|  |  |
| --- | --- |
| 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 № 7**

**Вариант № 6**

**Название:** Потоки

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | Д.А. Залимханов |
|  | (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 (Kotlin).

**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:

class Store {

private var stock = 0

private val mutex = Mutex()

suspend fun addProduct(quantity: Int) {

mutex.withLock {

stock += quantity

println("Производитель добавил $quantity продукт(ов). На складе: $stock")

}

}

suspend fun buyProduct(quantity: Int) {

mutex.withLock {

if (stock < quantity) {

println("Недостаточно товара для покупателя. На складе: $stock")

}

else{

stock -= quantity

println("Покупатель купил $quantity продукт(ов). Остаток на складе: $stock")

}

}

}

}

**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:

class BankAccount {

private var balance = 0

private val mutex = Mutex()

suspend fun deposit(amount: Int) {

mutex.withLock {

balance += amount

println("Пополнение счета на $amount. Текущий баланс: $balance")

}

}

suspend fun withdraw(amount: Int) {

mutex.withLock {

if (amount <= balance) {

balance -= amount

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

} else {

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

}

}

}

}

Обе программы тестировались в одной функции Main.

Код функции Main:

fun main() = runBlocking {

val store = Store()

val producer = launch(Dispatchers.Default) {

while (isActive) {

val quantity = Random.nextInt(1, 10)

store.addProduct(quantity)

delay(1000)

}

}

val consumer = launch(Dispatchers.Default) {

while (isActive) {

val quantity = Random.nextInt(1, 10)

store.buyProduct(quantity)

delay(1500)

}

}

delay(10000)

producer.cancel()

consumer.cancel()

val account = BankAccount()

val jobDeposit = launch(Dispatchers.Default) {

repeat(10) {

account.deposit(Random.nextInt(100, 500))

delay(200)

}

}

val jobWithdraw = launch(Dispatchers.Default) {

repeat(10) {

account.withdraw(Random.nextInt(100, 500))

delay(300)

}

}

delay(10000)

jobDeposit.cancel()

jobWithdraw.cancel()

}

Результат работы программы показан на рисунке 2.

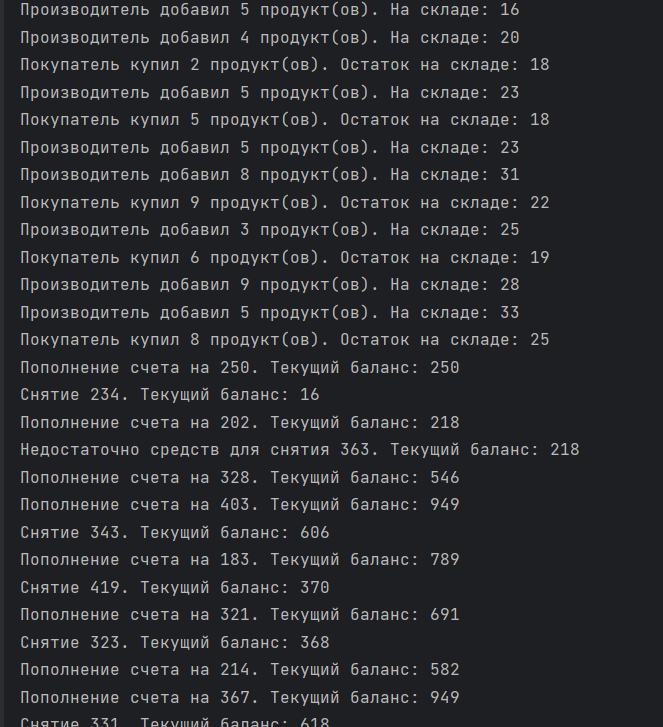


Рисунок 1 – Результат работы программ

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