Practice 8

Deadline: 2 weeks from now. Should be checked onsite (during labs).

In this practice, we'll use executor service as a thread pool to control groups of tasks and increase the efficiency of task execution.

Task 1

This task lets users input a word, and count the **total occurrences** of this word in all files in a directory tree. Unzip the `src.zip` of JDK 8 and use it as our target directory.

To speed up task execution, we could make a separate task for each file, i.e., count the occurrence of the given word in one file. Since there are many files in a directory, we can submit all these tasks to a thread pool so that they'll execute concurrently. Steps:

1. Create a thread pool using the static factory methods of `Executors`.
2. Define a `Callable`, which represents a task that counts the occurrence of the given word in a given file.
3. Traverse the target directory to create a list of such `Callables`.
4. Use executor's `invokeAll` method to run the list of tasks asynchronously, and get back a list of `Future` objects.
5. Sum all the `Future` objects to compute the total occurrence of the given word.

You may use the following code to compute time elapsed for the tasks. Try use different executors (e.g., `CachedThreadPool`,`SingleThreadExecutor`, `FixedThreadPool`) to observe the difference of efficiency.

```
Instant startTime = Instant.now();

// Task execution ......

Instant endTime = Instant.now();

System.out.println("Time elapsed: "
        + Duration.between(startTime, endTime).toMillis() + " ms\n");
```

Sample output

```
Enter keyword (e.g. volatile): volatile
Occurrences of volatile: 441
Time elapsed: 1535 ms
```

```
Enter keyword (e.g. volatile): synchronized
Occurrences of synchronized: 3656
Time elapsed: 2268 ms
```

Task 2

In the second task, we search for the first file that contains the given word. Again, we use the directory (by unzipping `src.zip`) as our target directory. As long as we found a file (any file) that contains the given word, we consider the task to succeed.

We can use `invokeAny` to parallelize the search tasks, with the following steps:

1. Create a thread pool using the static factory methods of `Executors`.
2. Define a `Callable`, which represents a task that search for the given word in a given file, and returns the file path when search succeeds.
3. Traverse the target directory to create a list of such `Callables`.
4. Use executor's `invokeAny` method to parallel the search, and get back the path of the first file that succeeds the search.

We have to be more careful about formulating the tasks in step 2. Since the `invokeAny` method terminates as soon as any task returns, we cannot have the search tasks return a `boolean` to indicate success or failure, because we don't want to stop searching when a search task failed. Instead, a failing search task (i.e., the file doesn't contain the given word) should throws a `NoSuchElementException`.

Also, when one task has succeeded, the others are canceled. Therefore, we monitor the interrupted status. If the underlying thread is interrupted, the search task prints a message before terminating, so that you can see that the cancellation is effective. You may use the following code for this purpose:

```
if (Thread.currentThread().isInterrupted())
{
    System.out.println("Search in " + path + " canceled.");
    return null;
}
```

If you're using `CachedThreadPool`, you may use the following code to observe the largest thread pool size during execution.

```
if (executor instanceof ThreadPoolExecutor)
    System.out.println("Largest pool size: "
            + ((ThreadPoolExecutor) executor).getLargestPoolSize());
```

Sample output:

```
Enter keyword (e.g. volatile): volatile
Found the first file that contains volatile: src\java\util\Locale.java
Search in src\com\sun\source\tree\TreeVisitor.java canceled.
Search in src\com\sun\org\apache\xml\internal\security\algorithms\Algorithm.java
canceled.
Search in src\com\sun\org\apache\bcel\internal\classfile\ConstantObject.java
```

```
canceled.
Search in src\com\sun\java\swing\plaf\motif\MotifLookAndFeel.java canceled.
Search in src\com\sun\corba\se\impl\interceptors\IORInfoImpl.java canceled.
Search in src\com\sun\security\auth\SolarisNumericUserPrincipal.java canceled.
        ......
Largest pool size: 271
```

```
Enter keyword (e.g. volatile): synchronized
Found the first file that contains synchronized:
src\com\sun\corba\se\impl\presentation\rmi\StubFactoryBase.java

Search in src\javax\swing\border\TitledBorder.java canceled.
Search in src\java\awt\datatransfer\DataFlavor.java canceled.
Search in src\com\sun\org\apache\xerces\internal\xpointer\ShortHandPointer.java
canceled.
Search in src\com\sun\org\apache\xpath\internal\ExtensionsProvider.java canceled.
Search in src\java\sql\SQLTransientConnectionException.java canceled.
        ......
Largest pool size: 52
```