Lambda Hands-on Lab

Stuart Marks
Principal Member of Technical Staff, Oracle

Simon Ritter
Technology Evangelist, Oracle

Angela Caicedo
Java Evangelist, Oracle

1. Getting Ready

1.1. **Install NetBeans 7.4** (https://netbeans.org)

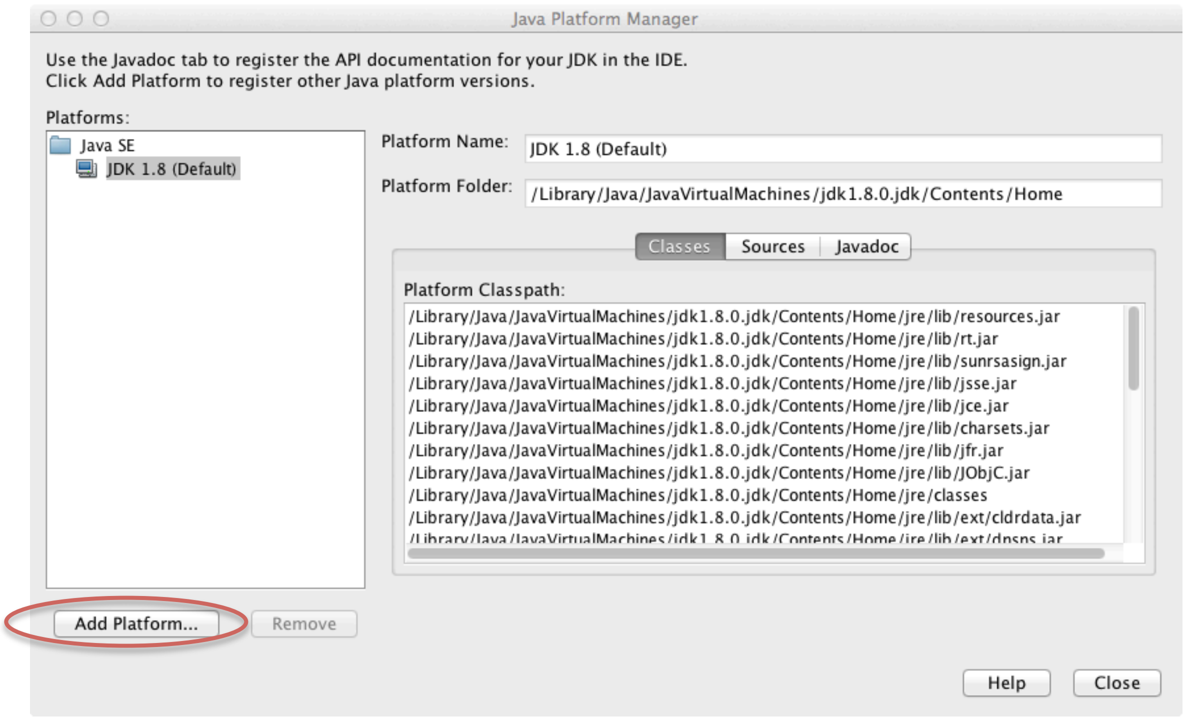
- Make sure installation is successful.

1.2. **Install JDK 8 Early Access latest release.** (https://jdk8.java.net/download.html)

- By the time this documentation was done, the latest build was b115.
- Make sure you also download the JDK zip documentation.

1.3. **Configure NetBeans** to use the installed JDK.

- From the top menu select **Tools** and then select **Java Platforms**
- Click on Add Platform



Use the Javadoc tab to register the API documentation for your JDK in the IDE.
Click Add Platform to register other Java platform versions.

JDK 1.8 (Default)

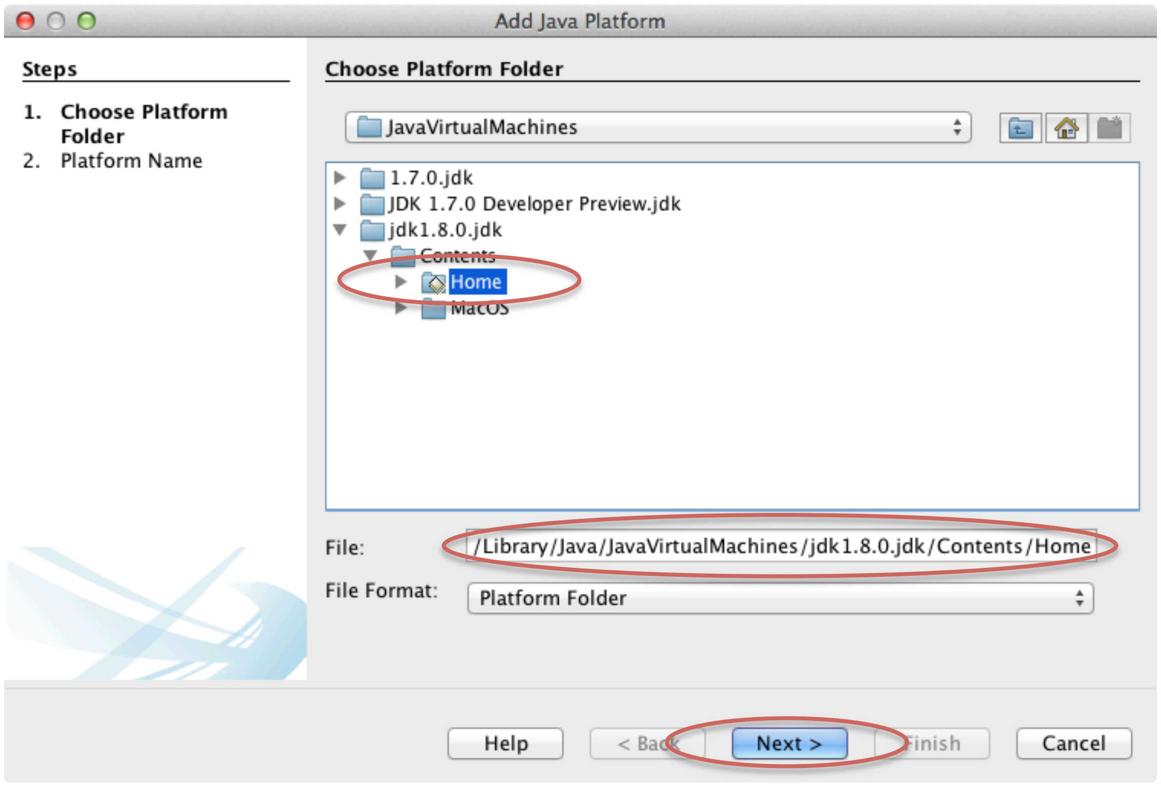
Platform Name: JDK 1.8 (Default)

Platform Folder: /Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home

Platform Classpath:

```
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/resources.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/rt.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/sunrsasign.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/jsse.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/jce.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/charsets.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/jfr.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/JObjC.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/classes
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/ext/cldrdata.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/ext/dnsns.jar
```

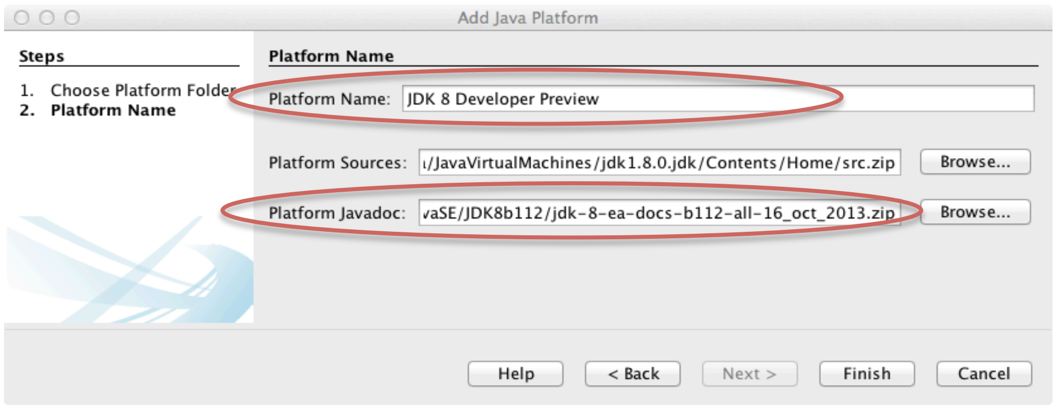
- Navigate and select the installation directory for the JDK 8, and click on Next.



File: /Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home

File Format: Platform Folder

- Enter **JDK 8 Developer Preview** for Platform Name.
- Specify the location for the Platform Javadoc: browse and select the location of the JDK documentation you downloaded.

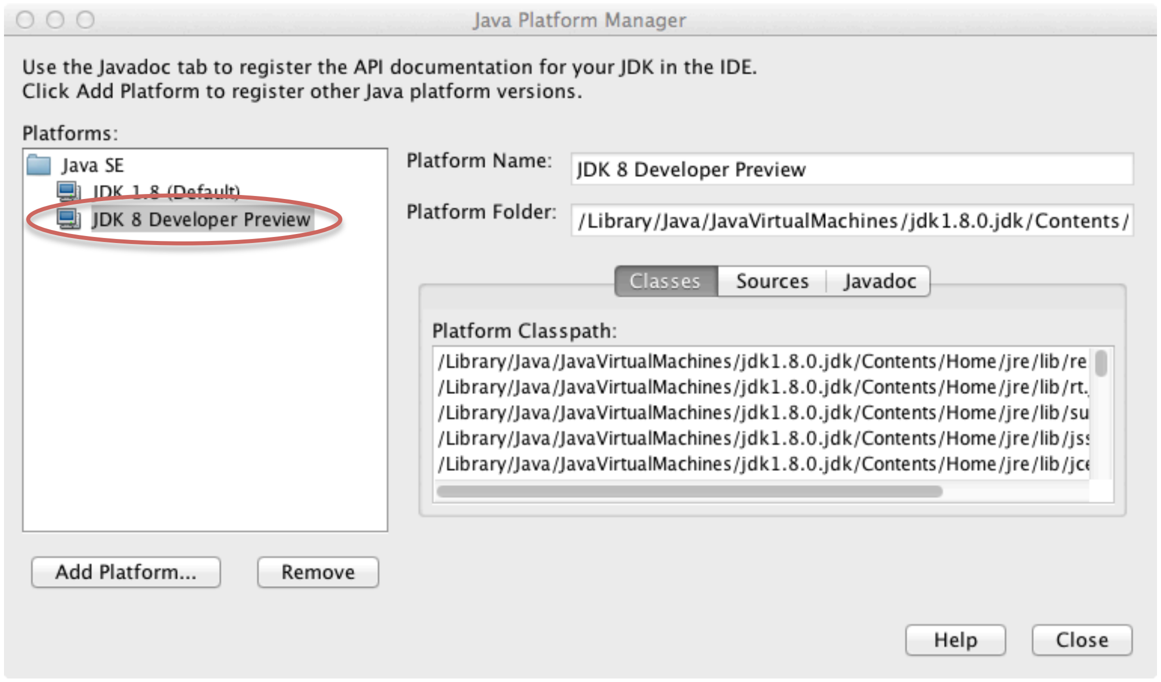


Platform Name: JDK 8 Developer Preview

Platform Sources: ı/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/src.zip

Platform Javadoc: vaSE/JDK8b112/jdk-8-ea-docs-b112-all-16_oct_2013.zip

- Click on Finish.
- You should see now a new listed Java SE platform



Java Platform Manager

Use the Javadoc tab to register the API documentation for your JDK in the IDE.
Click Add Platform to register other Java platform versions.

Platforms:

- Java SE
  - JDK 1.8 (Default)
  - JDK 8 Developer Preview

Platform Name: JDK 8 Developer Preview

Platform Folder: /Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/

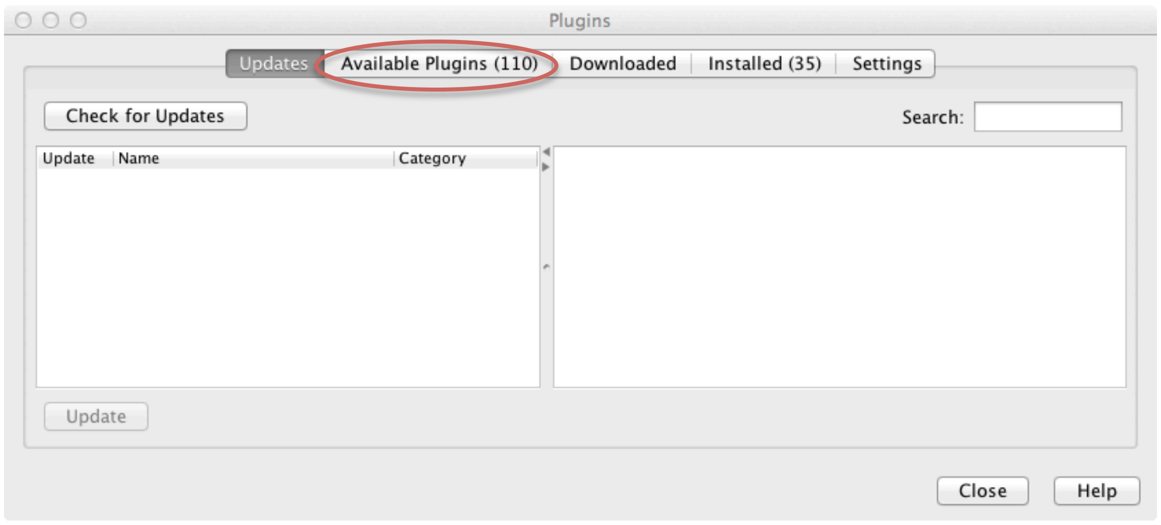
Platform Classpath:

```
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/re
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/rt.
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/su
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/jss
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/jre/lib/jce
```

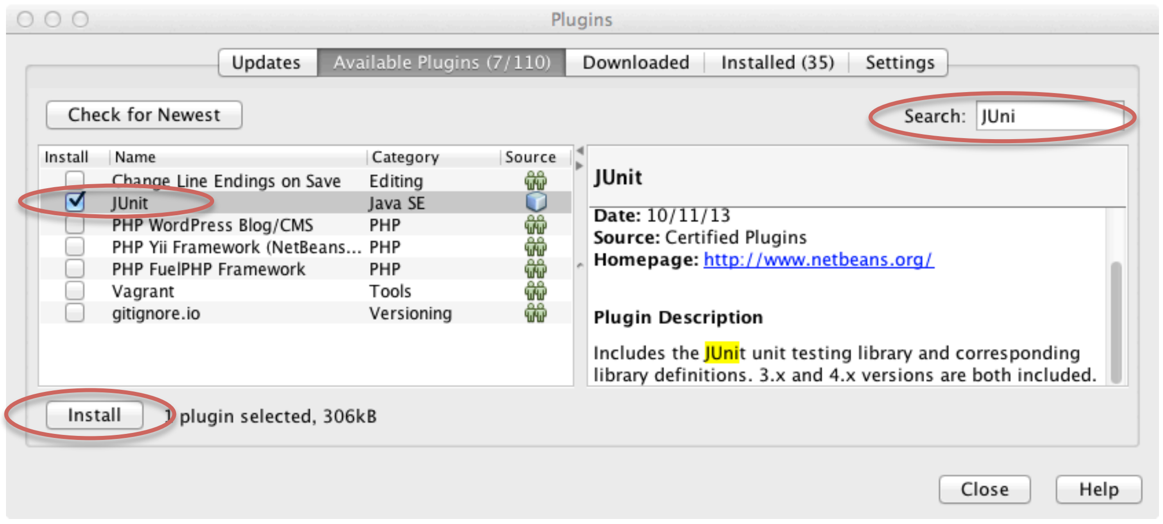
- Close your Java Platform Manager.

1.4. Installing JUnit with NetBeans Update Center

- From the main menu, select **Tools > Plugins**
- Click on **Available Plugins** at the Plugins window



- Type **"JUni"** in the Search Text area. (Don't worry if the list from the image looks different from your list)
- Click the checkbox for JUnit installation
- Click Install



Install	Name	Category	Source
☐	Change Line Endings on Save	Editing	
☑	JUnit	Java SE	
☐	PHP WordPress Blog/CMS	PHP	
☐	PHP Yii Framework (NetBeans...	PHP	
☐	PHP FuelPHP Framework	PHP	
☐	Vagrant	Tools	
☐	gitignore.io	Versioning	

JUnit

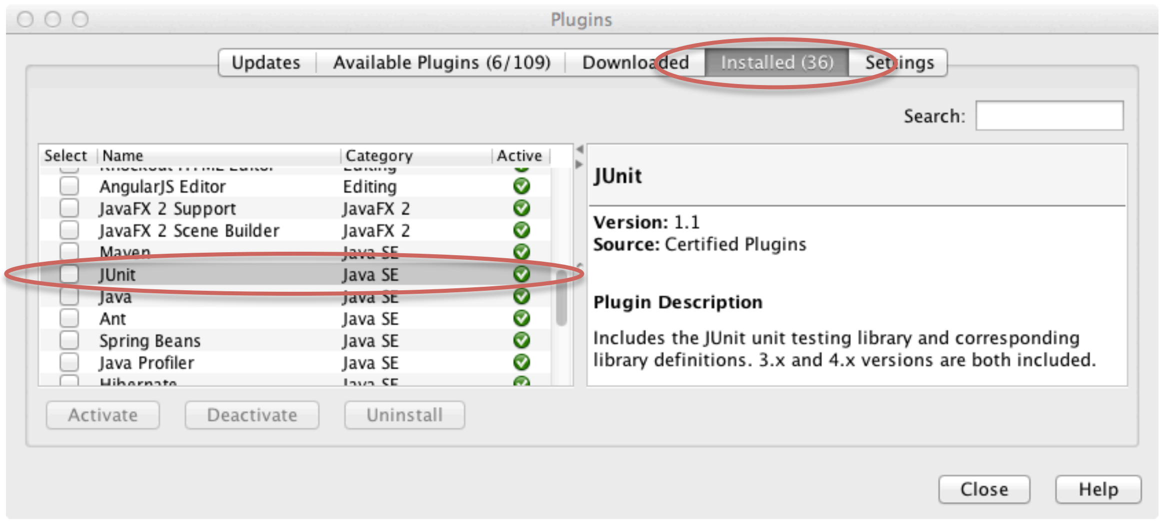
Date: 10/11/13
Source: Certified Plugins
Homepage: http://www.netbeans.org/

Plugin Description

Includes the JUnit unit testing library and corresponding library definitions. 3.x and 4.x versions are both included.

1 plugin selected, 306kB

- The NetBeans IDE installer gets displayed. Just follow the instructions for the installation.
- Once the installation finished, make sure JUnit is now part of the Installed plugins and it's **Active**



Plugins

Updates | Available Plugins (6/109) | Downloaded | Installed (36) | Settings

Search:

Select	Name	Category	Active
☐	AngularJS Editor	Editing	✓
☐	JavaFX 2 Support	JavaFX 2	✓
☐	JavaFX 2 Scene Builder	JavaFX 2	✓
☐	Maven	Java SE	✓
☐	JUnit	Java SE	✓
☐	Java	Java SE	✓
☐	Ant	Java SE	✓
☐	Spring Beans	Java SE	✓
☐	Java Profiler	Java SE	✓
☐	Hibernate	Java SE	✓

JUnit

Version: 1.1
Source: Certified Plugins

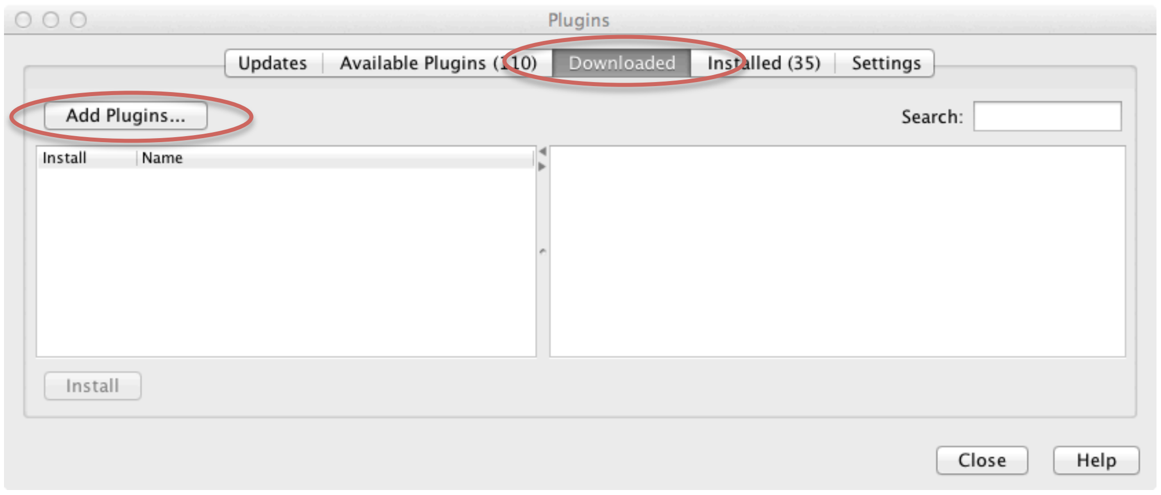
Plugin Description

Includes the JUnit unit testing library and corresponding library definitions. 3.x and 4.x versions are both included.

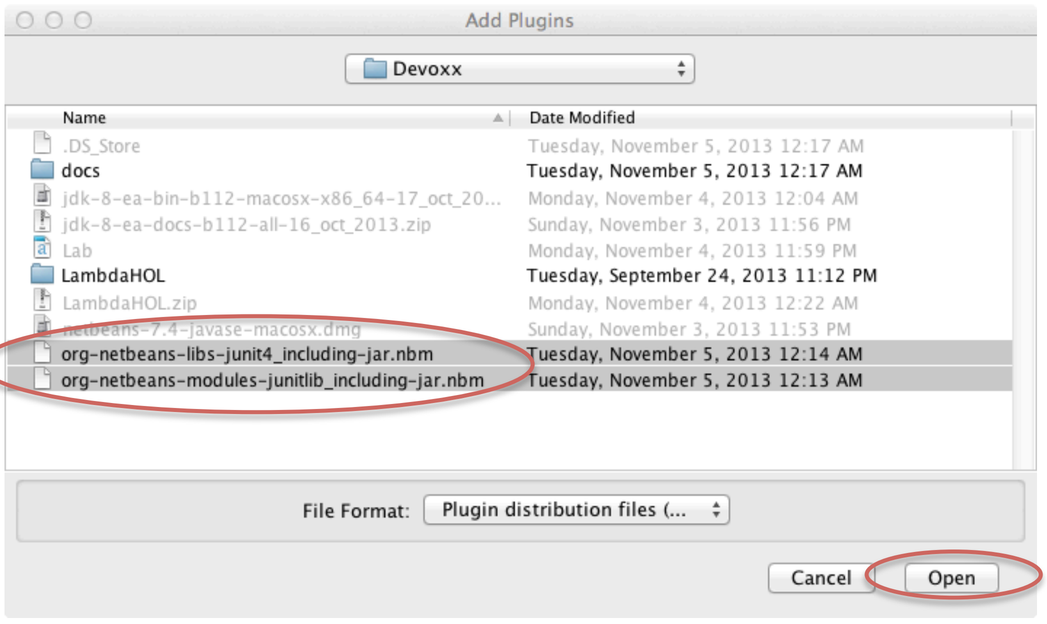
- Close the plugin window.

1.5. Installing JUnit offline

- If you don't have Internet connection, you can still install the JUnit plugin from the provided software for this lab.
- In the lab software, you should have two files named: **org-netbeans-libs-junit4_including-jar.nbm** and **org-netbeans-modules-junitlib_including-jar.nbm**.
- From the main menu, select **Tools > Plugins**
- Click on **Downloaded** at the Plugins window
- Click on **Add Plugins…** button

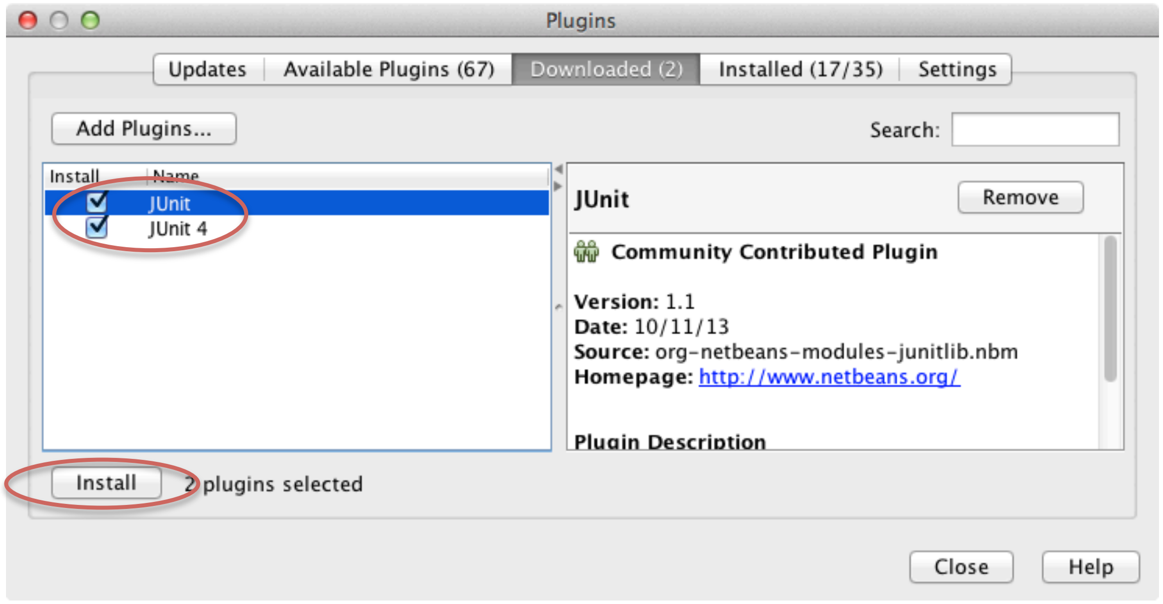


- Look for the lab material and select **org-netbeans-libs-junit4_including-jar.nbm** and **org-netbeans-modules-junitlib_including-jar.nbm** files.
- Click on Open



Name	Date Modified
.DS_Store	Tuesday, November 5, 2013 12:17 AM
docs	Tuesday, November 5, 2013 12:17 AM
jdk-8-ea-bin-b112-macosx-x86_64-17_oct_20...	Monday, November 4, 2013 12:04 AM
jdk-8-ea-docs-b112-all-16_oct_2013.zip	Sunday, November 3, 2013 11:56 PM
Lab	Monday, November 4, 2013 11:59 PM
LambdaHOL	Tuesday, September 24, 2013 11:12 PM
LambdaHOL.zip	Monday, November 4, 2013 12:22 AM
netbeans-7.4-javase-macosx.dmg	Sunday, November 3, 2013 11:53 PM
org-netbeans-libs-junit4_including-jar.nbm	Tuesday, November 5, 2013 12:14 AM
org-netbeans-modules-junitlib_including-jar.nbm	Tuesday, November 5, 2013 12:13 AM

- Now you should see JUnit and JUnit 4 selected for installation
- Click on Install button


Plugins

Updates | Available Plugins (67) | Downloaded (2) | Installed (17/35) | Settings

Add Plugins... Search:

Install	Name
☑	JUnit
☑	JUnit 4

JUnit Remove

Community Contributed Plugin

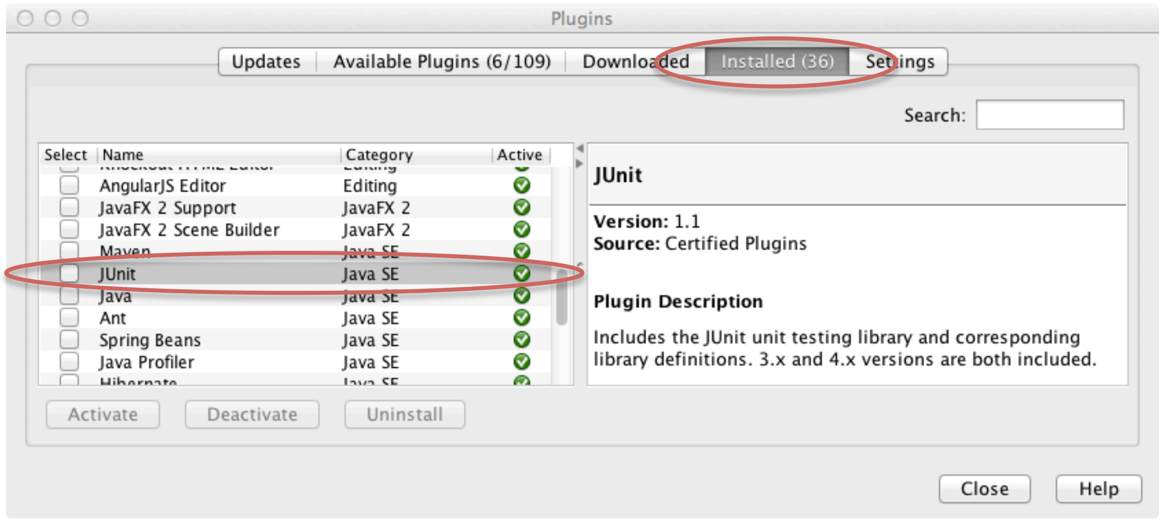
Version: 1.1
Date: 10/11/13
Source: org-netbeans-modules-junitlib.nbm
Homepage: http://www.netbeans.org/

Plugin Description

Install 2 plugins selected

Close Help

- The NetBeans IDE installer gets displayed. Just follow the instructions for the installation.
- Once the installation finished, make sure JUnit is now part of the Installed plugins and it's **Active**


Plugins

Updates | Available Plugins (6/109) | Downloaded | Installed (36) | Settings

Search:

Select	Name	Category	Active
☐	AngularJS Editor	Editing	✓
☐	JavaFX 2 Support	JavaFX 2	✓
☐	JavaFX 2 Scene Builder	JavaFX 2	✓
☐	Maven	Java SE	✓
☐	JUnit	Java SE	✓
☐	Java	Java SE	✓
☐	Ant	Java SE	✓
☐	Spring Beans	Java SE	✓
☐	Java Profiler	Java SE	✓
☐	Hibernate	Java SE	✓

JUnit

Version: 1.1
Source: Certified Plugins

Plugin Description

Includes the JUnit unit testing library and corresponding library definitions. 3.x and 4.x versions are both included.

Activate Deactivate Uninstall

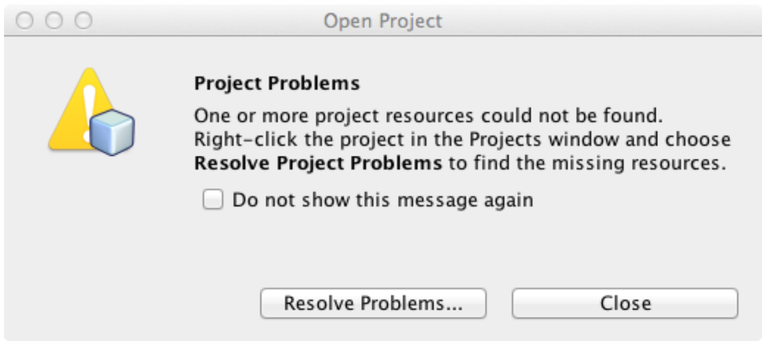
Close Help

- Close the plugin window.

2. Starting with the exercises

2.1 Open LambdaHOL project

- You might get a message about missing some project resources. If this is your case, follow the next steps to fix the project, otherwise continue with the exercises.
- Close the warning window



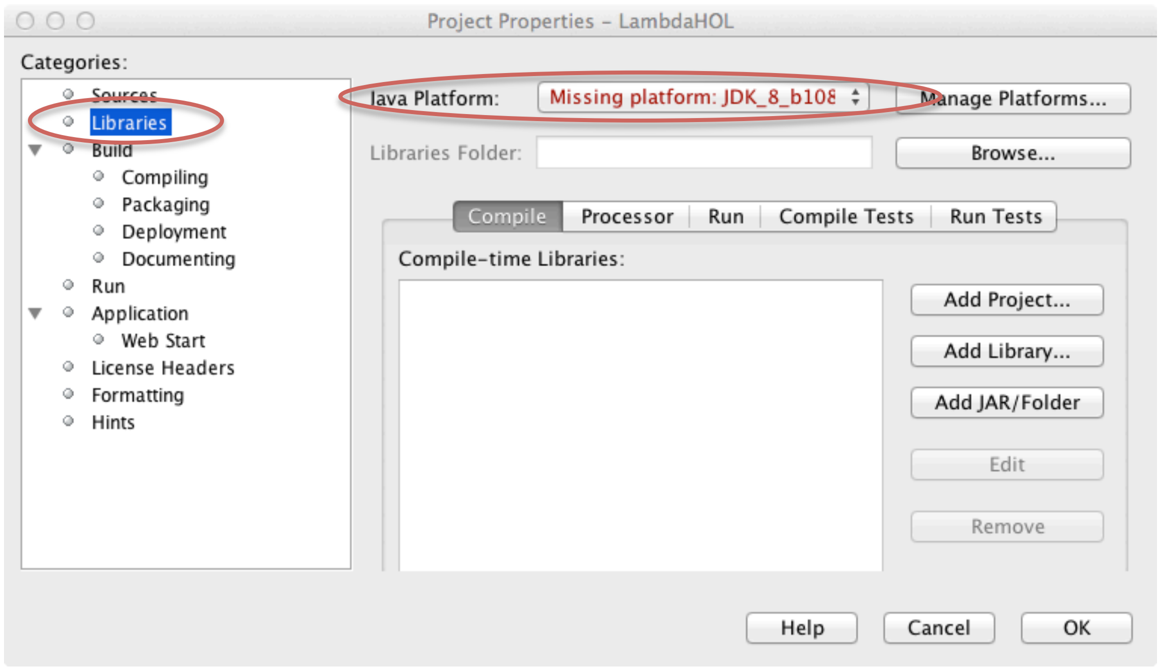
Open Project

Project Problems

One or more project resources could not be found.
Right-click the project in the Projects window and choose **Resolve Project Problems** to find the missing resources.

☐ Do not show this message again

- Right click on your project, and select **Properties**
- From the **Categories** list, select **Libraries**
- On the right hand side, you might notice that the Java Platform is missing. This error is cause if you used a different name for the new added Java Platform, or because some misspelling in the Platform name.



Java Platform: Missing platform: JDK_8_b108

Libraries Folder:

Compile-time Libraries:

- From the Java Platform drop-down list select the **JDK 8 Developer Preview**, and click Ok.
- Now your project should show no errors.

2.2. Exercise Overview

The project is structured as a set of JUnit tests. Open the Test Packages folder, and in the "exercises" package, open the file Exercises.java.

Each exercise (except for the first) is a single test with a problem statement in a comment, a local variable with a TODO comment next to it, and some assertions at the end. Each test exercise has an @Ignore annotation on it, which causes the JUnit framework not to run the test. Run the tests in this file using Alt-F6 (Cmd-F6 on Macintosh). If you do this now, the result should be no tests passed, no failures, and 12 tests skipped.

Your task is, for each exercise, to replace the TODO comment with code so that the assertions are satisfied. Remove the @Ignore annotation from a test in order to have JUnit run it.

The first exercise doesn't have any assertions. It's just there to help you get started. It's also a useful place to try out some experiments without risking getting a red bar.

Continue through the exercises, removing each @Ignore tag and replacing the TODO comment with code to make the test work. Try to keep the bar green! When you're finished, all 12 tests should pass, and none should be skipped.

There are two additional files in the exercises.answers package. The first, ExerciseAnswers.java, is the set of tests with an answer filled in. Look in here if you want some hints as to how to proceed, or if you want to check your answer against what the lab's authors came up with. There are often variations of how the APIs can be put together, so don't worry if you don't come up with exactly the same solutions we did.

The other file in exercises.answers is ExercisesOrig.java. This is a copy of the original state of the Exercises.java file. You can copy stuff from here if you've accidentally messed up your copy of Exercises.java and you want to restore it.

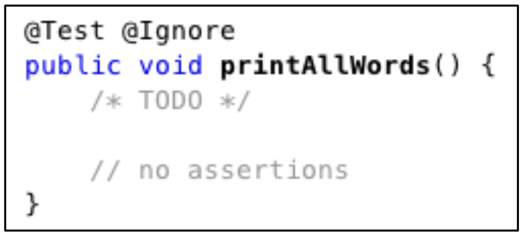
2.3. API Hints

In the API specification (javadoc) most of new APIs are in the java.util.stream package. The Stream interface itself has most of the interesting methods.

To get a stream from a collection, use Collection.stream() or Collection.parallelStream(). Other useful ways to get streams are IntStream.range(), Stream.of(), Arrays.stream(), BufferedReader.lines(), and CharSequence.chars(), and Pattern.splitAsStream().

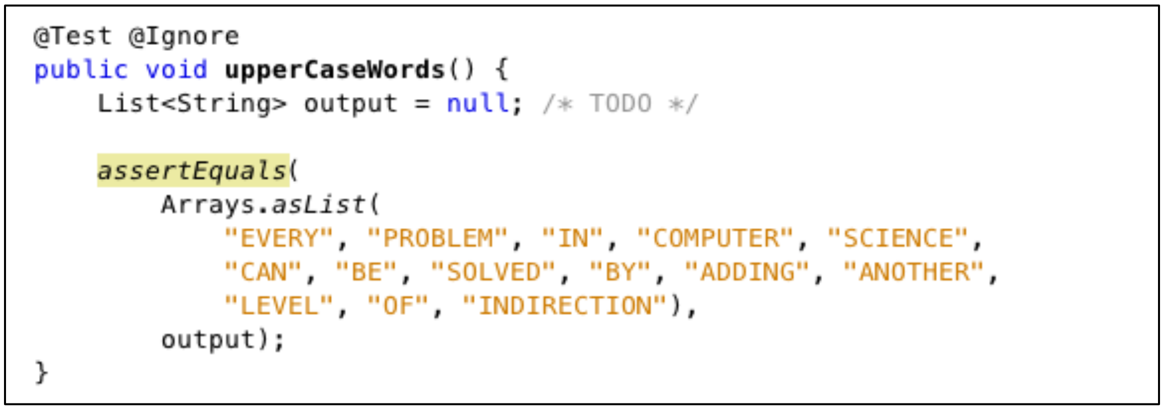
2.4. Exercises

Exercise 1: Print out all the words in wordList, which is a static List<String> defined at the bottom of this file.



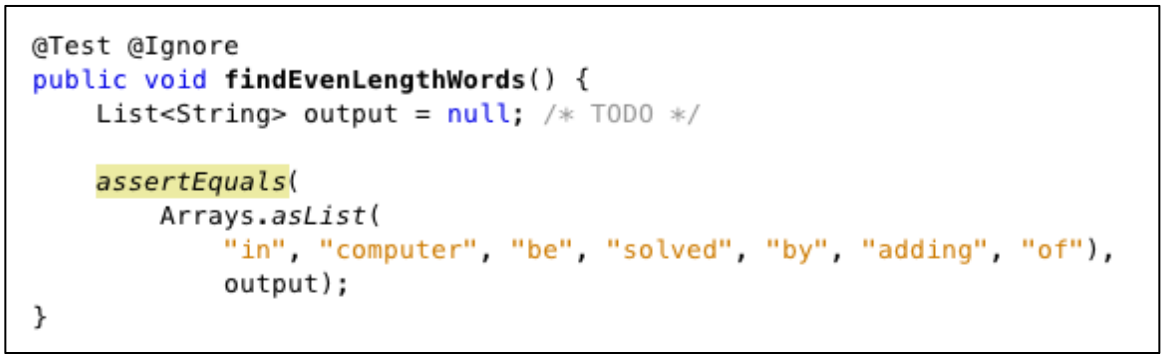
```java
@Test @Ignore
public void printAllWords() {
    /* TODO */

    // no assertions
}
```

Exercise 2: Convert all words in wordList to upper case, and gather the result into an output list.



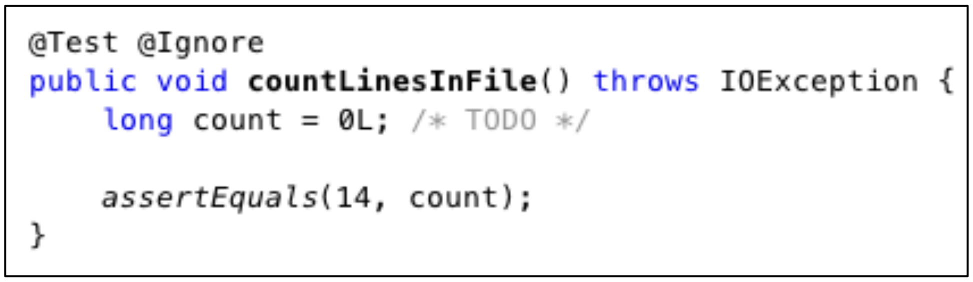
```java
@Test @Ignore
public void upperCaseWords() {
    List<String> output = null; /* TODO */

    assertEquals(
        Arrays.asList(
            "EVERY", "PROBLEM", "IN", "COMPUTER", "SCIENCE",
            "CAN", "BE", "SOLVED", "BY", "ADDING", "ANOTHER",
            "LEVEL", "OF", "INDIRECTION"),
        output);
}
```

Exercise 3: Find all the words in wordList that have even length and put them into an output list.



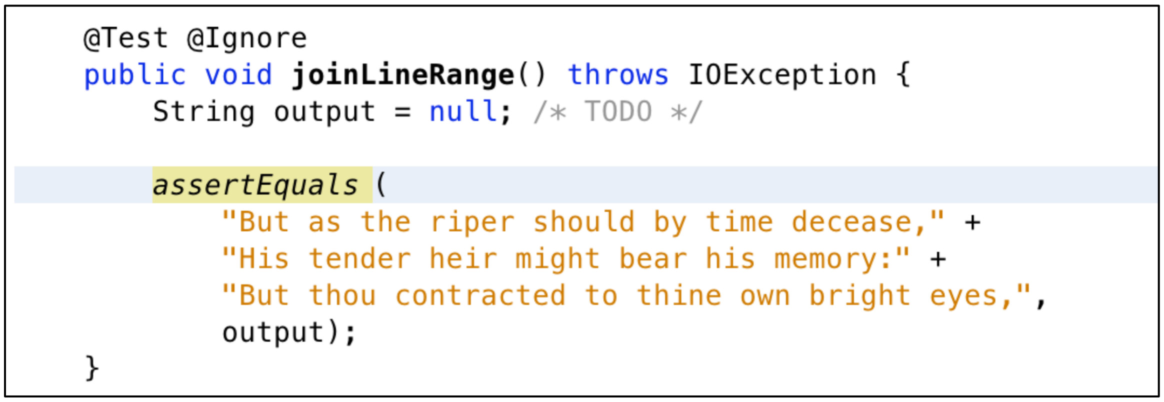
```java
@Test @Ignore
public void findEvenLengthWords() {
    List<String> output = null; /* TODO */

    assertEquals(
        Arrays.asList(
            "in", "computer", "be", "solved", "by", "adding", "of"),
            output);
}
```

Exercise 4: Count the number of lines in a file. The field *reader* is a BufferedReader which will be opened for you on the text file. See the JUnit @Before and @After methods at the bottom of this file. The text file is "SonnetI.txt" (Shakespeare's first sonnet) which is located at the root of this NetBeans project.



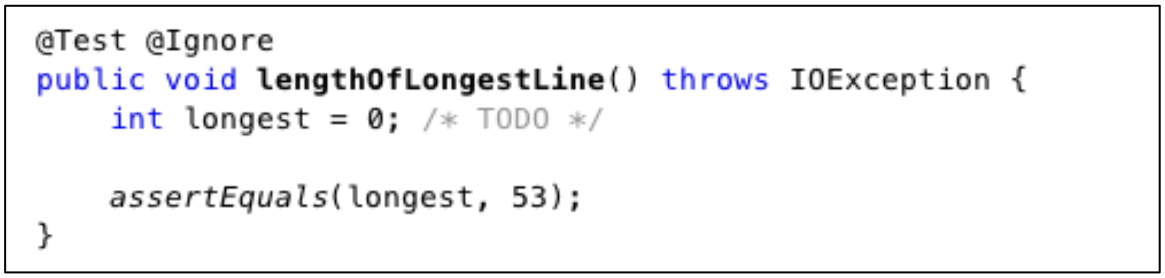
```java
@Test @Ignore
public void countLinesInFile() throws IOException {
    long count = 0L; /* TODO */

    assertEquals(14, count);
}
```

Exercise 5: Join lines 3-5 from the text file into a single string.



```
@Test @Ignore
public void joinLineRange() throws IOException {
    String output = null; /* TODO */

    assertEquals (
        "But as the riper should by time decease," +
        "His tender heir might bear his memory:" +
        "But thou contracted to thine own bright eyes,",
        output);
}
```

Exercise 6: Find the length of the longest line in the file.



```
@Test @Ignore
public void lengthOfLongestLine() throws IOException {
    int longest = 0; /* TODO */

    assertEquals(longest, 53);
}
```

Exercise 7: Collect all the words from the text file into a list.

Hint: use String.split(REGEXP) to split a string into words. Splitting this way results in "words" that are the empty string, which should be discarded. REGEXP is defined at the bottom of this file.

```
@Test @Ignore
public void listOfAllWords() throws IOException {
    List<String> output = null; /* TODO */

    assertEquals(
        Arrays.asList(
            "From", "fairest", "creatures", "we", "desire", "increase",
            "That", "thereby", "beauty", "s", "rose", "might", "never", "die",
            "But", "as", "the", "riper", "should", "by", "time", "decease",
            "His", "tender", "heir", "might", "bear", "his", "memory", "But",
            "thou", "contracted", "to", "thine", "own", "bright", "eyes",
            "Feed", "st", "thy", "light", "s", "flame", "with", "self",
            "substantial", "fuel", "Making", "a", "famine", "where",
            "abundance", "lies", "Thy", "self", "thy", "foe", "to", "thy",
            "sweet", "self", "too", "cruel", "Thou", "that", "art", "now",
            "the", "world", "s", "fresh", "ornament", "And", "only", "herald",
            "to", "the", "gaudy", "spring", "Within", "thine", "own", "bud",
            "buriest", "thy", "content", "And", "tender", "churl", "mak",
            "st", "waste", "in", "niggarding", "Pity", "the", "world", "or",
            "else", "this", "glutton", "be", "To", "eat", "the", "world", "s",
            "due", "by", "the", "grave", "and", "thee"),
        output);
}
```

Exercise 8: Create a list containing the words, lowercased, in alphabetical order.

```
@Test @Ignore
public void sortedLowerCase() throws IOException {
    List<String> output = null; /* TODO */

    assertEquals(
        Arrays.asList(
            "a", "abundance", "and", "and", "and", "art", "as", "be",
            "bear", "beauty", "bright", "bud", "buriest", "but", "but",
            "by", "by", "churl", "content", "contracted", "creatures",
            "cruel", "decease", "desire", "die", "due", "eat", "else",
            "eyes", "fairest", "famine", "feed", "flame", "foe", "fresh",
            "from", "fuel", "gaudy", "glutton", "grave", "heir", "herald",
            "his", "his", "in", "increase", "lies", "light", "mak",
            "making", "memory", "might", "might", "never", "niggarding",
            "now", "only", "or", "ornament", "own", "own", "pity",
            "riper", "rose", "s", "s", "s", "s", "self", "self", "self",
            "should", "spring", "st", "st", "substantial", "sweet",
            "tender", "tender", "that", "that", "the", "the", "the",
            "the", "the", "the", "thee", "thereby", "thine", "thine",
            "this", "thou", "thou", "thy", "thy", "thy", "thy", "thy",
            "time", "to", "to", "to", "to", "too", "waste", "we", "where",
            "with", "within", "world", "world", "world"),
        output);
}
```

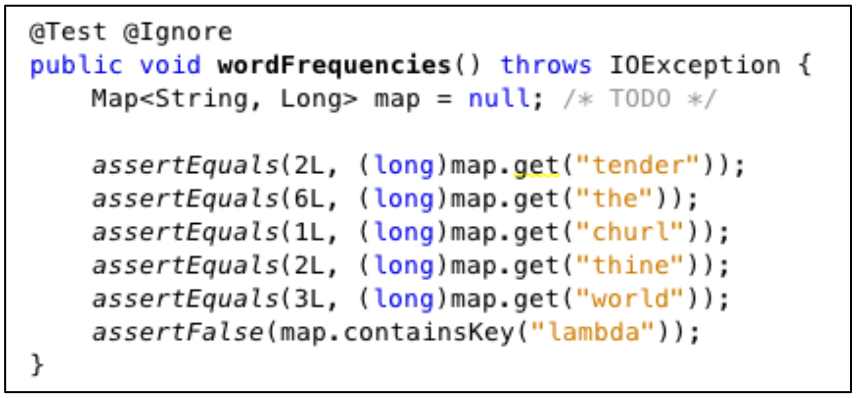
Exercise 9: Sort unique, lower-cased words by length, then alphabetically within length, and place the result into an output list.

```java
@Test @Ignore
public void sortedLowerCaseDistinctByLengthThenAlphabetically()
        throws IOException {
    List<String> output = null; /* TODO */

    assertEquals(
        Arrays.asList(
            "a", "s", "as", "be", "by", "in", "or", "st", "to", "we",
            "and", "art", "bud", "but", "die", "due", "eat", "foe", "his",
            "mak", "now", "own", "the", "thy", "too", "bear", "else",
            "eyes", "feed", "from", "fuel", "heir", "lies", "only",
            "pity", "rose", "self", "that", "thee", "this", "thou",
            "time", "with", "churl", "cruel", "flame", "fresh", "gaudy",
            "grave", "light", "might", "never", "riper", "sweet", "thine",
            "waste", "where", "world", "beauty", "bright", "desire",
            "famine", "herald", "making", "memory", "should", "spring",
            "tender", "within", "buriest", "content", "decease",
            "fairest", "glutton", "thereby", "increase", "ornament",
            "abundance", "creatures", "contracted", "niggarding",
            "substantial"),
        output);
}
```

Exercise 10: Categorize the words into a map, where the map's key is the length of each word, and the value corresponding to a key is a list of words of that length. Don't bother with uniqueness or lower-casing the words.

```java
@Test @Ignore
public void mapLengthToWordList() throws IOException {
    Map<Integer, List<String>> map = null; /* TODO */

    assertEquals(6, map.get(7).size());
    assertEquals(Arrays.asList("increase", "ornament"), map.get(8));
    assertEquals(Arrays.asList("creatures", "abundance"), map.get(9));
    assertEquals(Arrays.asList("contracted", "niggarding"), map.get(10));
    assertEquals(Arrays.asList("substantial"), map.get(11));
    assertFalse(map.containsKey(12));
}
```

Exercise 11: Gather the words into a map, accumulating a count of the number of occurrences of each word. Don't worry about upper case and lower case. Extra challenge: implement two solutions, one that uses groupingBy() and the other that uses toMap().

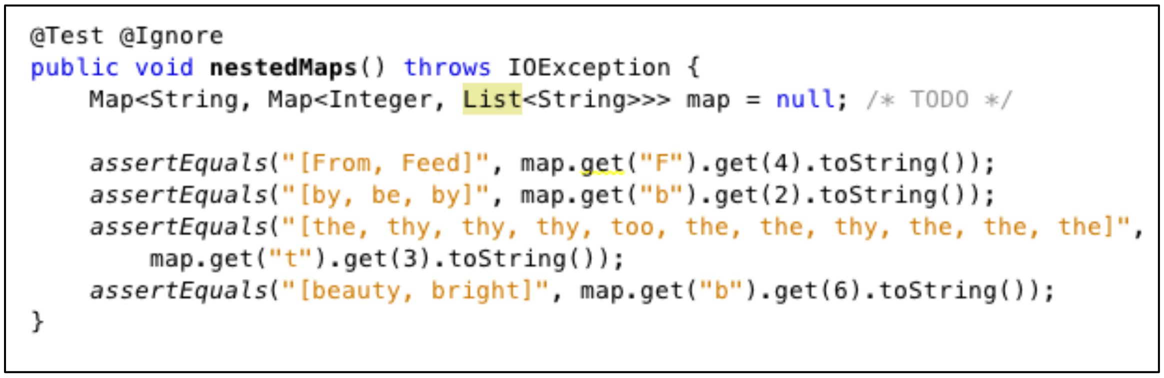


```
@Test @Ignore
public void wordFrequencies() throws IOException {
    Map<String, Long> map = null; /* TODO */

    assertEquals(2L, (long)map.get("tender"));
    assertEquals(6L, (long)map.get("the"));
    assertEquals(1L, (long)map.get("churl"));
    assertEquals(2L, (long)map.get("thine"));
    assertEquals(3L, (long)map.get("world"));
    assertFalse(map.containsKey("lambda"));
}
```

Exercise 12: Create nested maps, where the outer map is a map from the first letter of the word to an inner map. (Use a string of length one as the key.) The inner map, in turn, is a mapping from the length of the word to a list of words with that length. Don't bother with any lowercasing or uniquifying of the words.

For example, given the words "foo bar baz bazz" the string representation of the result would be:

{f={3=[foo]}, b={3=[bar, baz], 4=[bazz]}}.



```
@Test @Ignore
public void nestedMaps() throws IOException {
    Map<String, Map<Integer, List<String>>> map = null; /* TODO */

    assertEquals("[From, Feed]", map.get("F").get(4).toString());
    assertEquals("[by, be, by]", map.get("b").get(2).toString());
    assertEquals("[the, thy, thy, thy, too, the, the, thy, the, the, the]",
        map.get("t").get(3).toString());
    assertEquals("[beauty, bright]", map.get("b").get(6).toString());
}
```

Test Infrastructure

```
static List<String> wordList = Arrays.asList(
    "every", "problem", "in", "computer", "science",
    "can", "be", "solved", "by", "adding", "another",
    "level", "of", "indirection");
        // Butler Lampson

static final String REGEXP = "\\W+"; // for splitting into words

private BufferedReader reader;

@Before
public void setUpBufferedReader() throws IOException {
    reader = Files.newBufferedReader(
            Paths.get("SonnetI.txt"), StandardCharsets.UTF_8);
}

@After
public void closeBufferedReader() throws IOException {
    reader.close();
}
```

3. Web Links

Java Tutorial on Lambda Expressions:

http://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html

Lambda FAQ:

http://www.lambdafaq.org/

State of the Lambda (language design background and rationale):

http://cr.openjdk.java.net/~briangoetz/lambda/lambda-state-final.html

State of the Lambda Libraries (library design):

http://cr.openjdk.java.net/~briangoetz/lambda/lambda-libraries-final.html

Oracle Learning Library webcast (Michael Williams):

http://www.youtube.com/watch?v=giyKPnrpcjA

Jump-Starting Lambda Programming presentation (Stuart Marks, JavaOne 2012):

http://www.youtube.com/watch?v=bzO5GSujdqI

Jump-Starting Lambda Programming slides only (Stuart Marks, JavaOne 2013):
http://stuartmarks.files.wordpress.com/2013/09/tut3877_marks-jumpstartinglambda-v6.pdf