Assignment 2: System Dependencies

version 1.0

Mark Dras

May 9, 2017

(This is a release version, but a changelog will still list changes in Section 4.)

1 Introduction

Components of computer systems often have dependencies—other components that must be installed before they will function properly.[1] These dependencies are frequently shared by multiple components. For example, both the TELNET client program and the FTP client program require that the TCP/IP networking software be installed before they can operate. If you install TCP/IP and the TELNET client program, and later decide to add the FTP client program, you do not need to reinstall TCP/IP.

For some components it would not be a problem if the components on which they depended were reinstalled; it would just waste some resources. But for others, like TCP/IP, some component configuration may be destroyed if the component was reinstalled.

It is useful to be able to remove components that are no longer needed. When this is done, components that only support the removed component may also be removed, freeing up disk space, memory, and other resources. But a supporting component, not explicitly installed, may be removed only if all components which depend on it are also removed. For example, removing the FTP client program and TCP/IP would mean the TELNET client program, which was not removed, would no longer operate. Likewise, removing TCP/IP by itself would cause the failure of both the TELNET and the FTP client programs. Also if we installed TCP/IP to support our own development, then installed the TELNET client (which depends on TCP/IP) and then still later removed the TELNET client, we would not want TCP/IP to be removed.

We want a program to automate the process of adding and removing components. To do this we will maintain a record of installed components and component dependencies. A component can be installed explicitly in response to a command (unless it is already installed), or implicitly if it is needed for some other component being installed. Likewise, a component, not explicitly installed, can be explicitly removed in response to a command (if it is not needed to support other components) or implicitly removed if it is no longer needed to support another component.

[1] The overall framework of the assignment, and the task in the Pass section, come from a past ACM Programming Contest problem. Many of these past problems are now hosted on automated judging sites, including this one. You might be interested in using the automated judging as part of developing a solution for this assignment. The original version of this particular problem is at `https://www.bnuoj.com/v3/problem_show.php?pid=17645`

Installing an already implicitly-installed component won't make that component become explicitly installed.

Input The input will contain a sequence of commands (as described below), each on a separate line containing no more than eighty characters. Item names are case sensitive, and each is no longer than ten characters. The command names (`DEPEND`, `INSTALL`, `REMOVE` and `LIST`) always appear in uppercase starting in column one, and item names are separated from the command name and each other by one or more spaces. All appropriate `DEPEND` commands will appear before the occurrence of any `INSTALL` command that uses them; there will be at most one dependency list per item. The end of the input is marked by a line containing only the word `END`. There will be at most 1000 lines in the input.

Command Syntax	*Interpretation / Response*
`DEPEND item1 item2 [item3 ...]`	**item1** depends on **item2** (and **item3** ...)
`INSTALL item1`	install **item1** and those on which it depends
`REMOVE item1`	remove **item1**, and those on which it depends, if possible
`LIST`	list the names of all currently-installed components

Output For each test case, the output must follow the description given in each level and illustrated by the examples. **It must match the corresponding output file exactly**, as the JUnit tests will use a string match to compare your program output with the expected output.

Code There is a zipfile provided with the file `DepInstall.java` and some JUnit tests in `AllTest.java`. `DepInstall.java` contains some code for reading the list of commands from the input file into a vector of strings, and code for reading the expected output. There are also method stubs for code you will have to write, indicated by the comment `// TODO`.

2 Your Tasks

For your tasks, you'll be adding attributes and methods to the class given in the code bundle accompanying these specs. Where it's given, **you should use exactly the method stub provided** for implementing your tasks. Don't change the names or the parameters. You can add more functions if you like.

The input data will be available the form of files with a `.in` suffix, and the corresponding output in files with a `.out` suffix.

2.1 Pass Level

To achieve at least a Pass (50–64%) for the assignment, you should be able to pass all the unit tests indicated as Pass level.

In the Pass level, there will be no circular dependencies.

Output Echo each line of input. Follow each echoed `INSTALL` or `REMOVE` line with the actions taken in response, making certain that the actions are given in the proper order. Also identify exceptional conditions (see Sample Output, below, for examples of all cases). For the `LIST` command, display the names of the currently installed components in the installation order. No output, except the echo, is produced for a `DEPEND` command or the line containing `END`. (Note that the actions all begin with 3 spaces.)

Required Code You should complete a method with the following definition to produce the output as specified, running the first `N` commands stored in the parameter `commands`.

```
public void runNCommands (Vector<String> commands, Integer N) {
  // PRE: commands contains set of commands read in by readCommandsFromFile()
  // POST: executed min(N, all) commands

  // TODO
}
```

Sample Input

```
DEPEND TELNET TCPIP NETCARD
DEPEND TCPIP NETCARD
DEPEND DNS TCPIP NETCARD
DEPEND BROWSER TCPIP HTML
INSTALL NETCARD
INSTALL TELNET
INSTALL foo
REMOVE NETCARD
INSTALL BROWSER
INSTALL DNS
LIST
REMOVE TELNET
REMOVE NETCARD
REMOVE DNS
REMOVE NETCARD
INSTALL NETCARD
REMOVE TCPIP
REMOVE BROWSER
REMOVE TCPIP
END
```

Sample Output

```
DEPEND TELNET TCPIP NETCARD
DEPEND TCPIP NETCARD
DEPEND DNS TCPIP NETCARD
DEPEND BROWSER TCPIP HTML
INSTALL NETCARD
   Installing NETCARD
```

```
INSTALL TELNET
   Installing TCPIP
   Installing TELNET
INSTALL foo
   Installing foo
REMOVE NETCARD
   NETCARD is still needed
INSTALL BROWSER
   Installing HTML
   Installing BROWSER
INSTALL DNS
   Installing DNS
LIST
   NETCARD
   TCPIP
   TELNET
   foo
   HTML
   BROWSER
   DNS
REMOVE TELNET
   Removing TELNET
REMOVE NETCARD
   NETCARD is still needed
REMOVE DNS
   Removing DNS
REMOVE NETCARD
   NETCARD is still needed
INSTALL NETCARD
   NETCARD is already installed
REMOVE TCPIP
   TCPIP is still needed
REMOVE BROWSER
   Removing BROWSER
   Removing HTML
   Removing TCPIP
REMOVE TCPIP
   TCPIP is not installed
END
```

2.2 Credit Level

To achieve at least a Credit (65–74%) for the assignment, you should do the following. You should also have completed all the Pass-level tests.

In the Credit level, there will be circular dependencies. Once a circular dependency is detected, there should be no further actions carried out.

Output The output is as in the Pass level, except for the following. Immediately after a circular dependency has been created through a `DEPEND` command, a warning should be given. Subsequent `DEPEND` commands should also be followed by the warning. Other remaining non-`DEPEND` commands after that point, however, should be echoed, but with no actions taken in response (so the echoed commands should be followed by nothing).

Required Code You should complete a method with the following definition to produce the output as specified.

```
public void runNCommandswCheck (Vector<String> commands, Integer N) {
  // PRE: commands contains set of commands read in by readCommandsFromFile()
  // POST: executed min(N, all) commands, checking for cycles

  // TODO
}
```

Sample Input

```
DEPEND TELNET TCPIP NETCARD
DEPEND TCPIP NETCARD
DEPEND DNS TCPIP NETCARD
DEPEND BROWSER TCPIP HTML
INSTALL foo
DEPEND NETCARD BROWSER
DEPEND BROWSER2 TCPIP HTML
INSTALL NETCARD
INSTALL TELNET
REMOVE NETCARD
INSTALL BROWSER
INSTALL DNS
LIST
REMOVE TELNET
REMOVE NETCARD
REMOVE DNS
REMOVE NETCARD
INSTALL NETCARD
REMOVE TCPIP
REMOVE BROWSER
REMOVE TCPIP
END
```

Sample Output

```
DEPEND TELNET TCPIP NETCARD
DEPEND TCPIP NETCARD
DEPEND DNS TCPIP NETCARD
DEPEND BROWSER TCPIP HTML
INSTALL foo
```

```
   Installing foo
DEPEND NETCARD BROWSER
   Found cycle in dependencies
DEPEND BROWSER2 TCPIP HTML
   Found cycle in dependencies
INSTALL NETCARD
INSTALL TELNET
REMOVE NETCARD
INSTALL BROWSER
INSTALL DNS
LIST
REMOVE TELNET
REMOVE NETCARD
REMOVE DNS
REMOVE NETCARD
INSTALL NETCARD
REMOVE TCPIP
REMOVE BROWSER
REMOVE TCPIP
END
```

2.3 (High) Distinction Level

To achieve at least a Distinction (75–100%) for the assignment, you should do the following. You should also have completed all the Credit-level tasks.

Output The output is as in the Credit level, except for the following. As in the Credit level, immediately after a circular dependency has been created through a `DEPEND` command, a warning should be given. In the Distinction level, there should be a second line recommending a dependency to delete. In the first instance, this should be the most recently encountered dependency statement from the *largest* cycle. Subsequent `DEPEND` commands should also be followed by the warning; however, the suggested dependency statement might change (because the cycle size has changed). As before, other remaining non-`DEPEND` commands after that point, however, should be echoed, but with no actions taken in response (so the echoed commands should be followed by nothing).

This should be repeated, but recommending the most recently encountered dependency statement from the *smallest* cycle.

Required Code You should complete methods with the following definitions to produce the output as specified.

```
public void runNCommandswCheckRecLarge (Vector<String> commands, Integer N) {
  // PRE: commands contains set of commands read in by readCommandsFromFile()
  // POST: executed min(N, all) commands, checking for cycles and
  //       recommending fix by removing largest cycle

  // TODO
}
```

```
public void runNCommandswCheckRecSmall (Vector<String> commands, Integer N) {
  // PRE: commands contains set of commands read in by readCommandsFromFile()
  // POST: executed min(N, all) commands, checking for cycles and
  //       recommending fix by removing smallest cycle

  // TODO
}
```

Sample Input #1: Largest Cycle

```
DEPEND TELNET TCPIP NETCARD
DEPEND TCPIP NETCARD
DEPEND DNS TCPIP NETCARD
DEPEND BROWSER TCPIP HTML
INSTALL foo
DEPEND BROWSER2 BROWSER3 TCPIP HTML
DEPEND BROWSER3 BROWSER2
DEPEND NETCARD BROWSER
INSTALL NETCARD
INSTALL TELNET
REMOVE NETCARD
INSTALL BROWSER
INSTALL DNS
LIST
REMOVE TELNET
REMOVE NETCARD
REMOVE DNS
REMOVE NETCARD
INSTALL NETCARD
REMOVE TCPIP
REMOVE BROWSER
REMOVE TCPIP
END
```

Sample Output #1: Largest Cycle

```
DEPEND TELNET TCPIP NETCARD
DEPEND TCPIP NETCARD
DEPEND DNS TCPIP NETCARD
DEPEND BROWSER TCPIP HTML
INSTALL foo
   Installing foo
DEPEND BROWSER2 BROWSER3 TCPIP HTML
DEPEND BROWSER3 BROWSER2
   Found cycle in dependencies
   Suggest removing DEPEND BROWSER3 BROWSER2
DEPEND NETCARD BROWSER
   Found cycle in dependencies
```

```
   Suggest removing DEPEND NETCARD BROWSER
INSTALL NETCARD
INSTALL TELNET
REMOVE NETCARD
INSTALL BROWSER
INSTALL DNS
LIST
REMOVE TELNET
REMOVE NETCARD
REMOVE DNS
REMOVE NETCARD
INSTALL NETCARD
REMOVE TCPIP
REMOVE BROWSER
REMOVE TCPIP
END
```

Sample Input #2: Largest Cycle

```
DEPEND TELNET TCPIP NETCARD
DEPEND TCPIP NETCARD
DEPEND DNS TCPIP NETCARD
DEPEND BROWSER TCPIP HTML
INSTALL foo
DEPEND NETCARD BROWSER
DEPEND BROWSER2 BROWSER3 TCPIP HTML
DEPEND BROWSER3 BROWSER2
```

Sample Output #2: Largest Cycle

```
DEPEND TELNET TCPIP NETCARD
DEPEND TCPIP NETCARD
DEPEND DNS TCPIP NETCARD
DEPEND BROWSER TCPIP HTML
INSTALL foo
   Installing foo
DEPEND NETCARD BROWSER
   Found cycle in dependencies
   Suggest removing DEPEND NETCARD BROWSER
DEPEND BROWSER2 BROWSER3 TCPIP HTML
   Found cycle in dependencies
   Suggest removing DEPEND NETCARD BROWSER
DEPEND BROWSER3 BROWSER2
   Found cycle in dependencies
   Suggest removing DEPEND NETCARD BROWSER
```

Sample Input #3: Smallest Cycle

```
DEPEND TELNET TCPIP NETCARD
DEPEND TCPIP NETCARD
DEPEND DNS TCPIP NETCARD
DEPEND BROWSER TCPIP HTML
INSTALL foo
DEPEND BROWSER2 BROWSER3 TCPIP HTML
DEPEND BROWSER3 BROWSER2
DEPEND NETCARD BROWSER
```

Sample Output #3: Smallest Cycle

```
DEPEND TELNET TCPIP NETCARD
DEPEND TCPIP NETCARD
DEPEND DNS TCPIP NETCARD
DEPEND BROWSER TCPIP HTML
INSTALL foo
   Installing foo
DEPEND BROWSER2 BROWSER3 TCPIP HTML
DEPEND BROWSER3 BROWSER2
   Found cycle in dependencies
   Suggest removing DEPEND BROWSER3 BROWSER2
DEPEND NETCARD BROWSER
   Found cycle in dependencies
   Suggest removing DEPEND BROWSER3 BROWSER2
```

Sample Input #4: Smallest Cycle

```
DEPEND TELNET TCPIP NETCARD
DEPEND TCPIP NETCARD
DEPEND DNS TCPIP NETCARD
DEPEND BROWSER TCPIP HTML
INSTALL foo
DEPEND NETCARD BROWSER
DEPEND BROWSER2 BROWSER3 TCPIP HTML
DEPEND BROWSER3 BROWSER2
```

Sample Output #4: Smallest Cycle

```
DEPEND TELNET TCPIP NETCARD
DEPEND TCPIP NETCARD
DEPEND DNS TCPIP NETCARD
DEPEND BROWSER TCPIP HTML
INSTALL foo
   Installing foo
DEPEND NETCARD BROWSER
   Found cycle in dependencies
   Suggest removing DEPEND NETCARD BROWSER
```

```
DEPEND BROWSER2 BROWSER3 TCPIP HTML
   Found cycle in dependencies
   Suggest removing DEPEND NETCARD BROWSER
DEPEND BROWSER3 BROWSER2
   Found cycle in dependencies
   Suggest removing DEPEND BROWSER3 BROWSER2
```

3 What To Hand In

In the submission page on iLearn for this assignment you must include the following:

Submit DepInstall.java as defined in the package from the original assignment code bundle.

Your file must leave unchanged the specification of already implemented functions, and include your implementations of your selection of method stubs outlined above.

Do not change the names of the method stubs because the auto-tester assumes the names given. Do not change the package statement. You may however include additional auxiliary methods if you need them.

Please note that we are unable to check individual submissions and so it is very important to abide by the above submission instructions.

4 Changelog

- **9/5/16**: Assignment draft released.