Applying Java Functional Programming Features to ThreadJoinTest Case Study

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand how Java functional programming features are applied in a simple parallel program

See github.com/douglascraigschmidt/LiveLessons/tree/master/ThreadJoinTest/updated

Learning Objectives in this Part of the Lesson

- Understand how Java functional programming features are applied in a simple parallel program
  - This program searches for a list of phrases in the complete works of William Shakespeare

The Complete Works of William Shakespeare

Welcome to the Web's first edition of the Complete Works of William Shakespeare. This site has offered Shakespeare's plays and poetry to the Internet community since 1993.

```
Starting SearchStream
in thread 23 the phrase "Anon," was found at character offset 111628 in "The First Part of Henry VI"
in thread 20 the phrase "Anon," was found at character offset 30949 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 48850 in "The First Part of King Henry IV"
in thread 19 the phrase "Anon," was found at character offset 170485 in "The Tragedy of Hamlet"
in thread 20 the phrase "Anon," was found at character offset 49402 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 49640 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 50003 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 50140 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 50464 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 50486 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 51628 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 52190 in "The First Part of King Henry IV"
in thread 21 the phrase "Anon," was found at character offset 67832 in "Second Part of King Henry IV"
in thread 16 the phrase "Anon," was found at character offset 75139 in "The Comedy of Errors"
in thread 16 the phrase "Anon," was found at character offset 76511 in "The Comedy of Errors"
in thread 31 the phrase "Anon," was found at character offset 34971 in "The Tragedy of Macbeth"
in thread 40 the phrase "Anon," was found at character offset 37045 in "The Tragedy of Romeo & Juliet"
in thread 40 the phrase "Anon," was found at character offset 46837 in "The Tragedy of Romeo & Juliet"
Ending SearchStream
```

See shakespeare.mit.edu

Overview of the ThreadJoinTest Program

Overview of the ThreadJoinTest Program

- Use Java functional programming features to start() & join() threads to search for phrases in works of William Shakespeare

```
workerThreads
   .forEach(Thread::start);

workerThreads
   .forEach(thread ->
     { try { thread.join(); }
     catch (Exception e)
     { throw new RuntimeException(e);
     }});
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/ThreadJoinTest/updated

Overview of the ThreadJoinTest Program

- This program is "embarrassingly parallel"

```
Starting SearchStream
in thread 23 the phrase "Anon," was found at character offset 111628 in "The First Part of Henry VI"
in thread 20 the phrase "Anon," was found at character offset 30949 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 48850 in "The First Part of King Henry IV"
in thread 19 the phrase "Anon," was found at character offset 170485 in "The Tragedy of Hamlet"
in thread 20 the phrase "Anon," was found at character offset 49402 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 49640 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 50003 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 50140 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 50464 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 50486 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 51628 in "The First Part of King Henry IV"
in thread 20 the phrase "Anon," was found at character offset 52190 in "The First Part of King Henry IV"
in thread 21 the phrase "Anon," was found at character offset 67832 in "Second Part of King Henry IV"
in thread 16 the phrase "Anon," was found at character offset 75139 in "The Comedy of Errors"
in thread 16 the phrase "Anon," was found at character offset 76511 in "The Comedy of Errors"
in thread 31 the phrase "Anon," was found at character offset 34971 in "The Tragedy of Macbeth"
in thread 40 the phrase "Anon," was found at character offset 37045 in "The Tragedy of Romeo & Juliet"
in thread 40 the phrase "Anon," was found at character offset 46837 in "The Tragedy of Romeo & Juliet"
Ending SearchStream
```

See en.wikipedia.org/wiki/Embarrassingly_parallel

Overview of the ThreadJoinTest Program

- This program is "embarrassingly parallel"
  - i.e., there are no data dependencies between worker threads

See en.wikipedia.org/wiki/Embarrassingly_parallel

Overview of the ThreadJoinTest Program

- The program obtains the complete works of Shakespeare & a list of phrases from two text files

```
List<String> mInputList =
  TestDataFactory.getInput
    (sSHAKESPEARE_DATA_FILE,
     "@");
...
@The Tragedy of Hamlet
...
@The Tragedy of Julius Caesar
...
@The Tragedy of Macbeth
...
```

Each work begins with a '@' character

Overview of the ThreadJoinTest Program

- The program obtains the complete works of Shakespeare & a list of phrases from two text files

```
List<String> mPhrasesToFind =
  TestDataFactory.getPhraseList
    (sPHRASE_LIST_FILE);
...
Neither a borrower nor a lender be
Beware the Ides of March
Brevity is the soul of wit
All that glisters is not gold
Sit you down, father; rest you
my kingdom for a horse!
...
```

Each phrase appears on a separate line

Overview of the ThreadJoinTest Program

- Return the input data in the given file as a list of non-empty strings

```
static List<String> getInput(String file, String splitter) {
  URI uri = ClassLoader.getSystemResource(file).toURI();


  String bytes = new String(Files.readAllBytes
                                    (Paths.get(uri)));



  return Pattern
    .compile(splitter)
    .splitAsStream(bytes)
    .filter(((Predicate<String>) String::isEmpty).negate())
    .collect(toList());
}
```

These methods also demonstrate some Java streams features!

Overview of the ThreadJoinTest Program

- Return the input data in the given file as a list of non-empty strings

```
static List<String> getInput(String file, String splitter) {
  URI uri = ClassLoader.getSystemResource(file).toURI();
```

Convert the file name into a path name

```
  String bytes = new String(Files.readAllBytes
                                    (Paths.get(uri)));



  return Pattern
    .compile(splitter)
    .splitAsStream(bytes)
    .filter(((Predicate<String>) String::isEmpty).negate())
    .collect(toList());
}
```

Overview of the ThreadJoinTest Program

- Return the input data in the given file as a list of non-empty strings

```
static List<String> getInput(String file, String splitter) {
  URI uri = ClassLoader.getSystemResource(file).toURI();


  String bytes = new String(Files.readAllBytes
                                  (Paths.get(uri)));

  return Pattern
    .compile(splitter)
    .splitAsStream(bytes)
    .filter(((Predicate<String>) String::isEmpty).negate())
    .collect(toList());
}
```

Open the file & read all the bytes

Overview of the ThreadJoinTest Program

- Return the input data in the given file as a list of non-empty strings

```
static List<String> getInput(String file, String splitter) {
  URI uri = ClassLoader.getSystemResource(file).toURI();


  String bytes = new String(Files.readAllBytes
                                    (Paths.get(uri)));


  return Pattern
    .compile(splitter)
    .splitAsStream(bytes)
    .filter(((Predicate<String>) String::isEmpty).negate())
    .collect(toList());
}
```

Compile a regular expression used to split the file into a stream of strings

See docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

Overview of the ThreadJoinTest Program

- Return the input data in the given file as a list of non-empty strings

```
static List<String> getInput(String file, String splitter) {
  URI uri = ClassLoader.getSystemResource(file).toURI();


  String bytes = new String(Files.readAllBytes
                                    (Paths.get(uri)));


  return Pattern
    .compile(splitter)
    .splitAsStream(bytes)
    .filter(((Predicate<String>) String::isEmpty).negate())
    .collect(toList());
}
```

Filter out any empty strings in the stream

Overview of the ThreadJoinTest Program

- Return the input data in the given file as a list of non-empty strings

```
static List<String> getInput(String file, String splitter) {
  URI uri = ClassLoader.getSystemResource(file).toURI();


  String bytes = new String(Files.readAllBytes
                                        (Paths.get(uri)));



  return Pattern
    .compile(splitter)
    .splitAsStream(bytes)
    .filter(((Predicate<String>) String::isEmpty).negate())
    .collect(toList());
}
```

Collect the results into a list of strings

Overview of the ThreadJoinTest Program

- Return the phrase list in the file as a list of non-empty strings

```
static List<String> getPhraseList(String file) {
  return Files
    .lines(Paths
          .get(ClassLoader.getSystemResource(file).toURI()))



    .filter(((Predicate<String>) String::isEmpty).negate())


    .collect(toList());
}
```

Overview of the ThreadJoinTest Program

- Return the phrase list in the file as a list of non-empty strings

```
static List<String> getPhraseList(String file) {
  return Files
    .lines(Paths
        .get(ClassLoader.getSystemResource(file).toURI()))

    .filter(((Predicate<String>) String::isEmpty).negate())


    .collect(toList());
}
```

Read all lines from file into a stream

Overview of the ThreadJoinTest Program

- Return the phrase list in the file as a list of non-empty strings

```
static List<String> getPhraseList(String file) {
  return Files
    .lines(Paths
           .get(ClassLoader.getSystemResource(file).toURI()))
```

Filter out any empty strings in the stream

```
    .filter(((Predicate<String>) String::isEmpty).negate())


    .collect(toList());
}
```

Overview of the ThreadJoinTest Program

- Return the phrase list in the file as a list of non-empty strings

```
static List<String> getPhraseList(String file) {
  return Files
    .lines(Paths
           .get(ClassLoader.getSystemResource(file).toURI()))



    .filter(((Predicate<String>) String::isEmpty).negate())


    .collect(toList());
}
```

Collect the results into a list of strings

Overview of the ThreadJoinTest Program

- The main program creates & runs an instance of SearchOneShotThreadJoin

```
public void main(String[] args) {
  new SearchOneShotThreadJoin()
    .run();
}
```

Create/run an object to search for all phrases in parallel

See ThreadJoinTest/updated/src/main/java/ThreadJoinTest.java

End of Applying Java Functional Programming Features to ThreadJoinTest Case Study