|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/12 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений.**

**Отчет**

**по лабораторной работе № 8**

**Вариант № 13**

**Название:** потоки

**Дисциплина:** языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-23М |  |  | С.П.Пантелеев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Цель:** изучить потоки в java

**Задание 1:** 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.

Код класса Main:

package Zadanie\_1;  
  
public class Main {  
 static AffableThread *mSecondThread*;  
 static class AffableThread extends Thread  
 {  
 @Override  
 public void run() //Этот метод будет выполнен в побочном потоке  
 {  
 for (int i = 0; i < 5; i++) {  
 try{  
 *sleep*(1000); //Приостанавливает поток на 1 секунду  
 }catch(InterruptedException e){}  
 System.*out*.println("Left");  
 }  
 }  
 }  
 public static void main(String []args){  
 System.*out*.println("Сделать 10 шагов");  
  
 *mSecondThread* = new AffableThread(); //Создание потока  
 *mSecondThread*.start(); //Запуск потока  
  
 for (int i = 0; i < 5; i++) {  
 try{  
 Thread.*sleep*(1000); //Приостанавливает поток на 1 секунду  
 }catch(InterruptedException e){}  
 System.*out*.println("Right");  
 }  
  
 }  
}

Работа программы показана на рисунке 1.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Работа программы

**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.

Код класса Main:

package Zadanie\_2;  
  
public class Main {  
 static Proizv *FirstThread*;  
 static Magaz *SecondThread*;  
 static Pokup *ThreadThread*;  
 static boolean *cheach\_dostavka*;  
 static boolean *cheach\_nalichie*;  
 static class Proizv extends Thread  
 {  
 @Override  
 public void run() //Этот метод будет выполнен в побочном потоке  
 {  
 double x = 0;  
 for (int i = 0; i < 5; i++) {  
 try{  
 *sleep*(2000);//Приостанавливает поток на 1 секунду  
 x = Math.*random*();  
 }catch(InterruptedException e){}  
 if (x > 0.5){  
 System.*out*.println("отправить товар");  
 *cheach\_dostavka* = true;  
 }  
 }  
 }  
 }  
 static class Magaz extends Thread  
 {  
 @Override  
 public void run() //Этот метод будет выполнен в побочном потоке  
 {  
 for (int i = 0; i < 5; i++) {  
 try{  
 *sleep*(1000); //Приостанавливает поток на 1 секунду  
 }catch(InterruptedException e){}  
 if (*cheach\_dostavka*){  
 System.*out*.println("товар доставлен, можно забрать");  
 *cheach\_nalichie* = true;  
 } else {  
 System.*out*.println("Товар ожидается");  
 }  
 }  
 }  
 }  
 static class Pokup extends Thread  
 {  
 @Override  
 public void run() //Этот метод будет выполнен в побочном потоке  
 {  
 for (int i = 0; i < 5; i++) {  
 try{  
 *sleep*(1000); //Приостанавливает поток на 1 секунду  
 }catch(InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 if (*cheach\_nalichie*){  
 System.*out*.println("товар забрали");  
 *cheach\_nalichie* = false;  
 }  
 }  
 }  
 }  
 public static void main(String []args){  
 *FirstThread* = new Proizv();  
 *FirstThread*.start();  
 *SecondThread* = new Magaz(); //Создание потока  
 *SecondThread*.start();  
 *ThreadThread* = new Pokup();  
 *ThreadThread*.start();//Запуск потока  
 }  
}

Работа программы показана на рисунке 2.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Работа программы

**Вывод:** во время выполнения лабораторной работы были изучены потоки в java.