.place.equals("storage1")) {
                    if (!storage1.isEmpty()) {
                        hand = storage1.getProduct();
                        if (hand.weight != -1) {
                            if (hand.weight + car.amount <= 150) {
                                car.giveProduct(hand);
//                                System.out.println("Грузчик " + index + " положил в машину предмет весом " + hand.weight);
                            } else {
                                waiting = true;
                                storage1.giveProduct(hand);
                                car.waitMembers += 1;
                                if (car.waitMembers >= 3) {
                                    car.waitMembers = 0;
```

Рисунок 12 - Класс Mover (1)

```
                        } else {
                            waiting = true;
                            storage1.giveProduct(hand);
                            car.waitMembers += 1;
                            if (car.waitMembers >= 3) {
                                car.waitMembers = 0;
                                car.place = "storage2";
                            }

                        }
                    }
                } else {
                    waiting = true;
                    car.waitMembers += 1;
                    if (car.waitMembers >= 3) {
                        car.waitMembers = 0;
                        car.place = "storage2";
                    }
                }
            } else {
                if (!car.isEmpty()) {
                    hand = car.getProduct();
                    if (hand.weight != -1) {
//                        System.out.println("Грузчик " + index + " достал из машины предмет весом " + hand.weight);
                        storage2.giveProduct(hand);
                    }
                } else {
                    waiting = true;
                    if (car.waitMembers + 1 >= 3) {
                        car.waitMembers = 0;
                        car.place = "storage1";
                    } else {
                        car.waitMembers += 1;
                    }
                }
            }
        } else {
            if (!waitPlace.equals(car.place)) {
                waiting = false;
                waitPlace = car.place;
            }
```

Рисунок 13 - Класс Mover (2)