# CS24210: Intelligent Editing

Edel Sherratt

7 October 2000

This practical is about using pattern matching and substitution to edit a number of files containing Java programs that implement some simple stack programs. First, you will use matching and substitution to improve the readability of the stack programs, and then you will use similar operations to change the stack programs to a set of Queue programs. The editors you will use are `gvim`, a screen editor, and `sed`, a stream editor.

`gvim` has two modes – an insert mode and a command mode. Commands can be issued directly, or, usually in the case of more elaborate commands, by typing : followed by the rest of the command. It is this last kind of command that you will be using.

Commands in both editors look rather similar. For example, here is the general form of the substitute command, which you will be using.

$< range > s/ < pattern > / < replacement > / < flags >$

$< range >$ says which lines are to be modified, $s$ is the substitute command itself, and $< pattern >$ and $< replacement >$ indicate the changes to be made. $ is a special symbol, which means the last line in the file, when it appears in a $< range >$, and which means the end of line when it appears in a $< pattern >$. The flag you'll mainly use will be $g$, which says do the replacement throughout each line visited, and not just once.

If you happen to get into insert mode while using `gvim`, you can get back to command mode by typing *escape*. You can undo the most recent command by typing `u`. And, if all else fails, you can always restart the practical.

Note: while it is not essential to know Java to complete this practical (which is concerned with pattern matching and substitution), you may still want to know what the Java code does. If you want to know more about the programs, ask one of the demonstrators, or ask Edel Sherratt in the next lecture.

Now – have fun.

1. Download a copy of EasyStack.tar from the CS24210 web page. Unpack it using `tar xvf EasyStack.tar`. This will create a subdirectory called EasyStack in your current working directory. The subdirectory will contain Stack.java, Fruit.java and five test harnesses – StackHarness1.java, ... StackHarness5.java.

2. Change directory to EasyStack: `cd EasyStack`. Your next task is to improve the identifiers in Stack.java using the `gvim` editor: `gvim Stack.java`. Here are some `gvim` commands to do this:

```
:1,$s/I\>/Interface/g
:1,$s/D\>/DataType/g
:1,$s/V\>/Visitor/g

:1,$s/\<s\>/stack/g
:1,$s/\<_s\>/_stack/g

:1,$s/\<f\>/thing/g
:1,$s/\<_f\>/_thing/g
```

Note the use of

```
\<
```

and

```
\>
```

to mark the beginnings and ends of words.

Can you describe in your own words what strings are matched by each of the following two patterns?

```
I\>
```

```
\<f\>
```

If not, ask a demonstrator to help, or take a look at `man -s 5 regexp`

Use `:x` to exit `gvim` – or use the `save-quit` option on the `file` pull down menu.

Now use pattern matching and replacement to improve the identifiers in each of the StackHarness files. Use `gvim StackHarness[1-5].java` to edit each of these files in turn. Use the following two substitutions to improve each file.

```
:1,$s/D\>/DataType/g
:1,$s/V\>/Visitor/g
```

When you've finished editing StackHarness1.java, use `:w` to save it and `:n` to move on to the next file. Don't use the `save-quit` option this time!

When you've finished editing StackHarness.java, use `:x` (or `save-quit`) to exit.

3. Compile and run Stack.java and Fruit.java with each of the five test harnesses in turn.

```
javac Stack.java
javac Fruit.java
javac StackHarness[1-5].java

java StackHarness1
```

```
...

java StackHarness5
```

4. Copy EasyStack and all its contents to a new directory, called EasyQueue.

   ```
   cd ..
   cp -r EasyStack EasyQueue
   ```

   In the EasyQueue directory – `cd EasyQueue` – rename Stack.java to Queue.java: `mv Stack.java Queue.java`

   Can you describe in your own words what each of the following commands does?

   ```
   cd ..
   cp -r EasyStack EasyQueue
   cd EasyQueue
   mv Stack.java Queue.java
   ```

   If not, look again at the paragraph above, or ask a demonstrator, or use `man` to find out about the command(s).

5. Your next task is to modify Queue.java so that it actually implements a queue data structure and not a stack. Here are some gvim commands to do this:

   ```
   :1,$s/Stack/Queue/g
   :1,$s/Push/Join/g
   :1,$s/Pop/Leave/g

   :1,$s/stack/queue/g
   ```

   Can you explain in your own words what each of these commands does? Do you think you could use similar commands when editing your own programs? If not, ask a demonstrator.

6. Now for the clever bit. Line 43 in the file is in the class PopV, and it now says (following the above substitutions):

   ```
       return (QueueDataType) queue;
   ```

   See if you can figure out what the following gvim command will do, assuming you execute it as a **single line** without any line break:

   ```
   :43s/return (QueueDataType) queue/if (queue instanceof Empty) &;
    else return new Join(thing,(QueueDataType)queue.accept(this))
   ```

Try it, but remember, no line break.

You may wish to improve the layout of the resulting code – please do this, then save and exit using `:x` or `save-quit`.

7. It would be possible, but slightly tedious, to use gvim to edit all the StackHarness files. Instead, you will use sed to do this.

   Create a file called `sed_commands` containing the following `sed` commands:

   ```
   1,$s/Stack/Queue/g
   1,$s/Push/Join/g
   1,$s/Pop/Leave/g

   1,$s/stack/queue/g
   ```

   Create a file containing the `bash` script:

   ```
   #!bash

   let i=1
   while [ $i -le 5 ]
   do
     sed -f sed_commands StackHarness$i.java > QueueHarness$i.java
     let i=i+1
   done
   ```

   Be very careful to type spaces etc. exactly as given above.

   Now use the the command

   ```
   source <bash script file>
   ```

   to modify your StackHarnesses.

8. Finally compile and run your Queue programs:

   ```
   javac Queue.java
   javac Fruit.java
   javac QueueHarness[1-5].java

   java QueueHarness1

   ...

   java QueueHarness5
   ```