Practices for Lesson 16: The Fork-Join Framework

Chapter 16

Practices for Lesson 16: Overview

Practices Overview

In these practices, you will use the Fork-Join Framework.

Practice 16-1: Detailed Level: Using the Fork-Join Framework

Overview

In this practice, you will modify an existing project to use the Fork-Join framework.

Assumptions

You have reviewed the sections covering the use of the Fork-Join framework.

Summary

You are given an existing project that already leverages the Fork-Join framework to process the data contained within an array. Before the array is processed, it is initialized with random numbers. Currently the initialization is single-thread. You must use the Fork-Join framework to initialize the array with random numbers.

Tasks

1. Open the `ForkJoinFindMax16-01Prac` project as the main project.
   - Select File > Open Project.
   - Browse to `/home/oracle/labs/16-ForkJoin/practices/practice1`.
   - Select `ForkJoinFindMax16-01Prac` and click the Open Project button.
2. Expand the project directories.
3. Open the `Main` class in the `com.example` package.
   - Review the code within the `main` method. Take note of how the `FindMaxTask` class is called.
4. Open the `FindMaxTask` class in the `com.example` package.
   - Review the code within the class. Take note of the `for` loop used to initialize the `data` array with random numbers.
   - Take note of how the `compute` method splits the `data` array if the count of elements to process is too great.
5. Create a `RandomArrayAction` class in the `com.example` package.
   - Add four fields.

```
private final int threshold;
private final int[] myArray;
private int start;
private int end;
```

   - Add a constructor that receives parameters and saves their values within the fields defined in the previous step.

```
public RandomArrayAction(int[] myArray, int start, int end, int
threshold)
```

   - Modify the class signature to extend the `RecursiveAction` class from the `java.util.concurrent` package.

     Note: A `RecursiveAction` is used when a `ForkJoinTask` with no return values is needed.

- Add the `compute` method. Note that unlike the `compute` method from a `RecursiveTask`, the `compute` method in a `RecursiveAction` returns `void`.

```
protected void compute() { }
```

- Begin the `compute` method. If the number of elements to process is below the threshold, you should initialize the array.

```
if (end - start < threshold) {
  for (int i = start; i <= end; i++) {
    myArray[i] = ThreadLocalRandom.current().nextInt();
  }
```

Note: `ThreadLocalRandom` is used instead of `Math.random()` because `Math.random()` does not scale when executed concurrently by multiple threads and would eliminate any benefit of applying the Fork-Join framework to this task.

- Complete the `compute` method. If the number of elements to process is above or equal to the threshold you should find the midway point in the array and create two new `RandomArrayAction` instances for each section of the array to process. Start each `RandomArrayAction`.

  Note: When starting a `RecursiveAction`, you can use the `invokeAll` method instead of the `fork`/`join`/`compute` combination typically seen with a `RecursiveTask`.

```
} else {
  int midway = (end - start) / 2 + start;
  RandomArrayAction r1 = new RandomArrayAction(myArray, start,
                         midway, threshold);
  RandomArrayAction r2 = new RandomArrayAction(myArray,
                         midway + 1, end, threshold);
  invokeAll(r1, r2);
}
```

6. Modify the `main` method of the `Main` class to use the `RandomArrayAction` class.
   - Comment out the `for` loop within the `main` method that initializes the `data` array with random values.
   - After the line that creates the `ForkJoinPool`, create a new `RandomArrayAction`.
   - Use the `ForkJoinPool` to invoke the `ForkJoinPool`.
   - Your code should look like this:

```
//         for (int i = 0; i < data.length; i++) {
//             data[i] = ThreadLocalRandom.current().nextInt();
//         }

ForkJoinPool pool = new ForkJoinPool();

RandomArrayAction action = new RandomArrayAction(data, 0,
                           data.length - 1, data.length / 16);
pool.invoke(action);
```

7. Run the `ForkJoinFindMax16-01Prac` project by right-clicking the project and choosing *Run*.