|  |  |
| --- | --- |
| 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 **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 8 |

**Название:** Потоки

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | Н.Б. Гендина | |
|  | (Группа) |  | (Подпись, дата) | | (И.О. Фамилия) |
|  |  |  |  | |  |
| Преподаватель |  |  |  | | П.В. Степанов |
|  |  |  | (Подпись, дата) | | (И.О. Фамилия) |

Москва, 2024

**Цель работы:** освоить базовые принципы работы с потоками на языке Java.

**Задание 1 вариант 1**

Реализовать многопоточное приложение “Банк”. Имеется банковский счет. Сделать синхронным пополнение и снятие денежных средств на счет/со счет случайной суммой. При каждой операции (пополнения или снятие) вывести текущий баланс счета. В том случае, если денежных средств недостаточно – вывести сообщение.

|  |
| --- |
| import java.util.Random;  public class Main {  public static void main(String[] args) throws InterruptedException {  Bank bank = new Bank();  int[] accountIDs = {1, 2, 3, 4};  bank.addAccount(accountIDs[0]);  bank.addAccount(accountIDs[1]);  bank.addAccount(accountIDs[2]);  bank.addAccount(accountIDs[3]);  int rnd;  for (int i = 0; i < 10; i++) {  try {  rnd = new Random().nextInt(accountIDs.length - 1);  bank.cashIn(accountIDs[rnd], new Random().nextInt(1000));  rnd = new Random().nextInt(accountIDs.length - 1);  bank.cashOut(accountIDs[rnd], new Random().nextInt(1000));  } catch (RuntimeException e) {  System.out.println("Error: " + e);  }  }  }  } |

|  |
| --- |
| import java.util.Collections;  import java.util.HashMap;  import java.util.Map;  public class Bank {  Map<Integer, Integer> map = Collections.synchronizedMap(new HashMap<Integer, Integer>());  public void cashIn(int accountID, int amount) {  new Thread(() -> {  if (!map.containsKey(accountID)) {  throw new RuntimeException("Данного аккаунта не существует");  }  map.put(accountID, map.get(accountID) + amount);  System.out.println("Сумма пополнилась, остаток: " + map.get(accountID));  }).start();  }  public void cashOut(int accountID, int amount) {  new Thread(() -> {  if (!map.containsKey(accountID)) {  throw new RuntimeException("Данного аккаунта не существует");  }  if (Math.abs(map.get(accountID)) < amount) {  throw new RuntimeException("Сумма списания больше остатка");  }  map.put(accountID, map.get(accountID) - amount);  System.out.println("Сумма списалась, остаток: " + map.get(accountID));  }).start();  }  public void addAccount(int accountID){  if (map.containsKey(accountID)) {  throw new RuntimeException("Аккаунт уже существует");  }  map.put(accountID, 0);  }  } |

**Задание 1 вариант 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.

|  |
| --- |
| import java.io.\*;  public class Main {  public static void main(String[] args) throws IOException, InterruptedException {  System.out.println("Робот идет гулять:");  Robot.go();  }  } |

|  |
| --- |
| import java.util.concurrent.atomic.AtomicBoolean;  public class Robot {  static AtomicBoolean leg = new AtomicBoolean(false);  public static void go(){  leftLeg();  rightLeg();  }  private static void leftLeg() {  new Thread(() -> {  leg("LEFT", true);  }).start();  }  private static void rightLeg() {  new Thread(() -> {  leg("RIGHT", false);  }).start();  }  private static void leg(String side, boolean key) {  while(true) {  int timeout = (int) ((Math.random() \* (3000 - 1000)) + 1000);  try {  Thread.sleep(timeout);  } catch (InterruptedException e) {  throw new RuntimeException(e);  }  if (Robot.leg.get() == key) {  System.out.println(side);  Robot.leg.set(!Robot.leg.get());  }  }  }  } |

**Вывод:** в ходе лабораторной работы были освоены базовые принципы работы с потоками на языке Java.