COMP2010 Assignment 2:
How to Cast the Ultimate Fire Spell

ver 1.02

Mark Dras

May 14, 2024

1 Introduction

As many of you will know, there's a genre of fantasy-themed computer games where there's some overall quest or storyline to follow, and some side-quests along the way to flesh out the world and build up power for the grand finale.[1] A big part of those games is the magic system. In the magic system, you learn increasingly powerful spells, but you typically have to learn the lower-level spells before you can attempt the higher-level ones.[2] Sometimes you'll also want to unlearn some spells too: you get to the final Boss and discover that you actually needed `mordenkainens-undead-platoon-summon` rather than `divine-firestorm`.

This assignment is going to consider a simplified magic system for such a game. You will be provided with specifications of how the learning and unlearning work (§2). You will also be provided with a code framework, which defines classes and includes some pre-written code, as well as stubs of some incomplete methods (§3).

Assignment Aim Your assignment will be to implement this simplified magic system from §2 by completing the specified methods described in §3. You can use the accompanying sample JUnit tests as part of a check that you have done this correctly; the JUnit tests that will be used in the automarking will be similar (but e.g. including some unseen data).

2 The Spell Management System

The below sections start by explaining the specification types (§2.1) and the format of the specification data (§2.2), and then traces an example (§2.3) that covers all the possible circumstances under the basic (Pass-level) version of the problem.

2.1 Specs

In your hypothetical spell management system (SMS), you will be keeping track of which spells are currently learned. Which spells are prerequisites for which other spells will be given by the specification

`PREREQ ⟨spellname1⟩⟨spellname2⟩...⟨spellname`n`⟩`

(where the notation `⟨spellname⟩` means "fill in with some actual spell name"). In this `PREREQ` specification, there will be at least two spells: the focal one `⟨spellname1⟩`, and at least one (`⟨spellname2⟩`) that is a prerequisite for it. There will also be only one `PREREQ` specification for each `⟨spellname1⟩`.

A spell can be learned *explicitly* (unless it has already been learned) by the specification

`LEARN ⟨spellname⟩`

A spell can also be learned *implicitly* if you ask for some other spell to be learned, but its prerequisites haven't been; these prereq spells are added to the SMS automatically. (So if you specify explicitly learning `⟨spellname1⟩` from the `PREREQ` statement above, you'll also learn implicitly any of `⟨spellname2⟩...⟨spellname`n`⟩` that haven't been learned already, etc.) You can't learn a spell that's already been learned; you'll see how to handle this, like all cases of "what happens if ...", in §2.3.

[1] Think `https://baldursgate3.game/` or the classic `https://en.wikipedia.org/wiki/Dragon_Age:_Origins`.
[2] It's essentially like how you have to do COMP1010 before COMP2010, and COMP1000 before COMP1010.

A spell can be deleted explicitly from the SMS (or 'forgotten') via the specification

`FORGET ⟨spellname⟩`

It's possible to forget a spell only as long as it's not needed as a prereq by other spells that have been learned. A spell is also forgotten implicitly, as long as it was learned implicitly, when there are no other learned spells for which it is a prereq. (Note that specifying the learning of an already implicitly learned spell doesn't make it explicitly learned, which has implications for forgetting. It just stays implicitly learned. See §2.3 for how this is handled.)

There's also a way of enumerating which spells you've already learned, via the specification

`ENUM`

Finally, the end of a list of specifications is given by `END`.

2.2 Specification Format

The input will be a series of specifications, one per line given in a file. The output has the following properties:

- Each input line is echoed exactly as it appeared.
- Each echoed input line in the output is followed by zero or more responses, each of which starts with three space characters.
  - For `LEARN` specifications, the responses for spells that are prereqs have to come before the spells they are prereqs for.
  - For `FORGET` specifications, the responses for spells that are prereqs have to come *after* the spells they are prereqs for. Subject to this prereq ordering constraint, spells should be forgotten in the reverse order in which they were learned.
  - For `ENUM`, the spells should be listed in the order in which they were learned.
  - Nothing should appear after `END`.

The example below gives all of the formatting information necessary, including all the possible responses in the output.

2.3 Example

Here's an example of a sequence of specifications:

Sample Input #1

```
PREREQ fireball burning-cascade magic-missile
PREREQ burning-cascade magic-missile
PREREQ flaming-mantle burning-cascade magic-missile
PREREQ shining-sphere-of-protection burning-cascade sparks
LEARN magic-missile
LEARN fireball
LEARN minor-heal
FORGET magic-missile
LEARN shining-sphere-of-protection
LEARN flaming-mantle
ENUM
FORGET fireball
FORGET magic-missile
FORGET flaming-mantle
FORGET magic-missile
LEARN magic-missile
FORGET burning-cascade
FORGET shining-sphere-of-protection
FORGET burning-cascade
END
```

And here's what this sequence above should produce. (The annotations marked $\triangleright n$ for some integer n aren't part of what should be produced; these are just comments that will explain below particular aspects of why this is the output.)

Corresponding Sample Output #1

```
PREREQ fireball burning-cascade magic-missile
PREREQ burning-cascade magic-missile
PREREQ flaming-mantle burning-cascade magic-missile
PREREQ shining-sphere-of-protection burning-cascade sparks
LEARN magic-missile
   Learning magic-missile        ▷1
LEARN fireball
   Learning burning-cascade        ▷2
   Learning fireball
LEARN minor-heal
   Learning minor-heal
FORGET magic-missile
   magic-missile is still needed        ▷3
LEARN shining-sphere-of-protection
   Learning sparks
   Learning shining-sphere-of-protection
LEARN flaming-mantle
   Learning flaming-mantle        ▷4
ENUM
   magic-missile
   burning-cascade
   fireball
   minor-heal
   sparks
   shining-sphere-of-protection
   flaming-mantle
FORGET fireball
   Forgetting fireball
FORGET magic-missile
   magic-missile is still needed
FORGET flaming-mantle
   Forgetting flaming-mantle
FORGET magic-missile
   magic-missile is still needed
LEARN magic-missile
   magic-missile is already learned
FORGET burning-cascade
   burning-cascade is still needed
FORGET shining-sphere-of-protection
   Forgetting shining-sphere-of-protection
   Forgetting sparks        ▷5
   Forgetting burning-cascade
FORGET burning-cascade
   burning-cascade is not learned
END
```

Some remarks on why aspects of the output are what they are:

- ▷1 `magic-missile` is explicitly learned.
- ▷2 `burning-cascade` is *implicitly* learned in order to explicitly learn `fireball`, as per the second `PREREQ` statement.
- ▷3 `magic-missile` is still needed by `fireball`, as per first `PREREQ`.
- ▷4 Note that `burning-cascade` and `magic-missile` are already learned.
- ▷5 `sparks` is forgotten because it was only learned implicitly (when `shining-sphere-of-protection` was learned) and it's not a prereq for any other current spells. Ditto for `burning-cascade` immediately following. Also note that `sparks` can only be forgotten after `shining-sphere-of-protection` was forgotten. And `sparks` is forgotten before `burning-cascade` as it was learned later.

3 Assignment Code Structure

You will be working with a Java project that has 1 class called `BuildSpellbook`. For your tasks, you'll be adding attributes and methods to existing classes given in the code bundle accompanying these specs; this is indicated by the comment `// TODO`. Where it's given, **you should use exactly the method stub provided** for implementing your tasks. **Don't change the names or the parameters or exception handling.** You can add more methods if you like.

There also some methods provided, in particular for reading from files. These files, with suffixes `.in` for input files and `.out` for output files, will be read in such that each line is returned as a single string, and the whole file is turned into a Vector of strings. There are separate ones for reading in input files and output files:

```
public Vector<String> readSpecsFromFile(String fInName) throws IOException {
  // PRE: -
  // POST: returns lines from input file as vector of string

  [CODE PROVIDED]
}

public Vector<String> readSolnFromFile(String fInName, Integer N) throws IOException {
  // PRE: -
  // POST: returns (up to) N lines from input file as a vector of N strings;
  //       only the specification lines are counted in this N, not responses

  [CODE PROVIDED]
}
```

There is also a method for comparing the output of your code with the model output file:

```
public Boolean compareExecWSoln (Vector<String> execd, Vector<String> soln) {
  // PRE: -
  // POST: Returns True if execd and soln string-match exactly, False otherwise

  [CODE PROVIDED]
}
```

A Java file of sample JUnit tests, `SampleTests`, is also provided; it uses the sample data files provided.

3.1 Pass Level

To achieve at least a Pass ($\geq 50\%$) for the assignment, you should do all of the following tasks.

T1 You should complete a method with the following definition to produce the output as specified, executing the first `N` specifications stored in the parameter `specs`.

```
public Vector<String> execNSpecs (Vector<String> specs, Integer N)
  // PRE: specs contains set of specifications read in by readSpecsFromFile()
  // POST: executed min(N, all) specifications,
  //       returning required output, one line per string in vector

  // TODO
}
```

(Marks for assessment will include multiple JUnit tests that capture different levels of complexity of specification and response; for example, in the sample JUnit tests, there are some that assess just the first 5 lines or first 9 lines of this sample input file.)

3.2 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 prerequisites suggested, which are problematic. Once a circular prerequisite is detected, there should be no further actions carried out. That is, after a circular prereq, the system no longer functions and commands are ignored.

The output is as in §2.3, except for the following. Immediately after a circular prereq has been created through a PREREQ specification, a warning should be given. Subsequent PREREQ specifications should also be followed by the warning. Other remaining non-PREREQ specifications after that point, however, should be echoed, but with no actions taken in response (so the echoed specifications should be followed by nothing).

Sample Input #2

```
PREREQ fireball burning_cascade magic_missile
PREREQ burning_cascade magic_missile
PREREQ flaming_mantle burning_cascade magic_missile
PREREQ shining_sphere_of_protection burning_cascade sparks
LEARN minor_heal
PREREQ magic_missile shining_sphere_of_protection
LEARN magic_missile
LEARN fireball
FORGET magic_missile
LEARN shining_sphere_of_protection
LEARN flaming_mantle
ENUM
FORGET fireball
FORGET magic_missile
FORGET flaming_mantle
FORGET magic_missile
LEARN magic_missile
FORGET burning_cascade
FORGET shining_sphere_of_protection
FORGET burning_cascade
END
```

Corresponding Sample Output #2

```
PREREQ fireball burning_cascade magic_missile
PREREQ burning_cascade magic_missile
PREREQ flaming_mantle burning_cascade magic_missile
PREREQ shining_sphere_of_protection burning_cascade sparks
LEARN minor_heal
   Learning minor_heal
PREREQ magic_missile shining_sphere_of_protection
   Found cycle in prereqs
LEARN magic_missile
LEARN fireball
FORGET magic_missile
LEARN shining_sphere_of_protection
LEARN flaming_mantle
ENUM
FORGET fireball
FORGET magic_missile
FORGET flaming_mantle
FORGET magic_missile
LEARN magic_missile
FORGET burning_cascade
FORGET shining_sphere_of_protection
FORGET burning_cascade
END
```

T2 You should complete a method with the following definition to produce the output as specified.

```
public Vector<String> execNSpecswCheck (Vector<String> specs, Integer N) {
  // PRE: specs contains set of specifications read in by readSpecsFromFile()
  // POST: executed min(N, all) specifications, checking for cycles,
  //       returning required output, one line per string in vector
```

```
  // TODO
}
```

3.3 (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.

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 `PREREQ` specification, a warning should be given. In the Distinction level, there should be a second line recommending a prereq to delete. In the first instance, this should be the most recently encountered prereq specification from the *largest* cycle, defined by the number of elements in the cycle. Subsequent `PREREQ` specifications should also be followed by the warning; however, the suggested prereq specification might change (because the cycle size has changed). As before, other remaining non-`PREREQ` specifications after that point, however, should be echoed, but with no actions taken in response (so the echoed specifications should be followed by nothing).

T3 When there are multiple longest cycles, the recommendation for dependency deletion should be based on the *first encountered* longest cycle.

Sample Input #3: Largest Cycle

```
PREREQ fireball burning_cascade magic_missile
PREREQ burning_cascade magic_missile
PREREQ flaming_mantle burning_cascade magic_missile
PREREQ shining_sphere_of_protection burning_cascade sparks
LEARN minor_heal
PREREQ shining_sphere_of_protection2 shining_sphere_of_protection3 burning_cascade sparks
PREREQ shining_sphere_of_protection3 shining_sphere_of_protection2
PREREQ magic_missile shining_sphere_of_protection
LEARN magic_missile
LEARN fireball
FORGET magic_missile
LEARN shining_sphere_of_protection
LEARN flaming_mantle
ENUM
FORGET fireball
FORGET magic_missile
FORGET flaming_mantle
FORGET magic_missile
LEARN magic_missile
FORGET burning_cascade
FORGET shining_sphere_of_protection
FORGET burning_cascade
END
```

Corresponding Sample Output #3: Largest Cycle

```
PREREQ fireball burning_cascade magic_missile
PREREQ burning_cascade magic_missile
PREREQ flaming_mantle burning_cascade magic_missile
PREREQ shining_sphere_of_protection burning_cascade sparks
LEARN minor_heal
   Learning minor_heal
PREREQ shining_sphere_of_protection2 shining_sphere_of_protection3 burning_cascade sparks
PREREQ shining_sphere_of_protection3 shining_sphere_of_protection2
   Found cycle in prereqs
   Suggest forgetting PREREQ shining_sphere_of_protection3 shining_sphere_of_protection2
PREREQ magic_missile shining_sphere_of_protection
   Found cycle in prereqs
   Suggest forgetting PREREQ magic_missile shining_sphere_of_protection
LEARN magic_missile
```

```
LEARN fireball
FORGET magic_missile
LEARN shining_sphere_of_protection
LEARN flaming_mantle
ENUM
FORGET fireball
FORGET magic_missile
FORGET flaming_mantle
FORGET magic_missile
LEARN magic_missile
FORGET burning_cascade
FORGET shining_sphere_of_protection
FORGET burning_cascade
END
```

Sample Input #4: Largest Cycle

```
PREREQ fireball burning_cascade magic_missile
PREREQ burning_cascade magic_missile
PREREQ flaming_mantle burning_cascade magic_missile
PREREQ shining_sphere_of_protection burning_cascade sparks
LEARN minor_heal
PREREQ magic_missile shining_sphere_of_protection
PREREQ shining_sphere_of_protection2 shining_sphere_of_protection3 burning_cascade sparks
PREREQ shining_sphere_of_protection3 shining_sphere_of_protection2
END
```

Corresponding Sample Output #4: Largest Cycle

```
PREREQ fireball burning_cascade magic_missile
PREREQ burning_cascade magic_missile
PREREQ flaming_mantle burning_cascade magic_missile
PREREQ shining_sphere_of_protection burning_cascade sparks
LEARN minor_heal
   Learning minor_heal
PREREQ magic_missile shining_sphere_of_protection
   Found cycle in prereqs
   Suggest forgetting PREREQ magic_missile shining_sphere_of_protection
PREREQ shining_sphere_of_protection2 shining_sphere_of_protection3 burning_cascade sparks
   Found cycle in prereqs
   Suggest forgetting PREREQ magic_missile shining_sphere_of_protection
PREREQ shining_sphere_of_protection3 shining_sphere_of_protection2
   Found cycle in prereqs
   Suggest forgetting PREREQ magic_missile shining_sphere_of_protection
END
```

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

```
public void execNSpecswCheckRecLarge (Vector<String> specs, Integer N) {
  // PRE: specs contains set of specifications read in by readSpecsFromFile()
  // POST: executed min(N, all) specifications, checking for cycles and
  //       recommending fix by removing largest cycle,
  //       returning required output, one line per string in vector

  // TODO
}
```

T4 The task in **T3** should be repeated, but recommending the most recently encountered dependency statement from the *smallest* cycle.

Sample Input #5: Smallest Cycle

```
PREREQ fireball burning_cascade magic_missile
PREREQ burning_cascade magic_missile
```

```
PREREQ flaming_mantle burning_cascade magic_missile
PREREQ shining_sphere_of_protection burning_cascade sparks
LEARN minor_heal
PREREQ shining_sphere_of_protection2 shining_sphere_of_protection3 burning_cascade sparks
PREREQ shining_sphere_of_protection3 shining_sphere_of_protection2
PREREQ magic_missile shining_sphere_of_protection
```

Corresponding Sample Output #5: Smallest Cycle

```
PREREQ fireball burning_cascade magic_missile
PREREQ burning_cascade magic_missile
PREREQ flaming_mantle burning_cascade magic_missile
PREREQ shining_sphere_of_protection burning_cascade sparks
LEARN minor_heal
   Learning minor_heal
PREREQ shining_sphere_of_protection2 shining_sphere_of_protection3 burning_cascade sparks
PREREQ shining_sphere_of_protection3 shining_sphere_of_protection2
   Found cycle in prereqs
   Suggest forgetting PREREQ shining_sphere_of_protection3 shining_sphere_of_protection2
PREREQ magic_missile shining_sphere_of_protection
   Found cycle in prereqs
   Suggest forgetting PREREQ shining_sphere_of_protection3 shining_sphere_of_protection2
```

Sample Input #6: Smallest Cycle

```
PREREQ fireball burning_cascade magic_missile
PREREQ burning_cascade magic_missile
PREREQ flaming_mantle burning_cascade magic_missile
PREREQ shining_sphere_of_protection burning_cascade sparks
LEARN minor_heal
PREREQ magic_missile shining_sphere_of_protection
PREREQ shining_sphere_of_protection2 shining_sphere_of_protection3 burning_cascade sparks
PREREQ shining_sphere_of_protection3 shining_sphere_of_protection2
```

Corresponding Sample Output #6: Smallest Cycle

```
PREREQ fireball burning_cascade magic_missile
PREREQ burning_cascade magic_missile
PREREQ flaming_mantle burning_cascade magic_missile
PREREQ shining_sphere_of_protection burning_cascade sparks
LEARN minor_heal
   Learning minor_heal
PREREQ magic_missile shining_sphere_of_protection
   Found cycle in prereqs
   Suggest forgetting PREREQ magic_missile shining_sphere_of_protection
PREREQ shining_sphere_of_protection2 shining_sphere_of_protection3 burning_cascade sparks
   Found cycle in prereqs
   Suggest forgetting PREREQ magic_missile shining_sphere_of_protection
PREREQ shining_sphere_of_protection3 shining_sphere_of_protection2
   Found cycle in prereqs
   Suggest forgetting PREREQ shining_sphere_of_protection3 shining_sphere_of_protection2
```

```
public void execNSpecswCheckRecSmall (Vector<String> specs, Integer N) {
  // PRE: specs contains set of specifications read in by readSpecsFromFile()
  // POST: executed min(N, all) specifications, checking for cycles and
  //       recommending fix by removing smallest cycle,
  //       returning required output, one line per string in vector

  // TODO
}
```

4 The Code Bundle You'll Be Working With

There's a code bundle that contains the classes described in §3 that you need to complete. The code bundle has been saved as an Eclipse archive file, so you just need to open it as an archive file.

There are also some sample input and output data files containing specifications and corresponding expected solutions. Some of the JUnit tests in `SampleTests.java` refer to a file location where you will store these files. The default location is from my own installation; you'll need to change it to yours. If you're on a Windows machine, it might look something like:

```
String PATH = "C:/Users/YourAccountName/path/to/your/comp2010/data/";
```

At the start, the JUnit tests will give you all failures and errors, because the methods still have to be implemented.

The methods with non-void return types start with `return` statements that return empty values; these are just to make sure that the code compiles when you make a submission for any methods that you haven't completed. You should of course change this statement when you complete a method.

In working on this assignment, you should add your own JUnit tests as part of your code development. A specific suggestion for this problem is that you write JUnit tests for incrementally longer subsegments of the input file (i.e. for the method argument N set to $6, 7, 8, 9, \ldots$ lines of input).

5 What To Hand In

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

Submit an Eclipse archive (zip) file containing the Java BuildSpellbook class in the package from the original assignment code bundle. (Note that this is a different format from Assignment 1.) The Java class(es) should be in the form of the `.java` files.

You shouldn't submit a JUnit test file. The automarker will use its own JUnit test file which will be similar but not identical to `AllTest.java` (e.g. it will use additional data).

You should not use any non-standard libraries: the automarker cannot be expected to have access to them.

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 automarker assumes the names given. Do not change the package statement. You may, however, include additional auxiliary methods if you need them. (It should also be possible to add extra classes.)

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

Also, there will be trial automark opportunities to submit your code before the deadline to make sure that it compiles. **You are strongly encouraged to do this,** as it is your one way of making sure that you do not get zero for an unanticipated compilation error. It is especially important to do this if you have, e.g. used any libraries you think might possibly be non-standard, or added new classes. Dates for these pre-submission opportunities will be announced in advance.