|  |  |
| --- | --- |
| 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**

**Отчет**

**по лабораторной работе № 8**

**Название:** Многопоточность

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-21М |  |  | Д.Ю. Ермошин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**Задание**

Изучить принцип работы многопоточности в Java, ознакомиться с JMM.

**Выполнение работы**

Задана коллекция строк. Получить массив уникальных значений.

Bank.java

|  |
| --- |
| package lab8;  public class Bank {  private final Object object = new Object();  private long amount = 0L;  public void putMoneys(final long money) {  synchronized (object) {  amount += money;  System.out.println("Current amount: " + amount);  }  }  public void getMoneys(final long money) {  synchronized (object) {  if (amount >= money) {  amount -= money;  System.out.println("Current amount: " + amount);  } else {  System.out.println("Not enough money on amount");  }  }  }  } |

Lab8.java

|  |
| --- |
| package lab8;  import java.util.concurrent.ExecutorService;  import java.util.concurrent.Executors;  import java.util.concurrent.TimeUnit;  /\*\*  \* Реализовать многопоточное приложение “Банк”.  \* Имеется банковский счет.  \* Сделать синхронным пополнение и снятие денежных средств на счет/со счета случайной суммой.  \* При каждой операции (пополнения или снятие) вывести текущий баланс счета.  \* В том случае, если денежных средств недостаточно – вывести сообщение.  \*/  public class Lab8 {  private final ExecutorService executorService = Executors.newFixedThreadPool(2);  private final Bank bank = new Bank();  private void addMoneys() {  executorService.submit(() -> {  for (int i = 0; i < 100; i++) {  bank.putMoneys(70);  }  });  }  private void subtructMoneys() {  executorService.submit(() -> {  for (int i = 0; i < 200; i++) {  bank.getMoneys(60);  }  });  }  private void stop() throws InterruptedException {  executorService.shutdown();  final boolean done = executorService.awaitTermination(1, TimeUnit.MINUTES);  System.out.println("All tasks are done? - " + done);  }  public static void main(String[] args) throws InterruptedException {  final Lab8 lab8 = new Lab8();  lab8.subtructMoneys();  lab8.addMoneys();  lab8.stop();  }  } |

**Заключение**

В данной лабораторной работе мы изучили принцип работы с многопоточностью в языке Java, а также ознакомились с JMM.