|  |  |
| --- | --- |
| 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 № 7**

**Вариант № 3**

**Название:** Потоки

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

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

Москва, 2023

**Цель:** изучить и освоить принципы работы с потоками (Thread) в Java.

**Задание 1:** 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 Store:

import java.util.concurrent.BlockingQueue;  
import java.util.concurrent.LinkedBlockingQueue;  
  
public class Store {  
 private final BlockingQueue<Product> products = new LinkedBlockingQueue<>();  
  
 public void addProduct(Product product) {  
 try {  
 products.put(product);  
 System.out.println("Производитель поставил в магазин: " + product.getName());  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 }  
 }  
  
 public Product buyProduct() {  
 Product product = null;  
 try {  
 product = products.take();  
 System.out.println("Покупатель купил: " + product.getName());  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 }  
 return product;  
 }  
  
 public boolean hasProducts() {  
 return !products.isEmpty();  
 }  
}

Код класса Product:

public class Product {  
 private final String name;  
  
 public Product(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
}

Код класса Producer:

import java.util.Random;  
  
public class Producer implements Runnable {  
 private final Store store;  
  
 public Producer(Store store) {  
 this.store = store;  
 }  
  
 @Override  
 public void run() {  
 Random random = new Random();  
 while (true) {  
 try {  
 Thread.sleep(random.nextInt(3000)); // Случайная задержка до 3 секунд  
 Product product = new Product("Product " + random.nextInt(1000));  
 store.addProduct(product);  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 break;  
 }  
 }  
 }  
}

Код класса Consumer:

import java.util.Random;  
  
public class Consumer implements Runnable {  
 private final Store store;  
  
 public Consumer(Store store) {  
 this.store = store;  
 }  
  
 @Override  
 public void run() {  
 Random random = new Random();  
 while (true) {  
 try {  
 Thread.sleep(random.nextInt(5000)); // Случайная задержка до 5 секунд  
 if (store.hasProducts()) {  
 store.buyProduct();  
 } else {  
 System.out.println("В магазине нет товара!");  
 }  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 break;  
 }  
 }  
 }  
}

Код класса Main:

public class Main {  
 public static void main(String[] args) {  
 Store store = new Store();  
  
 Thread producerThread = new Thread(new Producer(store));  
 Thread consumerThread = new Thread(new Consumer(store));  
  
 producerThread.start();  
 consumerThread.start();  
 }  
}

Результат работы программы представлен на рисунке 1.

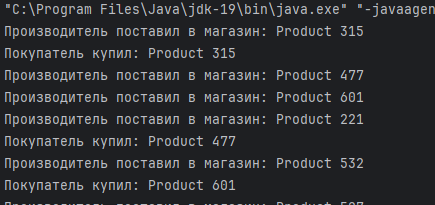


Рисунок 1 – Результат работы программы

**Задание 2:** 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.

Код класса BankAccount:

public class BankAccount {  
 private int balance;  
  
 public BankAccount(int initialBalance) {  
 this.balance = initialBalance;  
 }  
  
 public synchronized void deposit(int amount) {  
 balance += amount;  
 System.out.println("Пополнение на " + amount + ". Текущий баланс: " + balance);  
 }  
  
 public synchronized void withdraw(int amount) {  
 if (balance >= amount) {  
 balance -= amount;  
 System.out.println("Снятие " + amount + ". Текущий баланс: " + balance);  
 } else {  
 System.out.println("Недостаточно денег на счете для снятия " + amount + ". Текущий баланс: " + balance);  
 }  
 }  
}

Код класса Depositor:

import java.util.Random;  
  
public class Depositor implements Runnable {  
 private final BankAccount account;  
  
 public Depositor(BankAccount account) {  
 this.account = account;  
 }  
  
 @Override  
 public void run() {  
 Random random = new Random();  
 while (true) {  
 try {  
 Thread.sleep(random.nextInt(3000)); // Случайная задержка до 3 секунд  
 int amount = random.nextInt(1000); // Случайная сумма до 1000  
 account.deposit(amount);  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 break;  
 }  
 }  
 }  
}

Код класса Withdrawer:

import java.util.Random;  
  
public class Withdrawer implements Runnable {  
 private final BankAccount account;  
  
 public Withdrawer(BankAccount account) {  
 this.account = account;  
 }  
  
 @Override  
 public void run() {  
 Random random = new Random();  
 while (true) {  
 try {  
 Thread.sleep(random.nextInt(5000)); // Случайная задержка до 5 секунд  
 int amount = random.nextInt(10000); // Случайная сумма до 1000  
 account.withdraw(amount);  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 break;  
 }  
 }  
 }  
}

Код класса Main:

public class Main {  
 public static void main(String[] args) {  
 BankAccount account = new BankAccount(1000);  
  
 Thread depositorThread = new Thread(new Depositor(account));  
 Thread withdrawerThread = new Thread(new Withdrawer(account));  
  
 depositorThread.start();  
 withdrawerThread.start();  
 }  
}

Результат работы программы показан на рисунке 2.

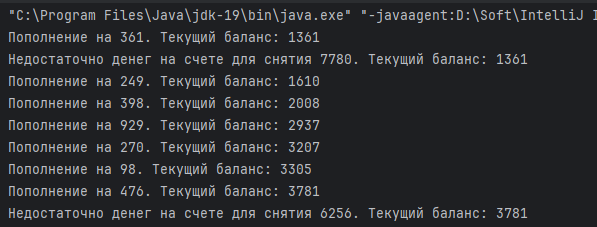


Рисунок 2 – Результат работы программы

**Вывод:** были освоены принципы работы с потоками (Thread) в Java.