|  |  |
| --- | --- |
| 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 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений.**

**Отчет**

**по лабораторной работе № 7**

**Вариант № 12**

**Название:** Потоки

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

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

Москва, 2024

**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 программы:

import java.util.concurrent.locks.ReentrantLock

import kotlin.concurrent.thread

import kotlin.random.Random

class BankAccount(var balance: Int) {

private val lock = ReentrantLock()

fun deposit(amount: Int) {

lock.lock()

try {

balance += amount

println("Пополнение: $amount, Текущий баланс: $balance")

} finally {

lock.unlock()

}

}

fun withdraw(amount: Int) {

lock.lock()

try {

if (balance >= amount) {

balance -= amount

println("Снятие: $amount, Текущий баланс: $balance")

} else {

println("Недостаточно средств для снятия: $amount, Текущий баланс: $balance")

}

} finally {

lock.unlock()

}

}

}

fun main() {

val bankAccount = BankAccount(1000)

val threads = List(10) {

thread {

repeat(10) {

val amount = Random.nextInt(1, 500)

if (Random.nextBoolean()) {

bankAccount.deposit(amount)

} else {

bankAccount.withdraw(amount)

}

Thread.sleep(100)

}

}

}

threads.forEach { it.join() }

println("Финальный баланс: ${bankAccount.balance}")

}

Результат работы программы представлен на рисунке 1.

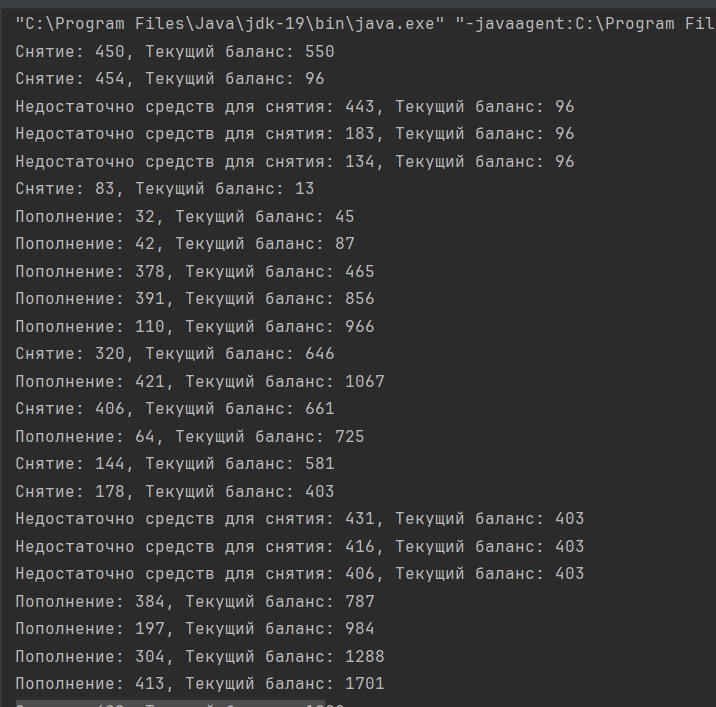


Рисунок 1 – Результат работы программы

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