**Пример использования Phaser**

В примере PhaserExample создается несколько потоков, играющих роль пассажиров. Phaser играет роль метро, которое должно проследовать вдоль нескольких станций. Каждая станция (фаза) имеет свой номер. Класс Passenger играет роль пассажира, который на одной из станции должен зайти в вагон, а на другой выйти. Количество пассажиров, а также их места посадки и высадки, формируются случайным образом.

**Листинг класса Passenger**

Конструктор класса Passenger получает значение идентификатора, номера станций посадки и назначения (высадки). При создании объекта в консоль выводится информация о пассажире (метод toString).

Как только Phaser переходит в определенную фазу, номер которой соответствует станции посадки пассажира, то поток данного Passenger стартует (run) и выводит в консоль сообщение, что пассажир вошел в вагон, т.е. находится в ожидании следующей станции/фазы (arriveAndAwaitAdvance). Если следующая станция/фаза не будет соответствовать станции назначения, то Passenger продолжит свой путь. Как только Phaser перейдет в фазу, номер которой соответствует номеру станции назначания пассажира, то цикл контроля завершится и поток продолжит работу. С задержкой в 500 ms он сообщит, что покинул вагон и отменит регистрацию в Phaser (arriveAndDeregister).

Таким образом, поток/пассажир дожидается свой фазы/станции в цикле, выделенной в коде пунктирными комментариями. Вызов метода *arriveAndAwaitAdvance* возвращает значение следующего номера фазы, т.е. участник будет вызван при переходе Phaser в новое состояние. Если в этом состоянии значение фазы (getPhase) будет соответствовать номеру destination, то цикл прервется, в противном случае, ожидание следующей фазы и повторное выполнение проверки условия while.

private static String WAIT = " ждёт на станции ";

private static String ENTER = " вошел в вагон" ;

private static String EXIT = " вышел из вагона ";

private static String SPACE = " ";

public static class Passenger implements Runnable

{

int id;

int departure;

int destination;

public Passenger(int id, int departure, int destination)

{

this.id = id;

this.departure = departure;

this.destination = destination;

System.out.println(this + WAIT + departure);

}

@Override

public void run() {

try {

System.out.println(SPACE + this + ENTER);

//-----------------------------------------------

// Заявляем об участии и ждем станции назначения

while (PHASER.getPhase() < destination)

PHASER.arriveAndAwaitAdvance();

//----------------------------------------------

Thread.sleep(500);

System.out.println(SPACE + this + EXIT);

// Отмена регистрации

PHASER.arriveAndDeregister();

} catch (InterruptedException e) {}

}

@Override

public String toString()

{

return "Пассажир " + id +

" {" + departure + " -> " + destination + '}';

}

}

***Примечание :****Passenger является внутренним классом примера/класса PhaserExample, и для описания вынесен из общего кода, чтобы не загромождать листинг.*

**Листинг примера PhaserExample**

В примере сначала создается объект синхронизации PHASER. После этого формируется массив пассажиров. При создании объекта Passenger случайным образом определяются станция посадки и станция назначения. После того, как массив пассажиров подготовлен, PHASER в цикле начинает менять свое состояние. На каждом шаге выполняется проверка «станции посадки пассажира». Если она соответствует значению фазы, то данный пассажир входит в вагон метро, т.е. регистрируется в PHASER и поток стартует. Таким образом, регистрация участников (исполнительных потоков) выполняется при нахождении PHASER в определенном состоянии/фазе. Пассажир покинет вагон при достижении метро станции назначения, т.е. при нахождении PHASER в соответствующей фазе. Но это произойдет уже в коде класса Passenger, рассмотренного выше.

import java.util.ArrayList;

import java.util.concurrent.Phaser;

public class PhaserExample

{

private static Phaser PHASER;

private static String OPEN = " открытие дверей ";

private static String CLOSE = " закрытие дверей ";

public static void main(String[] args)

throws InterruptedException

{

// Регистрация объекта синхронизации

PHASER = new Phaser(1);

ArrayList<Passenger> passengers;

passengers = new ArrayList<>();

// Формирование массива пассажиров

for (int i = 1; i < 5; i++) {

if ((int) (Math.random() \* 2) > 0)

// Этот пассажир проезжает одну станцию

passengers.add(new Passenger(10 + i, i,

i + 1));

if ((int) (Math.random() \* 2) > 0) {

// Этот пассажир едет до конечной

Passenger p = new Passenger(20 + i, i, 5);

passengers.add(p);

}

}

// Фазы 0 и 6 - конечные станции

// Фазы 1...5 - промежуточные станции

for (int i = 0; i < 7; i++) {

switch (i) {

case 0:

System.out.println(

"Метро вышло из тупика");

// Нулевая фаза, участников нет

PHASER.arrive();

break;

case 6:

// Завершаем синхронизацию

System.out.println(

"Метро ушло в тупик");

PHASER.arriveAndDeregister();

break;

default:

int currentStation = PHASER.getPhase();

System.out.println(

"Станция " + currentStation);

// Проверка наличия пассажиров

// на станции

for (Passenger pass : passengers)

if (pass.departure==currentStation){

// Регистрация участника

PHASER.register();

new Thread (pass).start();

}

System.out.println(OPEN);

// Phaser ожидает завершения фазы

// всеми участниками

PHASER.arriveAndAwaitAdvance();

System.out.println(CLOSE);

}

}

}

**Результаты выполнения программы**

Пассажир 11 {1 -> 2} ждёт на станции 1

Пассажир 21 {1 -> 5} ждёт на станции 1

Пассажир 12 {2 -> 3} ждёт на станции 2

Пассажир 23 {3 -> 5} ждёт на станции 3

Пассажир 14 {4 -> 5} ждёт на станции 4

Метро вышло из тупика

Станция 1

... открытие дверей ...

Пассажир 11 {1 -> 2} вошел в вагон

Пассажир 21 {1 -> 5} вошел в вагон

... закрытие дверей ...

Станция 2

... открытие дверей ...

Пассажир 12 {2 -> 3} вошел в вагон

Пассажир 11 {1 -> 2} вышел из вагона

... закрытие дверей ...

Станция 3

... открытие дверей ...

Пассажир 23 {3 -> 5} вошел в вагон

Пассажир 12 {2 -> 3} вышел из вагона

... закрытие дверей ...

Станция 4

... открытие дверей ...

Пассажир 14 {4 -> 5} вошел в вагон

... закрытие дверей ...

Станция 5

... открытие дверей ...

Пассажир 21 {1 -> 5} вышел из вагона

Пассажир 23 {3 -> 5} вышел из вагона

Пассажир 14 {4 -> 5} вышел из вагона

... закрытие дверей ...

Метро ушло в тупик