УДК 004

Использование потоков в Java

Еровлева Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье будут представлены примеры использования потоков в приложении. Работа будет происходить в среде программирования IntelijIdea с использованием языка программирования Java.
Ключевые слова: Java, Поток, Коллекция

Using Streams in Java

Erovleva Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article uses examples of using streams in applications. The work will be typical for IntelijIdea programming using the Java programming language.
Keywords: Java, Stream, Collection

Потоки и коллекции похожи друг на друга, но предназначены для разных целей. Коллекции обеспечивают эффективное управление доступом к каждому объекту, когда потоки напрямую не заинтересованы в доступе и манипулировании каждым имеющимся у них элементом. Потоки предназначены для обработки параллельных и последовательных операций агрегирования с использованием методологии цепочки методов.

Цель работы – продемонстрировать пример работы потоков в Java приложении.

Исследованиями в данной теме занимались следующие авторы. А.А.Шейн, Д.Г.Залевский, С.В. Автайкин, С.В.Карташев, С.А.Скороход разработали и описали действия программы, предназначенной для

автоматического создания набора классов для представления объектов модели Decode в виде нативных объектов языка Java [1]. Н.Н. Глибовец проанализировала в своей работе особенности агентных технологий и перспективы их использования для разработки сложных многопользовательских программных систем [2]. В своей работе М.К.Ермаков, С.П.Вартанов рассмотрели вопросы проведения анализа программ интерпретируемых языков программирования [3]. С.В. Мельников провел обзор на работу с отладочным интерфейсом Java и методом модификации функциональности приложения, не изменяющий его бинарные файлы [4]. Так же А.А. Птицын, Н.Л. Подколодный, Д.А.Григорович, С.В. Лаврюшев разработали молекулярно-биологический сервер и на его базе создали ряд информационно-вычислительных систем, для изучения регуляции экспрессии генов с использованием новейших технологий Java [5].

Сначала создадим класс «User». Он будет представлять базовый объект для образцов каждого пользователя (рис.1).

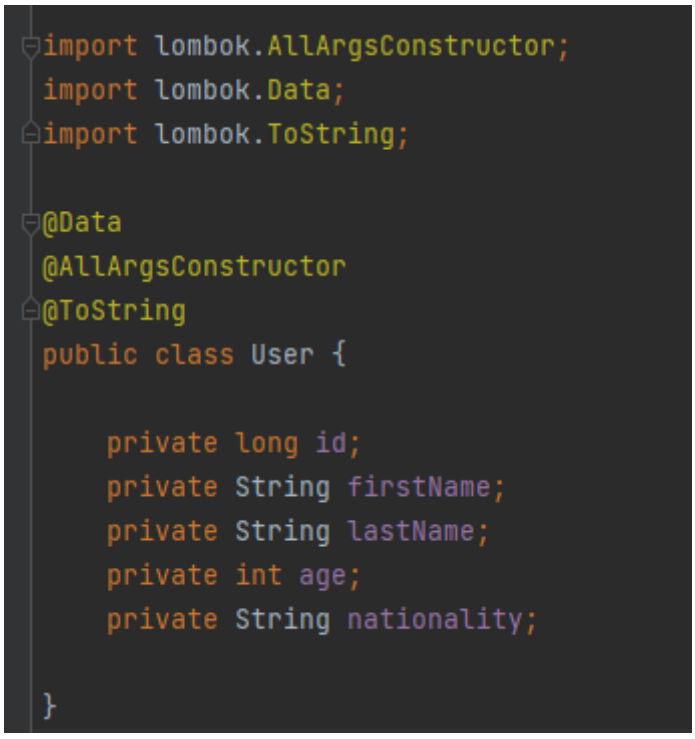

Рисунок 1 – Класс User

Теперь напишем «Sample» класс, в котором будут запускаются все образцы (рис.2).

```
import java.util.*;
import java.util.stream.Collectors;

public class Sample {

    private final List<User> userList = Arrays.asList(
            new User( id: 1,  firstName: "Иван",  lastName: "Ерофимов",  age: 37,  nationality: "Франция"),
            new User( id: 2,  firstName: "Петр",  lastName: "Ильник",  age: 11,  nationality: "США"),
            new User( id: 3,  firstName: "Федор",  lastName: "Степной",  age: 7,  nationality: "Франция"),
            new User( id: 4,  firstName: "Станислав",  lastName: "Греков",  age: 77,  nationality: "Китай"),
            new User( id: 5,  firstName: "Григорий",  lastName: "Шпутов",  age: 23,  nationality: "Великобритания"),
            new User( id: 6,  firstName: "Алина",  lastName: "Анинская",  age: 11,  nationality: "Китай"),
            new User( id: 7,  firstName: "Екатерина",  lastName: "Герова",  age: 37,  nationality: "Канада")
    );
```

Рисунок 2 – Класс Sample

Теперь создадим примеры использования потоков в приложении. Сначала попробуем вывести все записи (рис.3).

```
private void test1() {
    System.out.println("Test 1");
    userList.forEach(System.out::println);
}
```

Рисунок 3 – Пример для вывода всех записей

После запуска программы в консоле будет вывод (рис.4).

```
Test 1
User(id=1, firstName=Иван, lastName=Ерофимов, age=37, nationality=Франция)
User(id=2, firstName=Петр, lastName=Ильник, age=11, nationality=США)
User(id=3, firstName=Федор, lastName=Степной, age=7, nationality=Франция)
User(id=4, firstName=Станислав, lastName=Греков, age=77, nationality=Китай)
User(id=5, firstName=Григорий, lastName=Шпутов, age=23, nationality=Великобритания)
User(id=6, firstName=Алина, lastName=Анинская, age=11, nationality=Китай)
User(id=7, firstName=Екатерина, lastName=Герова, age=37, nationality=Канада)
```

Рисунок 4 – Вывод первого примера

Из-за того, что «userList» это «ArrayList», элементы записываются в консоль в порядке создания списка.

Следующим примером сделаем обработку каждого элемента и сделаем вывод в консоль (рис. 5-6).

```
private void test2() {
    System.out.println("Test 2");

    userList.stream()
            .map(u -> new User(
                    u.getId(),
                    firstName: "X " + u.getFirstName(),
                    lastName: "Y " + u.getLastName(),
                    age: u.getAge() + 10,
                    u.getNationality()))
            .collect(Collectors.toList())
            .forEach(System.out::println);
}
```

Рисунок 5 – Пример обработки каждого элемента

```
Test 2
User(id=1, firstName=X Иван, lastName=Y Ерофимов, age=47, nationality=Франция)
User(id=2, firstName=X Петр, lastName=Y Ильник, age=21, nationality=США)
User(id=3, firstName=X Федор, lastName=Y Степной, age=17, nationality=Франция)
User(id=4, firstName=X Станислав, lastName=Y Греков, age=87, nationality=Китай)
User(id=5, firstName=X Григорий, lastName=Y Шпутов, age=33, nationality=Великобритания)
User(id=6, firstName=X Алина, lastName=Y Анинская, age=21, nationality=Китай)
User(id=7, firstName=X Екатерина, lastName=Y Герова, age=47, nationality=Канада)
```

Рисунок 6 – Вывод второго примера

Третьим примером сделаем сортировку по возрасту от меньшего к большему (рис.7-8).

```
private void test3() {
    System.out.println("Test 3");

    userList.stream()
            .sorted(Comparator.comparing(User::getAge))
            .collect(Collectors.toList())
            .forEach(System.out::println);
}
```

Рисунок 7 – Пример сортировки по возрасту

```
Test 3
User(id=3, firstName=Федор, lastName=Степной, age=7, nationality=Франция)
User(id=2, firstName=Петр, lastName=Ильник, age=11, nationality=США)
User(id=6, firstName=Алина, lastName=Анинская, age=11, nationality=Китай)
User(id=5, firstName=Григорий, lastName=Шпутов, age=23, nationality=Великобритания)
User(id=1, firstName=Иван, lastName=Ерофимов, age=37, nationality=Франция)
User(id=7, firstName=Екатерина, lastName=Герова, age=37, nationality=Канада)
User(id=4, firstName=Станислав, lastName=Греков, age=77, nationality=Китай)
```

Рисунок 8 – Вывод третьего примера

Теперь попробуем сделать сортировку по трем параметрам: возраст, имя, фамилия (рис. 9-10).

```
private void test4() {
    System.out.println("Test 4");

    userList.stream()
            .sorted(Comparator.comparing(User::getAge)
                    .thenComparing(User::getFirstName)
                    .thenComparing(User::getLastName))
            .collect(Collectors.toList())
            .forEach(System.out::println);
}
```

Рисунок 9 – Пример сортировки по нескольким параметрам

```
Test 4
User(id=3, firstName=Федор, lastName=Степной, age=7, nationality=Франция)
User(id=6, firstName=Алина, lastName=Анинская, age=11, nationality=Китай)
User(id=2, firstName=Петр, lastName=Ильник, age=11, nationality=США)
User(id=5, firstName=Григорий, lastName=Шпутов, age=23, nationality=Великобритания)
User(id=7, firstName=Екатерина, lastName=Герова, age=37, nationality=Канада)
User(id=1, firstName=Иван, lastName=Ерофимов, age=37, nationality=Франция)
User(id=4, firstName=Станислав, lastName=Греков, age=77, nationality=Китай)
```

Рисунок 10 – Вывод четвертого примера

Так же имеется возможность находить средние, максимальные и минимальные значения. Попробуем найти средний возраст записанных пользователей и найти так же максимальную длину имени (рис.11-12).

```
private void test5() {
    System.out.println("Test 5");

    double averageAge = userList.stream() Stream<User>
            .mapToInt(User::getAge) IntStream
            .summaryStatistics() IntSummaryStatistics
            .getAverage();

    System.out.print("Средний возраст: " + averageAge);

    int maxFirstNameLenght = userList.stream() Stream<User>
            .mapToInt(user -> {
                return user.getFirstName().length();
            }) IntStream
            .summaryStatistics() IntSummaryStatistics
            .getMax();

    System.out.println("\nМаксимальная длина имени: " + maxFirstNameLenght);
}
```

Рисунок 11 – Пример нахождения среднего и максимального значения

```
Test 5
Средний возраст: 29.0
Максимальная длина имени: 9
```

Рисунок 12 – Вывод пятого примера

В потоках есть возможность создавать проверки. В следующем примере проверим, все ли пользователи имеют возраст больше 6. А также проверим, начинается ли имя на букву «С» у какого-либо пользователя (рис.13-14).

```
private void test6() {
    System.out.println("Test 6");

    boolean isAllAgesGreaterThan6 = userList.stream()
            .allMatch(user -> user.getAge() > 6);

    System.out.println("Все пользователи старше 6: " + isAllAgesGreaterThan6);
}

private void test7() {
    System.out.println("Test 7");
    boolean isFirstCharS = userList.stream()
            .anyMatch(user -> user.getFirstName().charAt(0) == 'C');

    System.out.println("Имя начинается на 'C': " + isFirstCharS);
}
```

Рисунок 13 – Пример проверки значений

```
Test 6
Все пользователи старше 6: true
Test 7
Имя начинается на 'C': true
```

Рисунок 14 – Вывод шестого и седьмого примера

Так же можно посчитать сколько национальностей имеется у записанных пользователей (рис. 15-16).

```
private void test8() {
    System.out.println("Test 8");

    long countDifferentNationalites = userList.stream() Stream<User>
            .map(User::getNationality) Stream<String>
            .distinct()
            .count();

    System.out.println("Количество национальностей: " + countDifferentNationalites);
}
```

Рисунок 15 – Пример нахождения количества национальностей

```
Test 8
Количество национальностей: 5
```

Рисунок 8 – Вывод восьмого примера

Подводя итог, Java Stream — это не структура данных, в которой хранятся элементы, вместо этого он передает элементы из источника, такого как структура данных, через конвейер вычислительных операций. Операция над потоком Java создает возвращаемое значение, но не изменяет исходный источник.

Библиографический список

1. Шейн А.А., Залевский Д.Г., Автайкин С.В., Карташев С.В., Скороход С.А. Генератор исходного кода на языке java по описанию бортовых компонентов decode (decode java generator 0.2) // Вестник волжского университета им. в.н. татищева. 2019. №3. С. 26-32.
2. Глибовец Н.Н. Использование jade (java agent development environment) для разработки компьютерных систем поддержки дистанционного обучения агентного типа // Заметки по информатике и математике. 2019. №10. С. 15-20.
3. Ермаков М.К., Вартанов С.П. Подход к проведению динамического анализа java-программ методом модификации виртуальной машины java // Научные труды Винницкого национального технологического университета. 2018. №6. С. 10-17.
4. Мельников С.В. Обзор и применение отладочного интерфейса java (jdi) для обратимой модификации программных продуктов // Современные проблемы науки и образования. 2018. №8. С. 8-19.
5. Птицын А.А., Подколодный Н.Л., Григорович Д.А., Лаврюшев С.В. Создание молекулярно-биологического сервера www с использованием новейших технологий java // Заметки по информатике и математике. 2020. №1. С. 11-20.