Bash Assignment

Bas Terwijn

1 Introduction

In this assignment we will write Bash scripts to change and analyze the Java code in git repository:

- `https://bitbucket.org/bterwijn/knapsack`

Clone this repository. For this assignment you will be mostly working in directory "./individual/".

2 Find and Replace

Write script **"find_name.sh"** that prints the file name and line number of each occurrence of a name (just a string) in all Java source files in the repository. The script should be usable as:

```
./find_name.sh <name>
```

where '<name>' is any name without white spaces. Also write script **"rename_name.sh"** to find all occurrences of '<name-old>' in all Java source files in this repository and replace it with the '<name-new>'. No need to parse the Java syntax or check if the code still compiles afterwards. The script is intended as just simple refactor tool, a simple string-based find-and-replace will do. The script should be usable as:

```
./rename_name.sh <name-old> <name-new>
```

Also write script **"rename_class.sh"** to rename a Java class in the repository and all its occurrences of the name throughout the code base. When you rename a class in Java you will also have to rename the file the class is in, therefore rename the file as well. The script should be usable as:

```
./rename_class.sh <class-old> <class-new>
```

3 Keyboard shortcuts

To work a lot faster in the Bash shell and to make it more enjoyable use keyboard shortcuts. Take some time now to practice using them and see which you like best:

shortcut key	***effect***
Tab	auto-completes filename, press again when stuck to see options
middle-mouse-button	copy selected text
Ctrl-p or Cursor-Up	previous command in command history
Ctrl-n or Cursor-Down	next command in command history
Ctrl-r <type-something>	search in command history
Alt-.	last word of previous commands in command history
Ctrl-a / Ctrl-e	move cursor to beginning/end of line
Ctrl-f / Alt-f	move cursor forward one character/word
Ctrl-b / Alt-b	move cursor backwards one character/word
Ctrl-xx	jump between current position and beginning of line
Ctrl-d / Alt-d	delete one character/word
Ctrl-h / Ctrl-w	backspace one character/word
Ctrl-k	delete all to the right of cursor
Ctrl-u	backspace all to the left of cursor
Ctrl-/	undo last edit
Ctrl-l	clear screen

Most shortcuts are the same as for the "emacs" text editor.

4 Class diagram

In file "example_class_diagram.dot" you find an example class diagram:

```
digraph D {

  A -> B                    # for inheritance
  C -> B [arrowhead=diamond] # for composition
  B -> D [arrowhead=dot]     # for import

  subgraph cluster_X { label = "package X"
    A [shape=box]
    B [shape=box]
  }

  subgraph cluster_Y { label = "package Y"
    C [shape=box]
    D [shape=box]
  }

}
```

Install program "graphviz" and generate a pdf file based on this file with:

```
dot -Tpdf example_class_diagram.dot > example_class_diagram.pdf
```

which should result in a pdf file depicted in figure 1. The class diagram shows different relations between class A, B, C and D:

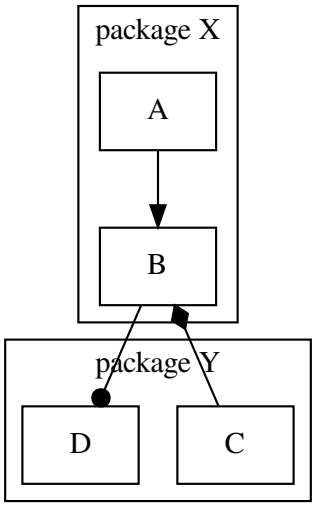

Figure 1: example_class_diagram.pdf

classes	***relation***
A	extends B (inheritance)
B	has an "imports D" statement in its file
C	has one or more objects of type B as member variables (composition)
A,B	are in package X
C,D	are in package Y

Now write the following scripts to generate similar diagrams for the classes in the Java source files in this repository:

script	***diagram that shows***
generate_diagram_classes.sh	just all classes
generate_diagram_extends.sh	all extends relations
generate_diagram_import.sh	all import relations
generate_diagram_composition.sh	all composition relations
generate_diagram_package.sh	all package relations
generate_diagram_all.sh	all relations above

Parse the Java sources to find the relations for example with regular expressions. Script "generate_diagram_classes.sh" is already given as an example. For other useful regular expression examples see "example_perl_regex.sh". The script "generate_diagrams.sh" should then afterwards generate all ".dot" files for each diagrams.

Parsing of source code with a complex syntax such as the Java language is in general very difficult. Therefore, your scripts only have to work for the sources in the Knapsack repository that for example have no nested classes. Classes that are not defined in this repository don't have to be added to the diagrams but you are allowed to add them if that is easier. An example of the diagram_all.pdf diagram is given in figure 2, your diagram doesn't have to match exactly.

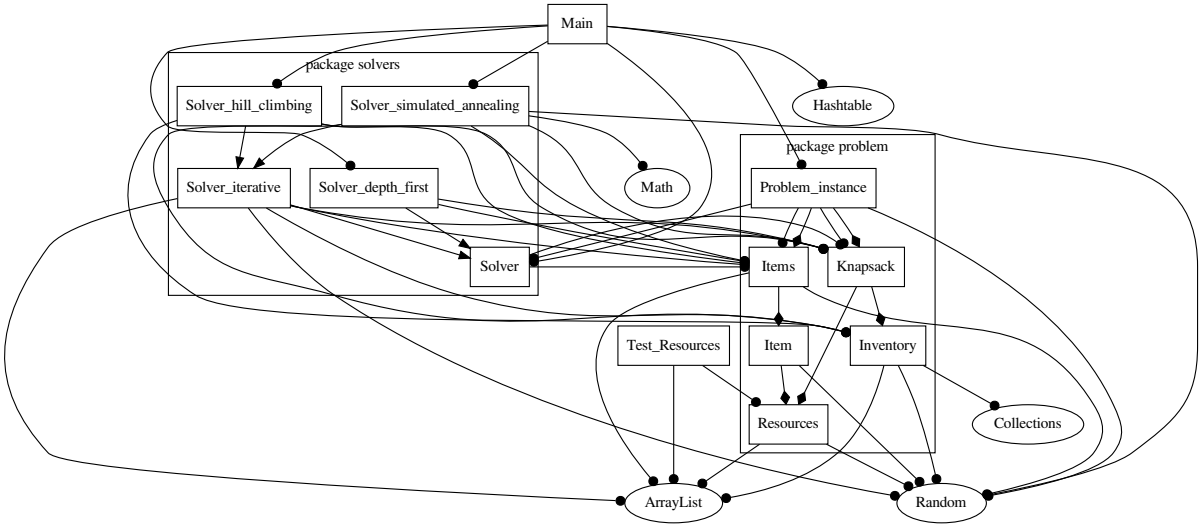

Figure 2: Example of the diagram_all.pdf diagram

5 Makefile

Add to the **"Makefile"** in the root directory so that with:

- **"make diagrams_dot"**, each dot file for each diagram gets generated
- **"make diagrams"**, a pdf is generated for each dot file that is present. For this you must use a **makefile pattern rule** so that only a pdf is re-generated when its dot file's modification time is later. See as example the "%.class: %.java" pattern in the Makefile, or the "%.ps: %.txt" pattern in the lecture slides.
- **"make clean"**, also all the files generated by your scripts (dot, pdf, and possibly other temporary files) in "./individual" get removed. Be very careful not to remove your own scripts by accident! Make a backup.

6 PATH environment variable

To run a particular script from inside an arbitrary directory without a path prefix (for example "./") you need to:

- Add the global directory of the script to your "PATH" environment variable. This is where the shell searches for commands (ls,mkdir,cp,etc.) you type.
- Set the executable bit of the script using "chmod +x <script>.
- Add the "#!/bin/bash" shebang line as the first line of the scripts to define how the script is to be executed.

To experiment with this create directory "$HOME/bin" and create some script in this directory. Then add line "export PATH=$PATH:$HOME/bin" to your shell start-up file (file "$HOME/.bashrc" for Ubuntu, file "$HOME/.profile" for MacOS) so that this new directory will be added to your "PATH" variable every time you start a new shell. Start a new shell/terminal and check with "echo $PATH". Then add the executable bit and shebang and try to execute your script. When this works "$HOME/bin" is your new place to collect useful scripts.

7 Submission

Submit your solutions on Canvas after cleaning up the repository (make clean) and zipping it as a whole to knapsack.zip. Double check that all your solution files are included.