Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**
МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

О Т Ч Е Т

по лабораторной работе №8

Название: Лабораторная работа №8

Дисциплина: Языки программирования для работы с большими данными

Студент	ИУ6-22М		М.И. Замула
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			П.В. Степанов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2023 г.

Вариант: 1

Задание 2

Реализовать многопоточное приложение “Робот”. Надо написать робота, который умеет ходить. За движение каждой его ноги отвечает отдельный поток. Шаг выражается в выводе в консоль LEFT или RIGHT.

Выполнение

```
package lab8;

public class Lab8_1_2 {
    public static void main(String[] args) {
        // Создаем два потока для левой и правой руки
        Thread leftArmThread = new Thread(new LeftArm());
        Thread rightArmThread = new Thread(new RightArm());

        // Запускаем потоки
        leftArmThread.start();
        rightArmThread.start();
    }
}

// Класс, реализующий левую руку
class LeftArm implements Runnable {
    @Override
    public void run() {
        while (true) {
            System.out.println("LEFT HAND");
            try {
                Thread.sleep(1000); // пауза в 1 секунду
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

// Класс, реализующий правую руку
class RightArm implements Runnable {
    @Override
    public void run() {
        while (true) {
            System.out.println("RIGHT HAND");
            try {
                Thread.sleep(1000); // пауза в 1 секунду
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Результаты

```
RIGHT HAND
LEFT HAND
RIGHT HAND
LEFT HAND
RIGHT HAND
LEFT HAND
LEFT HAND
RIGHT HAND
RIGHT HAND
LEFT HAND
LEFT HAND
RIGHT HAND
LEFT HAND
RIGHT HAND
LEFT HAND
RIGHT HAND
LEFT HAND
```

Задание 2

Реализовать многопоточное приложение “Магазин”. Вся цепочка: производитель-магазин-покупатель. Пока производитель не поставит на склад продукт, покупатель не может его забрать. Реализовать приход товара от производителя в магазин случайным числом. В том случае, если товара в магазине не хватает– вывести сообщение.

Выполнение

```java
package lab8;

import java.util.Random;

public class Lab8_1_3 {

    public static void main(String[] args) {
        Store store = new Store();
        Producer producer = new Producer(store);
        Consumer consumer = new Consumer(store);
```

```
        new Thread(producer).start();
        new Thread(consumer).start();
    }

}

class Store {
    private int products = 0; // количество товаров на складе
    private final int maxProducts = 10; // максимальное количество товаров на
складе
    private final int minProducts = 1; // минимальное количество товаров,
которое может купить покупатель

    public synchronized void get() {
        while (products < minProducts) { // пока на складе меньше
минимального количества товаров, покупатель ждет
            try {
                System.out.println("Покупатель ждет товара на складе");
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        Random random = new Random();
        int newProducts = random.nextInt(maxProducts - minProducts + 1) +
minProducts; // генерация случайного числа товаров для покупки
        while (products < newProducts) { // пока на складе меньше товаров,
чем хочет купить покупатель, покупатель ждет
            System.out.println("Покупатель не может купить " + newProducts +
" товаров, так как на складе лежит " + products + " товаров");
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        products -= newProducts; // вычитаем количество купленных товаров из
общего числа товаров на складе
        System.out.println("Покупатель купил " + newProducts + " товаров");
        System.out.println("Товаров на складе: " + products);
        notifyAll(); // сообщаем всем потокам, что товары были куплены
    }

    public synchronized void put() {
        Random random = new Random();
        int newProducts = random.nextInt(maxProducts - minProducts + 1) +
minProducts; // генерация случайного числа новых товаров

        while (products + newProducts > maxProducts) { // пока общее
количество товаров на складе превышает максимально возможное число товаров,
производитель ждет
            try {
                System.out.println("Производитель ждет освобождения места на
складе");
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        products += newProducts; // добавляем новые товары на склад
        System.out.println("Производитель поставил " + newProducts + "
товаров");
```

```
        System.out.println("Товаров на складе: " + products);
        notifyAll(); // сообщаем всем потокам, что поставка новых товаров
завершена
    }
}

class Producer implements Runnable {
    private final Store store;

    public Producer(Store store) {
        this.store = store;
    }

    @Override
    public void run() {
        while (true) {
            store.put();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class Consumer implements Runnable {
    private final Store store;

    public Consumer(Store store) {
        this.store = store;
    }

    @Override
    public void run() {
        while (true) {
            store.get();
            try {
                Thread.sleep(1500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Результат

```
Производитель поставил 10 товаров
Товаров на складе: 10
Покупатель купил 6 товаров
Товаров на складе: 4
Производитель поставил 1 товаров
Товаров на складе: 5
Покупатель купил 2 товаров
Товаров на складе: 3
Производитель поставил 3 товаров
Товаров на складе: 6
Покупатель не может купить 10 товаров, так как на складе лежит 6 товаров
Производитель ждет освобождения места на складе
```