Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Лабораторна робота № 1

Тема: “Потоки”

Виконав:  
ст. гр. ІПм-12-2  
Лубковський А. А.  
Перевірив:  
Лютак І.З.

Івано-Франківськ

2016

Мета: Освоїти на практиці роботу із потоками на мові Java.

Завдання

Розробити програму на мові Джава. Програма має обчислювати покази температури від 100 давачів. Давач має бути класом, який використовується в окремому потоці (Thread). У цьому класі має бути довільно згенероване число від -10 до +40, що позначатиме температуру. Кожен клас повинен генерувати випадкове число від 1 до 10, що має бути використано як затримка потоку на провіднику, зчитування даних із давача (давач умовний). Основна програма має зчитати всі давачі по 2-а рази і записати дані у файл. Формат файлу повинен бути сумісним із форматами імпорту в електронну таблицю.

Короткі теоретичні відомості

Самий простий спосіб створення потоку – це оголошення класу, що реалізує інтерфейс Runnable. Інтерфейс Runnable абстрагує одиницю коду, що виконується. Ви можете створити потік з любого об’єкту, що реалізує інтерфейс Runnable. Щоб реалізувати інтерфейс Runnable, клас повинен оголосити метод run().

Всередині методу run () визначається код, який і складає новий потік. Важливо пам’ятати, що метод run () може викликати інші методи, використовувати інші класи, оголошувати змінні – так же само, як це робить головний потік. Єдиною різницею є те, що метод run () встановлює точку входу для іншого, паралельного потоку всередині програми. Цей потік завершиться, коли метод run () поверне керування.

Деякі методи, що використовуються у класі Thread: MAX\_PRIORITY, dumpStack, sleep, interrupted.

Код

Клас ThreadGeneration

**public** **class** ThreadGeneration **implements** Runnable {

**private** String threadName;

**private** **int** generationIndex;

ThreadGeneration (**int** generationIndex) {

**this**.generationIndex = generationIndex;

threadName = String.*valueOf*(generationIndex);

Thread threadInstance = **new** Thread(**this**, threadName);

**try** {

threadInstance.start();

threadInstance.join();

} **catch** (InterruptedException e) {

System.***out***.println("Потоки проби " + generationIndex + " перервано.");

}

}

**public** **void** run() {

**for** (**int** index = 1; index < 101; index++) {

**try** {

ThreadGenerator threadGeneratorInstance = **new** ThreadGenerator(index, generationIndex);

} **catch** (Exception e) {

System.***out***.println("Помилка із сенсором " + (index));

}}}}

Клас ThreadGenerator

**public** **class** ThreadGenerator **implements** Runnable {

**private** String threadName;

**private** **int** sensorIndex;

**private** **int** generationIndex;

**private** **int** temperature;

ThreadGenerator (**int** sensorIndex, **int** generationIndex) {

**this**.sensorIndex = sensorIndex;

**this**.generationIndex = generationIndex;

threadName = String.*valueOf*(sensorIndex);

Thread threadInstance = **new** Thread(**this**, threadName);

System.***out***.println("Новий сенсор: " + threadInstance);

threadInstance.start();

}

**public** **void** run() {

**try** {

Sensor sensorInstance = **new** Sensor(); // клас, що генерується в окремому потоці

temperature = sensorInstance.getTemperature(); // температура від давача

System.***out***.println("Температура давача " + threadName + " = " + temperature);

Thread.*sleep*(sensorInstance.getDelay()); // затримка на провіднику

Writer writerInstance = **new** Writer();

System.***out***.println("Температура при записі = " + temperature);

writerInstance.write(generationIndex + "\t" + sensorIndex + "\t" + temperature + "\n");

} **catch** (InterruptedException e) {

System.***out***.println(threadName + " перерваний " + e);

} **catch** (Exception e) {

System.***out***.println(threadName + " з помилкою " + e);

}

System.***out***.println(threadName + " завершено.");

}}

Клас Sensor

**public** **class** Sensor {

**private** Random randomInstance = **new** Random();

**protected** **final** **int** getTemperature() {

**return** randomInstance.nextInt(50) - 10;

}

**protected** **final** **int** getDelay() {

**return** randomInstance.nextInt(9)+1;

}}

Висновок: Виконуючи лабораторну роботу було освоєно роботу із потоками мовою Java.