The ex45 Case Study (Part 1): Overview & Utility Classes

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

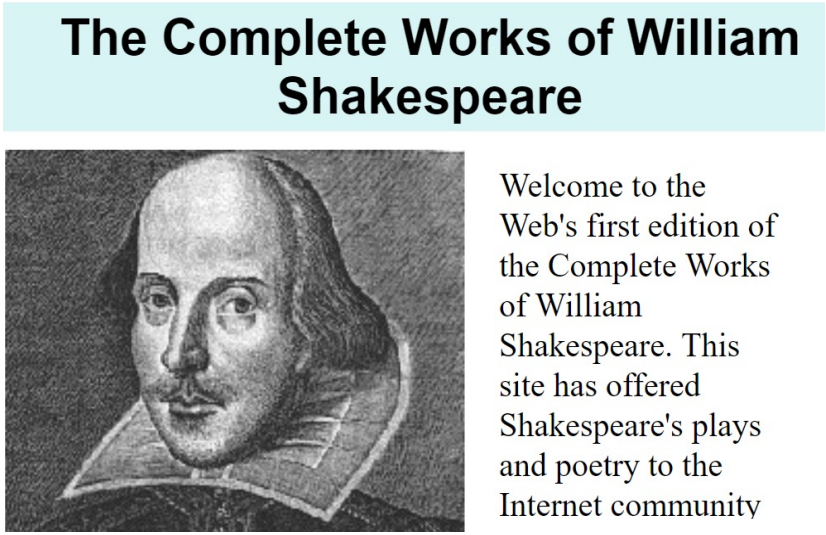
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of the ex45 case study that searches for words & phrases in the works of Shakespeare



See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex45

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of the ex45 case study that searches for words & phrases in the works of Shakespeare
  - The utility classes use the Java streams framework extensively

RegexUtils	
RegexUtils()	
makeRegex(List<String>)	String
getFirstLine(String)	String

BardDataFactory	
BardDataFactory()	
getInput(String, String)	List<String>?

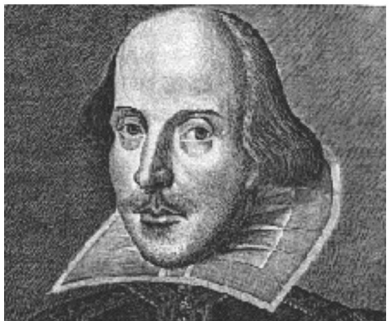
See www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

Structure & Functionality of the ex45 Case Study

Structure & Functionality of the ex45 Case Study

- This program shows how Java regex methods can be used in conjunction with Java sequential streams to search the complete works of Shakespeare for certain words & phrases

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

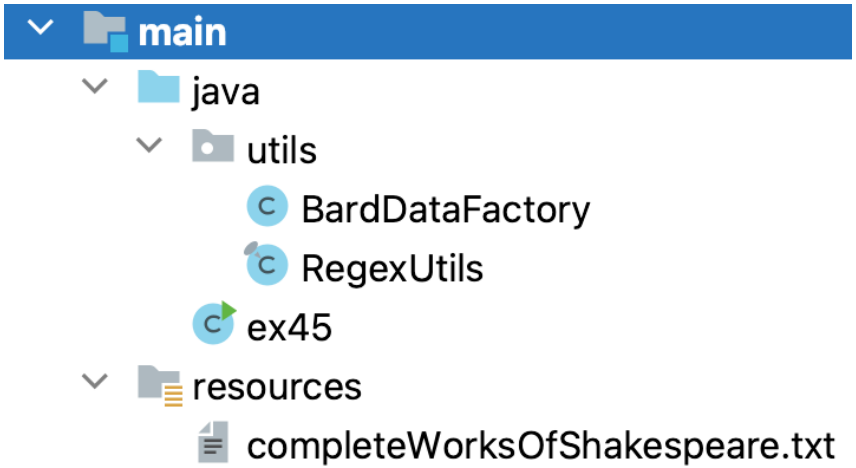
ex45 [:ex45.main()]

```
Number of works that match is 13 out of a total of 37
The Tragedy of Othello, Moor of Venice
"lord, your true" [131546]
"lord? with whom? how am I false" [131908]
The Tragedy of Titus Andronicus
"lord, sith true" [15533]
The History of Troilus and Cressida
"lord, will you be true" [120358]
The Taming of the Shrew
"LORD. 'Tis very true" [5568]
The Tragedy of King Lear
"lord, and true" [6392]
The Tragedy of Julius Caesar
"lord, are false" [108997]
Cymbeline
"LORD. [Aside] If it be a sin to make a true" [12336]
The Tragedy of Hamlet, Prince of Denmark
"lord, 'tis true" [20976]
The First Part of King Henry IV
```

See shakespeare.mit.edu

Structure & Functionality of the ex45 Case Study

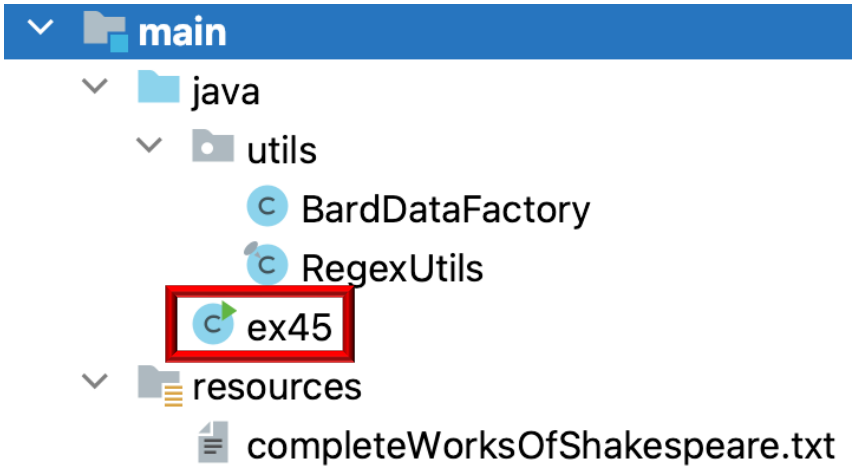
- The ex45 project source code is organized into several files & packages



See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex45

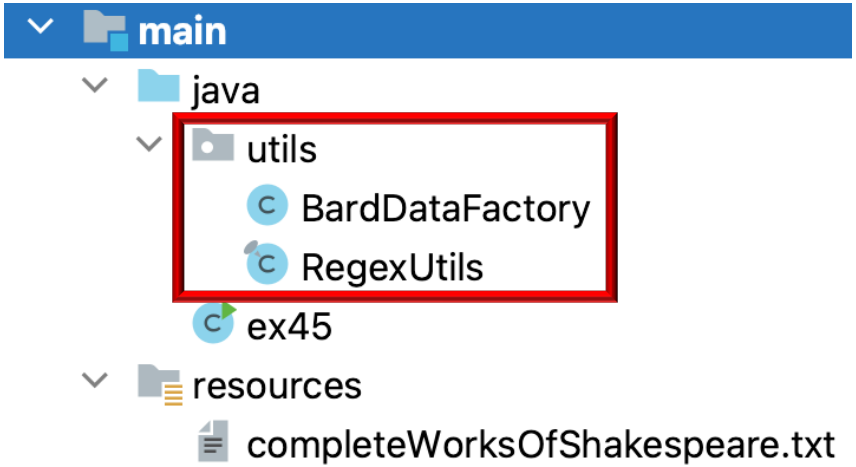
Structure & Functionality of the ex45 Case Study

- The ex45 project source code is organized into several files & packages
  - ex45
    - The main test driver program searches for words & phrases in the complete works of William Shakespeare



Structure & Functionality of the ex45 Case Study

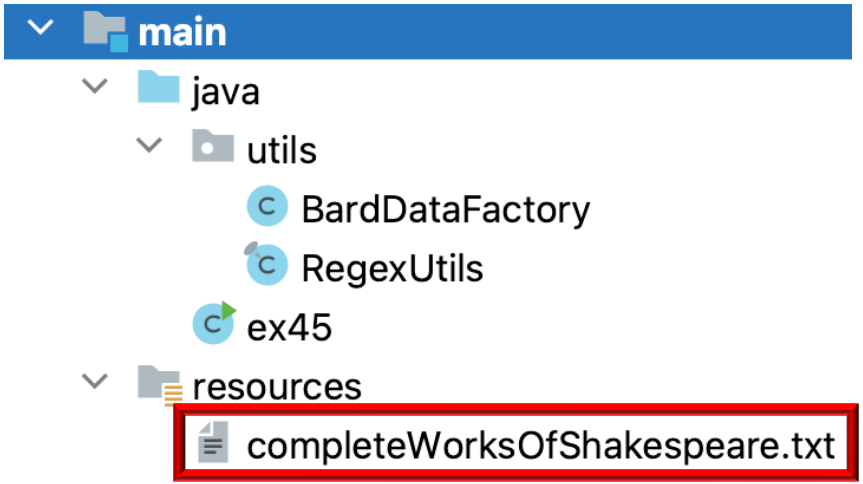
- The ex45 project source code is organized into several files & packages
  - ex45
  - utils
    - Contains general-purpose reusable classes for reading input from files & handling regular expressions



See christirapt.wordpress.com/2017/11/24/why-is-shakespeare-called-the-bard

Structure & Functionality of the ex45 Case Study

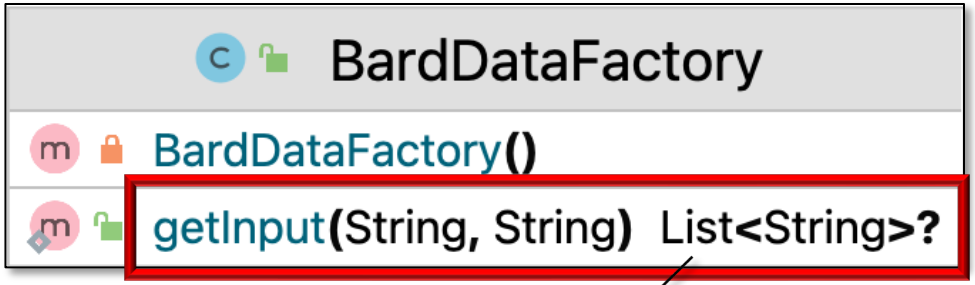
- The ex45 project source code is organized into several files & packages
  - ex45
  - utils
  - resources
    - Contains the complete works of Shakespeare



Methods in the BardDataFactory Class

Methods in the BardDataFactory Class

- The BardDataFactory utility class applies Java streams to read the contents of a text file & return this content as lists of strings

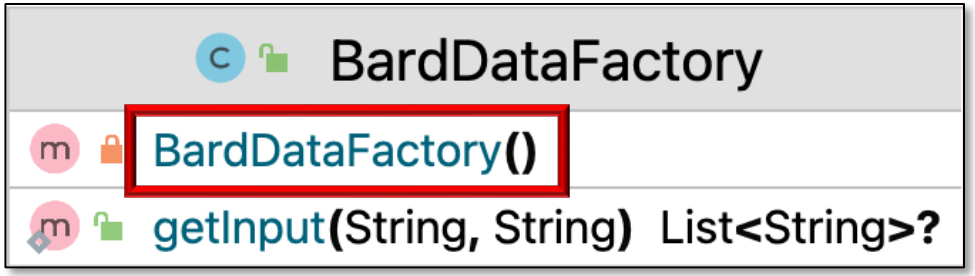


Each string contains a work of Shakepeare

See [Java8/ex45/src/main/java/utils/BardDataFactory.java]

Methods in the BardDataFactory Class

- The BardDataFactory utility class applies Java streams to read the contents of a text file & return this content as lists of strings

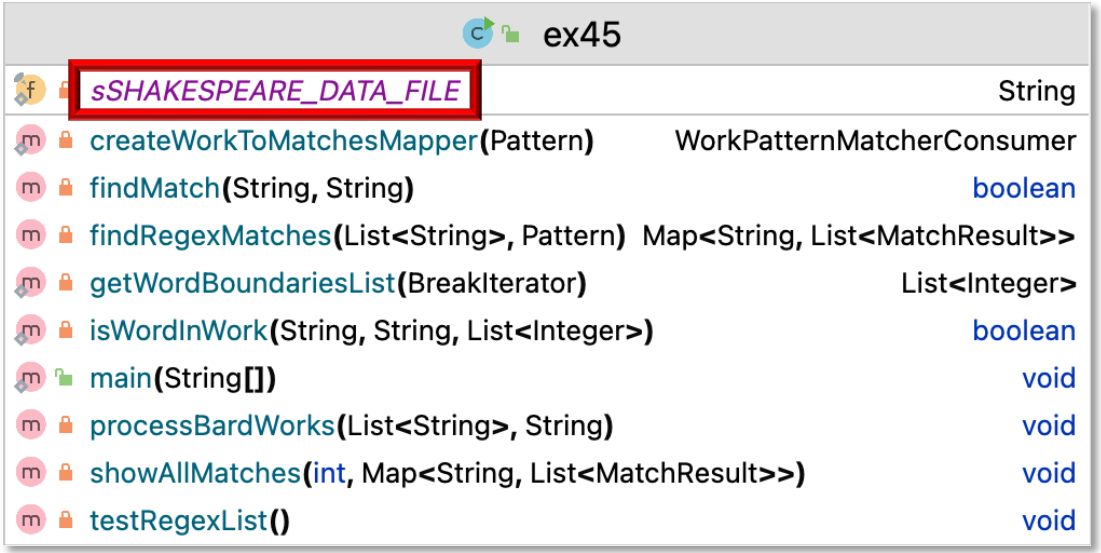


A utility class in Java is defined as final & has only static methods, no non-static state, & a private constructor

See www.quora.com/What-is-the-best-way-to-write-utility-classes-in-Java/answer/Jon-Harley

Methods in the BardDataFactory Class

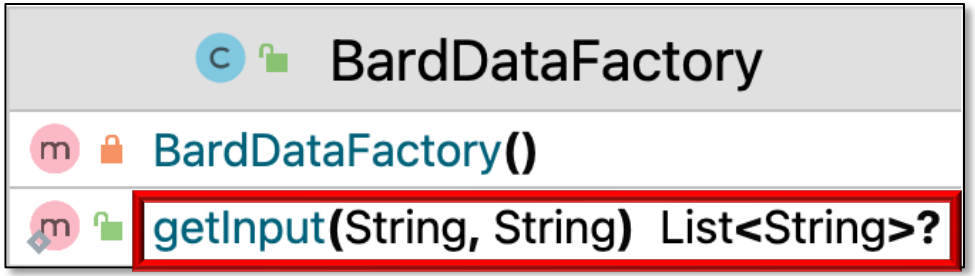
- BardDataFactory reads works of Shakespeare to search from a text file



Methods in the BardDataFactory Class

- BardDataFactory reads works of Shakespeare to search from a text file

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



Each work begins with '@', which is matched by a Java regular expression

Methods in the BardDataFactory Class

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

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


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

  return Pattern
     .compile(split)
     .splitAsStream(bytes)
     .filter(string -> !string.isEmpty())
     .toList();
  ...
```

See Java8/ex45/src/main/java/utils/BardDataFactory.java

Methods in the BardDataFactory Class

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

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

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

  return Pattern
     .compile(split)
     .splitAsStream(bytes)
     .filter(string -> !string.isEmpty())
     .toList();
  ...
```

Params include the file name & split character

Methods in the BardDataFactory Class

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

```
static List<String> getInput(String file, String split) {
  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(split)
     .splitAsStream(bytes)
     .filter(string -> !string.isEmpty())
     .toList();
  ...
```

Methods in the BardDataFactory Class

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

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


  String bytes = new String(Files.readAllBytes
                                  (Paths.get(uri)));
  return Pattern
     .compile(split)
     .splitAsStream(bytes)
     .filter(string -> !string.isEmpty())
     .toList();
  ...
```

Open the file & read all the bytes

Methods in the BardDataFactory Class

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

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


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

  return Pattern
     .compile(split)
     .splitAsStream(bytes)
     .filter(string -> !string.isEmpty())
     .toList();
  ...
```

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

Recall that the split param is the string "@"

Methods in the BardDataFactory Class

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

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


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

  return Pattern
     .compile(split)
     .splitAsStream(bytes)
     .filter(string -> !string.isEmpty())
     .toList();
  ...
```

Filter out any empty strings in the stream

Methods in the BardDataFactory Class

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

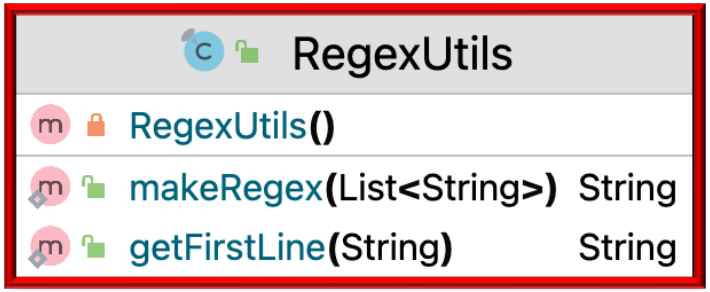
```
static List<String> getInput(String file, String split) {
  URI uri = ClassLoader.getSystemResource(file).toURI();


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

  return Pattern
     .compile(split)
     .splitAsStream(bytes)
     .filter(string -> !string.isEmpty())
     .toList();
  ...
```

Return the results as a list of strings

Methods in the RegexUtils Class

Methods in the RegexUtils Class

- The RegexUtils utility class provides helper methods for handling Java regular expressions

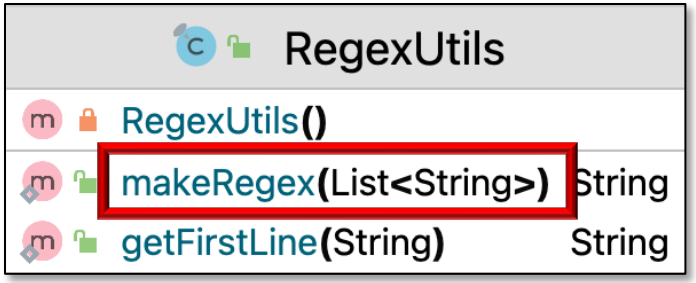


See Java8/ex45/src/main/java/utils/RegexUtils.java

Methods in the RegexUtils Class

`makeRegex()`

- Converts a list of strings containing queries into a single regular expression string
- The returned string matches any string containing any of the queries as whole words, regardless of case
- It uses the Java Stream API for transformations



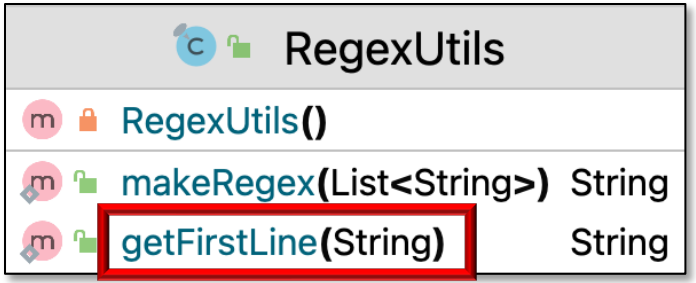

```
var wordsToMatch = List
  .of("Cat","Dog","Mouse","sox");

var regexString =
  makeRegex(wordsToMatch);
```

```
"(?i).*\b(\QCat\E|\QDog\E|\QMouse\E|\Qsox\E)\b.*"
```

Methods in the RegexUtils Class

`getFirstLine()`

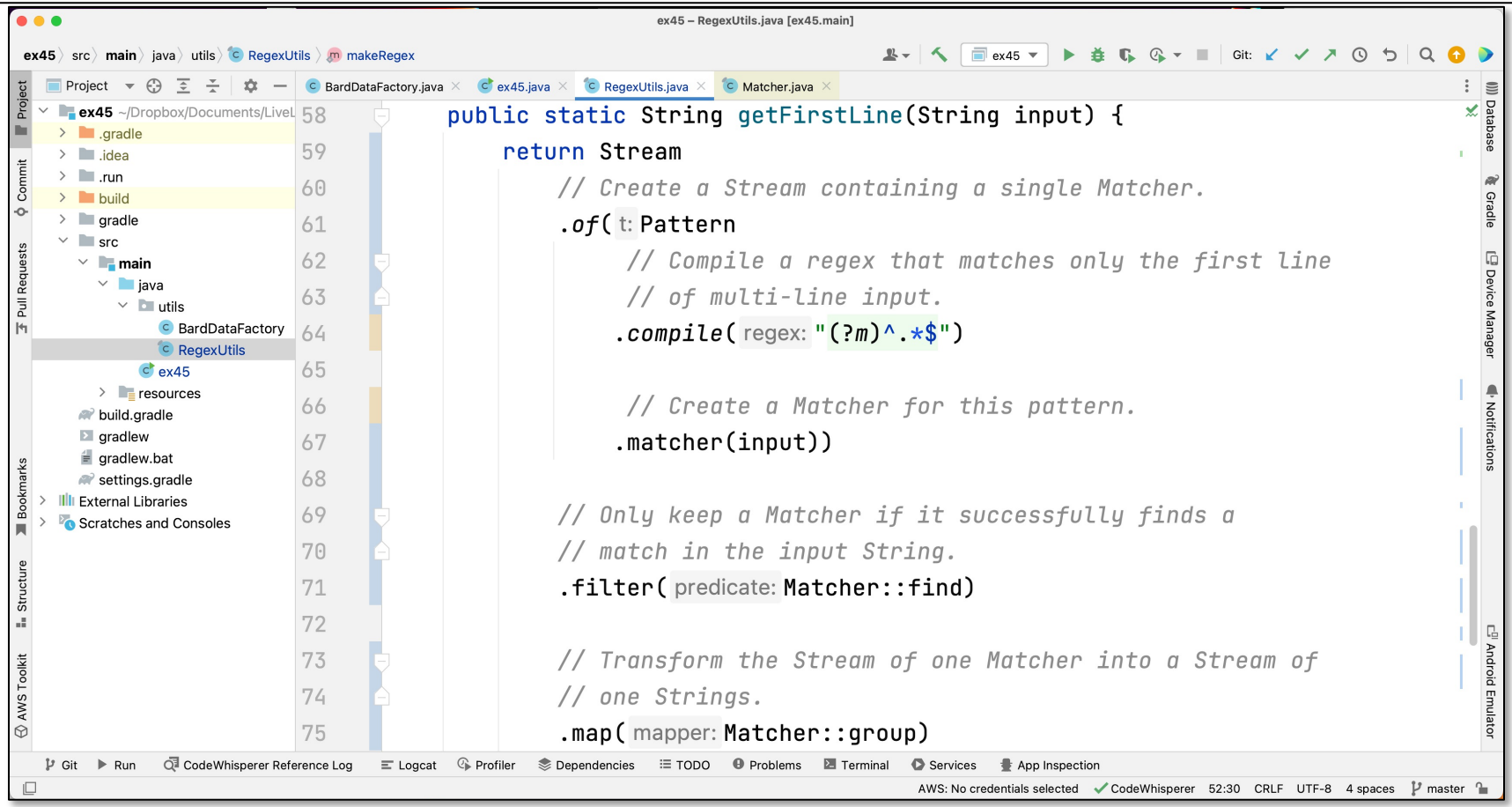
- Returns the first line of the given input string
  - i.e., the title of a work of Shakespeare
- The method uses Java's Pattern & Matcher classes along with the Java Stream API




```
RegexUtils
    .getFirstLine(work)
```

```
"The Tragedy of Hamlet, Prince of Denmark"
```

Methods in the RegexUtils Class



See Java8/ex45/src/main/java/utils/RegexUtils.java

End of the ex45 Case Study: Overview & Utility Classes