

Programming Assignment 1: Java Programming Essentials

Sayed Hadi Hashemi
Last update: August 29, 2015

1 Overview

Welcome to the Java Programming Assignment. This assignment evaluates the Java programming skill you will need to work on the upcoming assignment.

2 Requirements


This assignment is designed for and will be graded based on **JDK 7**.




If you are new to Java programming language, take a look at:
<https://docs.oracle.com/javase/tutorial/>

3 Procedure

Step 1: Find your User ID in the Assignment page. You'll need this ID to submit your solution.

**Cloud Computing Applications**
by Roy H. Campbell, Reza Farivar, PhD



Announcements

Syllabus

How to Pass the Class

COURSE MODULES

Course Orientation
(Mandatory; Start Here) ←

Assignments

User ID	35217034
Submission Login	hashemi3@illinois.edu
Submission Password	MGBbKn9sjh <input type="button" value="Generate New Password"/>

> Week 1

▼ Week 2

MapReduce Assignment

Step 2: Download the project files from the following link:

<https://github.com/xldrx/cloudapp-mp1>

Step 3: Edit the following template file. All you need to edit is the part marked with **TODO**. You will find more information on what to do in the next section.

MP1.java

Don't change the filename, class name, or the main function.

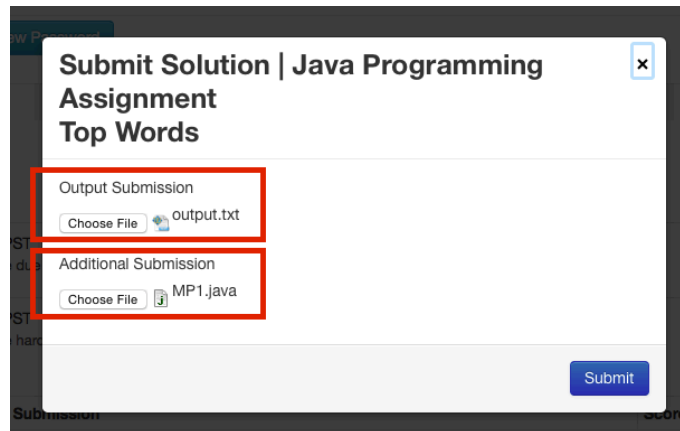
Step 4: Compile the file and run it on the provided input. Use **your own Coursera User ID** as the first argument. The following is an example of a command to run the application. The command line might need some modifications for your platform:

```
# java jar MP1.jar 35217034
```

Make sure the first argument is an exact match with your Coursera User ID; otherwise, your submission might not be graded.

Step 5: Save the output in a text file.

Step 6: Submit the file to the Coursera Website.



Submit Solution | Java Programming Assignment

Top Words

Output Submission

Choose File output.txt

Additional Submission

Choose File MP1.java

Submit

You have to submit both the results and java source code. Make sure that the result is generated using your own **User ID**. The java source code should also be compilable using Oracle JDK 7.

Exercise: Selective Word Count

In this exercise you are to implement an application to find the top **20** words used in Wikipedia titles (provided). To make the implementation easier, we have provided a boilerplate for this exercise in the following file:

MP1.java

All you need to do is to make necessary changes in the file.

Your application takes a huge list of Wikipedia titles (one in each line) as an input. You need to make some preprocessing on the input, and then return the top **20** words that appear the most in a selection of titles. One possible procedure is the following:

1. Divide each sentence into a list of words using delimiters provided in the “**delimiters**” variable.



One possible approach is to use StringTokenizer. For more information see:
<http://docs.oracle.com/javase/7/docs/api/java/util/StringTokenizer.html>

2. Make all the tokens lowercase and remove any trailing and leading spaces.



More information about string operations can be found in:
<http://docs.oracle.com/javase/7/docs/api/java/lang/String.html>

3. Ignore all common words provided in the “**stopWordsArray**” variable.



One possible approach is to use lists. For more information see:
[http://docs.oracle.com/javase/7/docs/api/java/util/List.html#contains\(java.lang.Object\)](http://docs.oracle.com/javase/7/docs/api/java/util/List.html#contains(java.lang.Object))

4. Keep track of word frequencies. To make the application more interesting, you have to process **only the titles with certain indexes**. These indexes are accessible using the “**getIndexes**” method, which returns an Integer Array with 0-based indexes to the input file. It is possible to have an index appear several times. In this case, just process the index multiple times.



One possible approach is to use Maps in java. For more information see:
<http://docs.oracle.com/javase/7/docs/api/java/util/Map.html>

- Sort the list by frequency in a descending order. If two words have the same number count, use the lexicography. For example, the following is a sorted list:

```
{{(Orange, 3), (Apple, 2), (Banana, 2)}}
```



On possible approach is to use sort function in Java. For more information, see:
[http://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#sort\(java.util.List,%20java.util.Comparator\)](http://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#sort(java.util.List,%20java.util.Comparator))

- Return the top 20 items from the sorted list as a String Array.

Here is the output of this application if “0” is used for user ID:

```
list
de
state
school
disambiguation
county
new
john
album
c
river
station
united
highway
national
saint
william
route
f
film
```

Here is the output of this application if “1” is used for user ID:

```
list
de
state
school
disambiguation
county
new
john
river
route
film
album
c
high
united
william
st
national
football
saint
```

Remember, in order to submit the application **your own Coursera user ID** should be used as the user ID. Changing the user ID may change the output.