**Лабораторная работа №7: Потоки Executable**

**Вавакин Владислав**

**Постановка задачи**

Пункт техобслуживания транспортных средств принимает машины в 6 очередей. Сначала у владельцев принимают документы, затем все машины попадают на общую стоянку, где проводится их осмотр. На общую стоянку не может попасть более 10 машин одновременно. После осмотра владельцы покидают стоянку осмотра по одному и одновременно может заехать следующий.

**Ход работы**

Листинг программы:

import java.util.concurrent.Semaphore;

public class VehicleServiceStation {

private static final int QUEUES\_COUNT = 6;

private static final int PARKING\_CAPACITY = 10;

private static final int TOTAL\_CARS = 30;

public static void main(String[] args) throws InterruptedException {

Semaphore parkingLot = new Semaphore(PARKING\_CAPACITY, true);

Thread[] threads = new Thread[TOTAL\_CARS];

System.out.println("Пункт техобслуживания открыт. Очередей: " + QUEUES\_COUNT);

for (int i = 0; i < TOTAL\_CARS; i++) {

int carNumber = i + 1;

int queueNumber = i % QUEUES\_COUNT + 1;

threads[i] = new Thread(() -> {

try {

System.out.printf("Машина %d в очереди %d: прием документов...\n",

carNumber, queueNumber);

Thread.sleep(100 + (int)(Math.random() \* 400));

System.out.printf("Машина %d в очереди %d: ожидание места на стоянке...\n",

carNumber, queueNumber);

parkingLot.acquire();

System.out.printf("Машина %d в очереди %d: заехала на стоянку. Свободных мест: %d\n",

carNumber, queueNumber, parkingLot.availablePermits());

Thread.sleep(1000 + (int)(Math.random() \* 1000));

System.out.printf("Машина %d в очереди %d: осмотр завершен, покидает стоянку\n",

carNumber, queueNumber);

parkingLot.release();

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

});

threads[i].start();

}

for (Thread thread : threads) {

thread.join();

}

System.out.println("Все машины обслужены. Пункт техобслуживания закрывается.");

}

}

Задачу можно решить в рамках метода main() главного класса. В данном случае таковым будет класс VehicleServiceStation.

В качестве полей данного класса выступают статические поля QUEUES\_COUNT – количество очередей (6), PARKING\_CAPACITY – вместимость общей парковки (10) и TOTAL\_CARS – общее число машин-потоков (30).

Алгоритм работы программы начинается с инициализации семафора parkingLot, который контролирует доступ к парковке и устанавливает максимальное количество одновременно разрешенных машин равным PARKING\_CAPACITY (10), с честным режимом распределения разрешений (fair=true), что гарантирует обслуживание потоков в порядке очереди.

Далее создается массив потоков threads размером TOTAL\_CARS (30), каждый из которых будет имитировать поведение машины в пункте техобслуживания. Программа выводит сообщение об открытии пункта и количестве очередей (6). Затем в цикле создаются и запускаются потоки для каждой машины, где каждой машине присваивается уникальный номер (carNumber) и номер очереди (queueNumber), вычисляемый как остаток от деления номера машины на количество очередей плюс один для корректной нумерации.

Каждый поток сначала имитирует прием документов, выводя соответствующее сообщение и задерживаясь на случайное время от 100 до 500 миллисекунд, после чего пытается получить разрешение от семафора для въезда на стоянку, выводя сообщение об ожидании места.

После успешного получения разрешения поток выводит информацию о заезде на стоянку и количестве оставшихся свободных мест, затем имитирует процесс осмотра, задерживаясь на случайное время от 1000 до 2000 миллисекунд, после чего освобождает разрешение семафора, выводя сообщение о завершении осмотра и покидании стоянки.

В случае прерывания потока обрабатывается исключение InterruptedException с восстановлением прерванного статуса. После запуска всех потоков основной поток программы ожидает их завершения с помощью метода join(), и после завершения работы всех потоков выводится сообщение о закрытии пункта техобслуживания.

Таким образом, программа моделирует работу пункта техобслуживания с несколькими очередями, ограниченной парковкой и случайными временными задержками на каждом этапе обслуживания машин, обеспечивая корректное управление параллельными потоками с использованием семафора для контроля доступа к общему ресурсу.

Последовательность выполнения первого потока можно представить в виде диграммы:

