Final Exam Prep

Melvyn Ian Drag

December 5, 2019

Abstract

Prepare for exam

1 Final Exam

Set your path to point to your compiler and JVM. Then compile a simple program.

1.1 What is a PATH

The Path/PATH is an ‘environment variable’. It’s a variable that tells your operating system where to look for various programs when you try to run them. For example, on the command line, when you type ‘javac’ your computer needs to know where ‘javac’ is saved. This is why we have to put it in our PATH.

1.2 Set Up Your Path

On windows you typically set your path as shown in figure 1 and figure 2.

Unfortunately, for this class we don’t have administrator access to our lab computers, so we need a work around. To this end, we have used the ’git bash’ shell instead of the windows ‘cmd’ shell. This gives us a small Linux-like environment in which we can modify the PATH using Linux techniques. To set your path in Linux you need to modify you path in a file called ‘˜/.bashrc’

You need to find where *java* and *javac* are installed on your computer. For me they are in the location showing in figure 3.

Then I update my .bashrc file as is shown in figure 4.

Make sure you create a .bashrc file just like mine - the only thing you should hcnage is the path to where your java bin directory is. Everything else must be excatly the same! This will be saved in your home directory. This is something like

```
C:\Users\NJCU_ID
```

Note where and how I save my file in figure 5.

Note that the file is called .bashrc, not bashrc. Not bashrc.txt. Not _bashrc. Make sure the file is named correctly.

We are only using gitbash in this class instead of cmd because a) we need to get around IT restrictions b) I know linux and git bash has some linux functionality that I know well, so I can confidently teach it to you. Generally you would modify your windows environment variables as shown in the beginning of this section.

When you have successfully configured your .bashrc and open gitbash you might see a warning as demonstrated in figure 8.

Then type the 'cd' command followed by 'cat .bashrc'. You should see something like what is shown in figure 7.

At this point javac should be working, so type javac in git bash as shown in figure ??.

1.3 Compile a simple program

So part 1 of your exam is to configure your PATH and you've now done it. Part 2 of your exam is to write and compile a standard foobar program. This is a standard first interview question that verifies that you understand the absolute basics of a programming language and can implement a simple concept in the language. The problem statement is:

Write a program that will loop over the numbers from 1 to 100 (inclusive) and generate the following outputs:

- if i is a multiple of 3 print foo on a line.
- if i is a multiple of 5 print bar on a line.
- if i is a multiple of 15 (both 3 and 5) print foobar on a line.

For example, if you run the code from i = 1 to 15 instead of 1 to 100 you should get the following output:

```
melvyn@gitbash$ java foobar
foo
bar
foo
foo
bar
foo
foobar
```

where the outputs correspond to 3, 5, 6, 9, 10, 12, 15. Your final exam is to set up your PATH on an NJCU machine, then write, compile and run a foobar program and get the right answer.

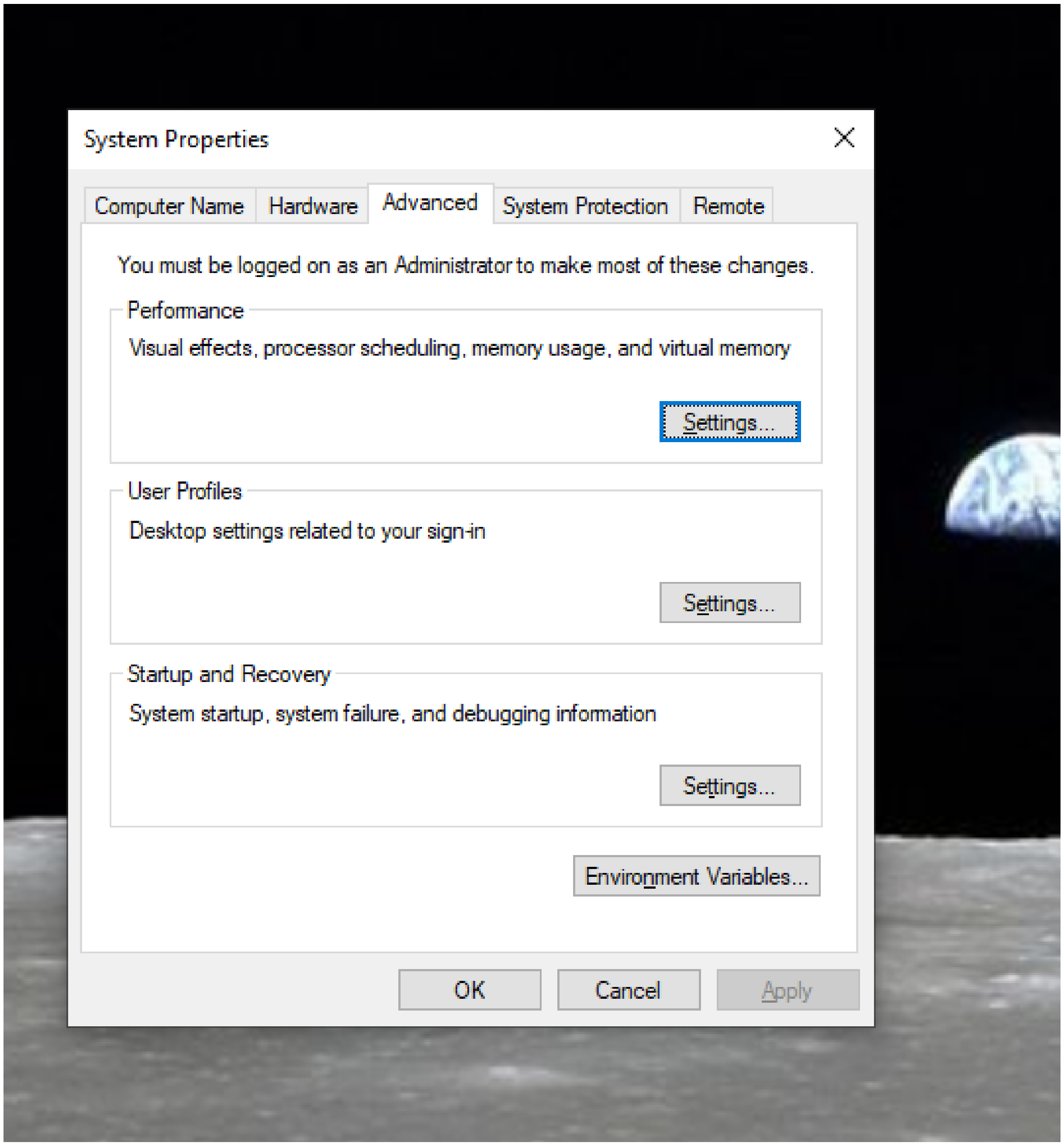
In class activity

Figure out what the output should be as a class. Write it on the board. Pretty easy stuff. It's up to the students to write the code, but we can determine the output together.

2 Grading for Final

You must do the exam on an NJCU machine. The computer will not have internet connectivity and I'll be logging all your keystrokes to verify it's you doing the work. This is why we have to do it on an NJCU machine. When we come next week I expect you here on time at 8:00 - we'll be done at 8:30 sharp, make sure you are here on time and ready to do the assignment. Get it done, get 50 pts for the path, 50 pts for the code. Get an A, please!!!!

This has been a hard semester, let's finish on a high note with a bunch of good grades.



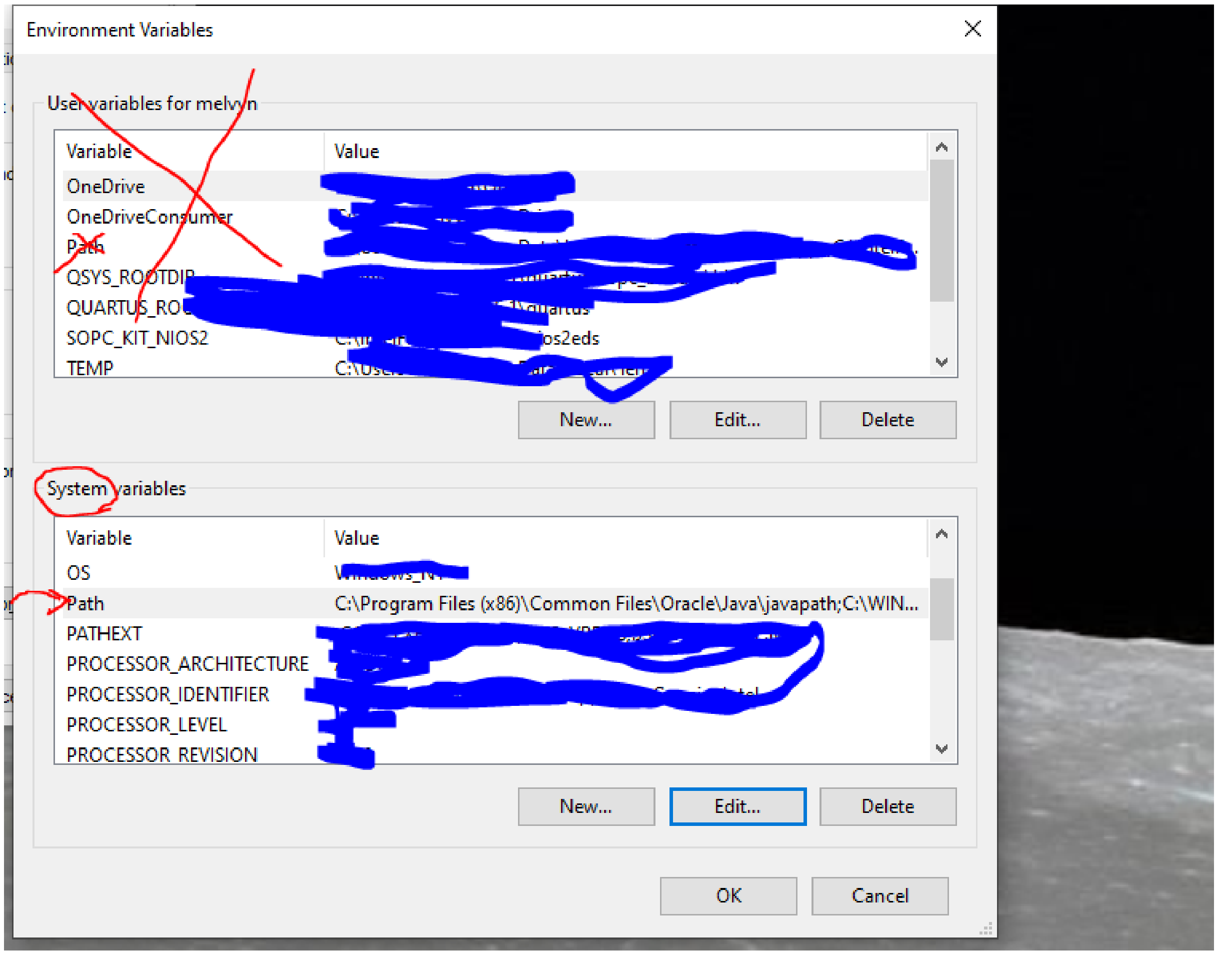
You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your sign-in

System startup, system failure, and debugging information

Figure 1: Click on ‘Environment Variables’.



Environment Variables

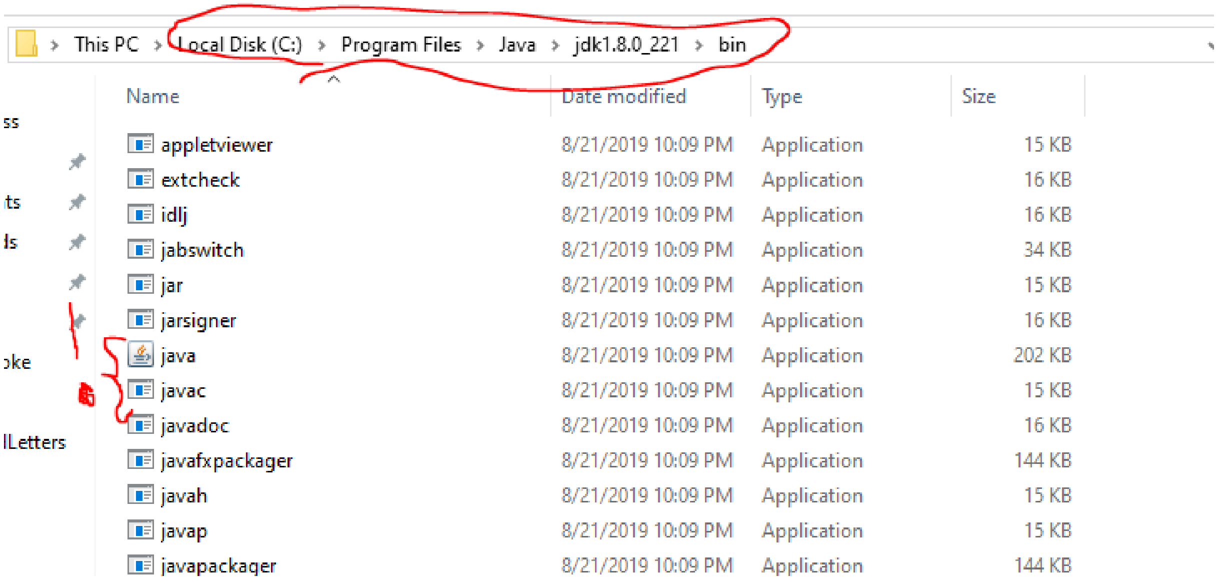
User variables for melvyn

Variable	Value
OneDrive	[redacted]
OneDriveConsumer	[redacted]
Path	[redacted]
QSYS_ROOTDIR	[redacted]
QUARTUS_RO[redacted]	[redacted]
SOPC_KIT_NIOS2	[redacted]
TEMP	[redacted]

System variables

Variable	Value
OS	[redacted]
Path	C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\WIN...
PATHEXT	[redacted]
PROCESSOR_ARCHITECTURE	[redacted]
PROCESSOR_IDENTIFIER	[redacted]
PROCESSOR_LEVEL	[redacted]
PROCESSOR_REVISION	[redacted]

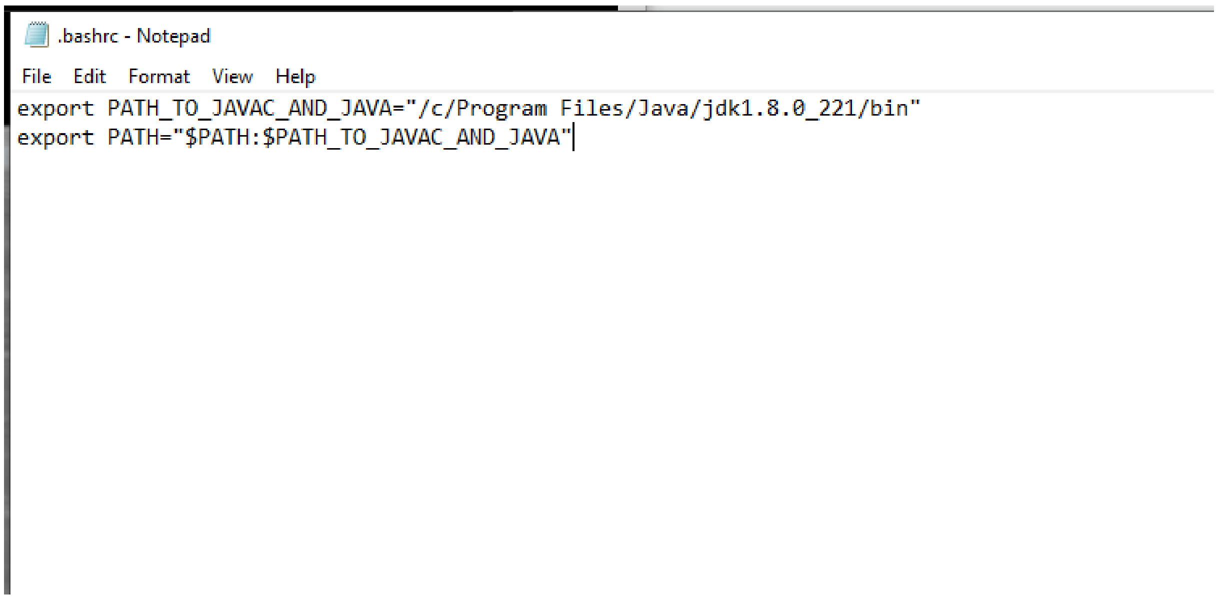
Figure 2: Modify what is in your PATH. Crossing stuff out in case there is anything you might use to hack me or whatever, this is my home PC.



This PC › Local Disk (C:) › Program Files › Java › jdk1.8.0_221 › bin

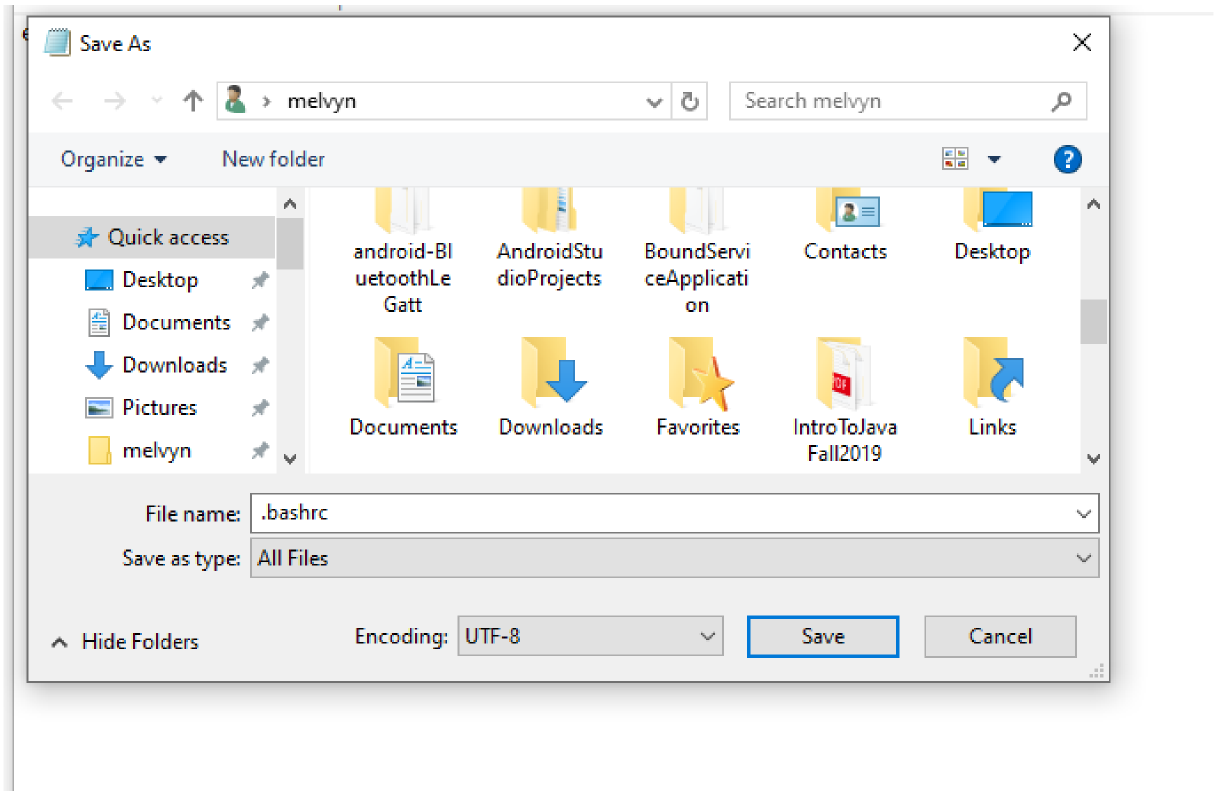
Name	Date modified	Type	Size
appletviewer	8/21/2019 10:09 PM	Application	15 KB
extcheck	8/21/2019 10:09 PM	Application	16 KB
idlj	8/21/2019 10:09 PM	Application	16 KB
jabswitch	8/21/2019 10:09 PM	Application	34 KB
jar	8/21/2019 10:09 PM	Application	15 KB
jarsigner	8/21/2019 10:09 PM	Application	16 KB
java	8/21/2019 10:09 PM	Application	202 KB
javac	8/21/2019 10:09 PM	Application	15 KB
javadoc	8/21/2019 10:09 PM	Application	16 KB
javafxpackager	8/21/2019 10:09 PM	Application	144 KB
javah	8/21/2019 10:09 PM	Application	15 KB
javap	8/21/2019 10:09 PM	Application	15 KB
javapackager	8/21/2019 10:09 PM	Application	144 KB

Figure 3: Install location of my java and javac



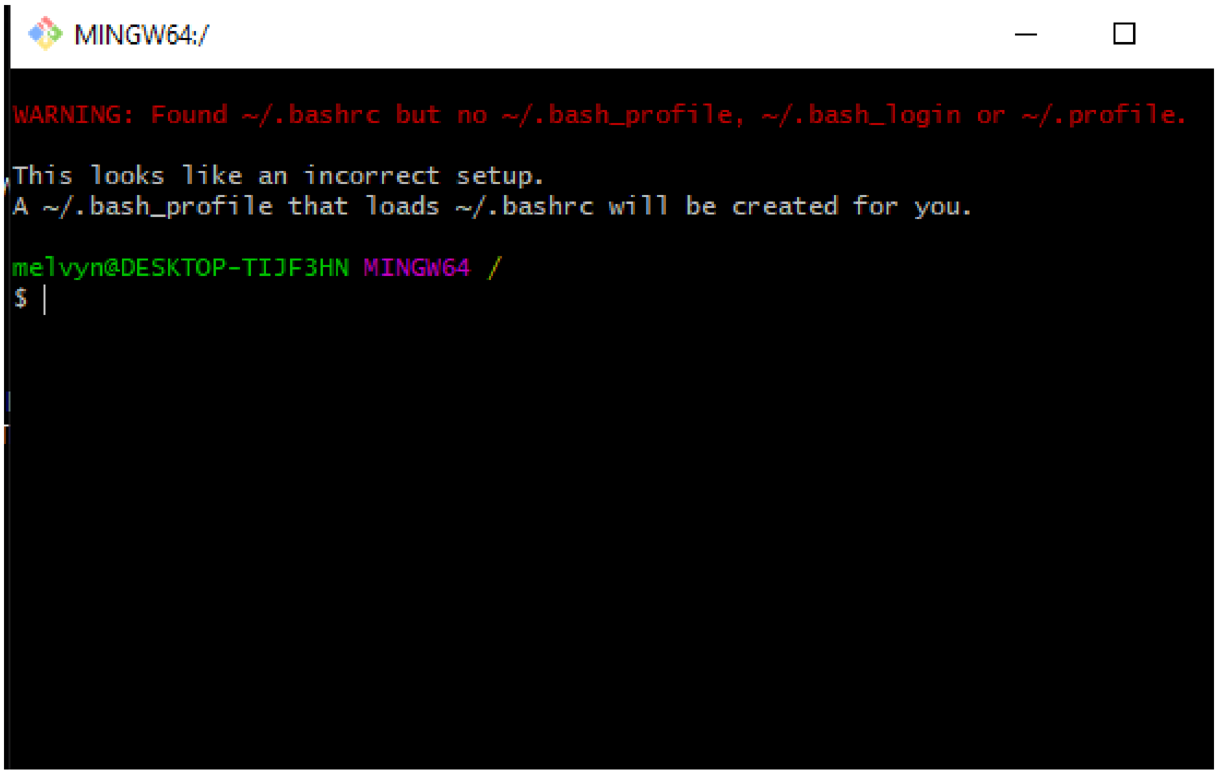
```
export PATH_TO_JAVAC_AND_JAVA="/c/Program Files/Java/jdk1.8.0_221/bin"
export PATH="$PATH:$PATH_TO_JAVAC_AND_JAVA"
```

Figure 4: my bashrc file



File name: .bashrc
Save as type: All Files
Encoding: UTF-8

Figure 5: where to save .bashrc

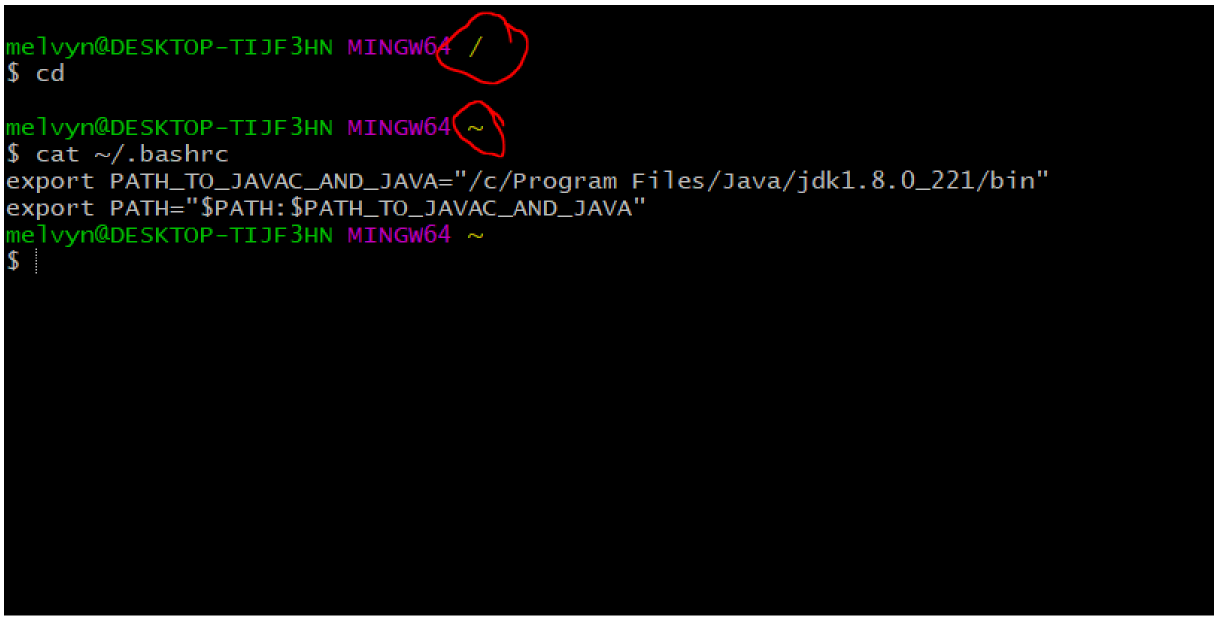


```
WARNING: Found ~/.bashrc but no ~/.bash_profile, ~/.bash_login or ~/.profile.

This looks like an incorrect setup.
A ~/.bash_profile that loads ~/.bashrc will be created for you.

melvyn@DESKTOP-TIJF3HN MINGW64 /
$ 
```

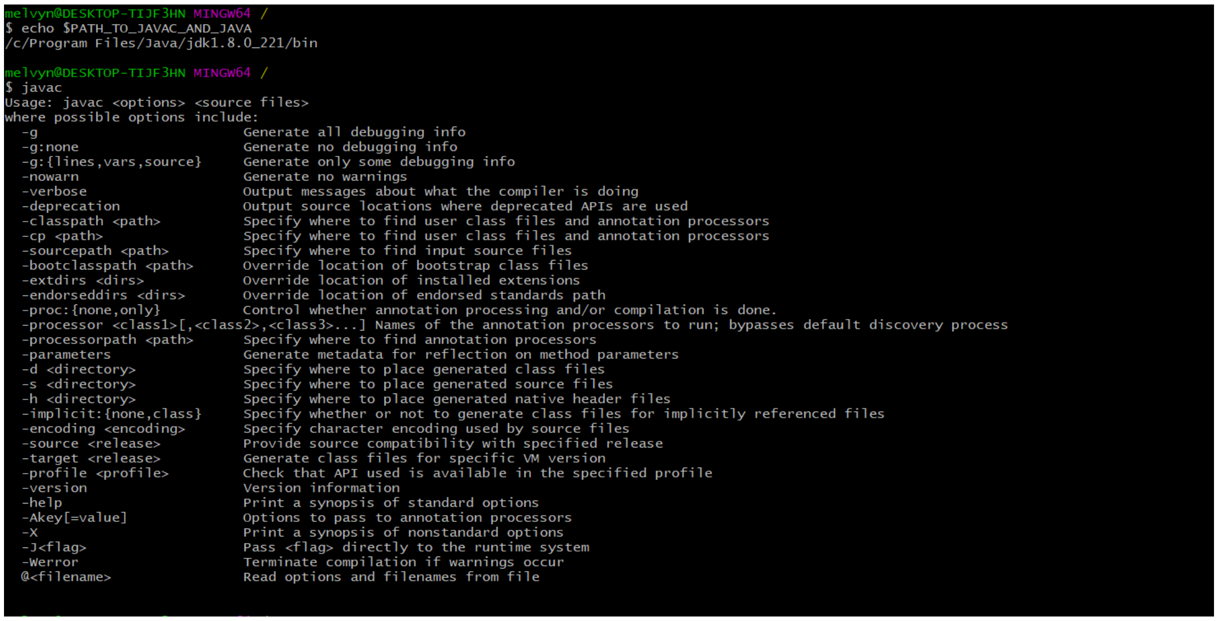
Figure 6: You might see this warning. Ignore it.



```
melvyn@DESKTOP-TIJF3HN MINGW64 /
$ cd

melvyn@DESKTOP-TIJF3HN MINGW64 ~
$ cat ~/.bashrc
export PATH_TO_JAVAC_AND_JAVA="/c/Program Files/Java/jdk1.8.0_221/bin"
export PATH="$PATH:$PATH_TO_JAVAC_AND_JAVA"
melvyn@DESKTOP-TIJF3HN MINGW64 ~
$ 
```

Figure 7: checking the contents of .bashrc



```
melvyn@DESKTOP-TIJF3HN MINGW64 /
$ echo $PATH_TO_JAVAC_AND_JAVA
/c/Program Files/Java/jdk1.8.0_221/bin

melvyn@DESKTOP-TIJF3HN MINGW64 /
$ javac
Usage: javac <options> <source files>
where possible options include:
  -g                         Generate all debugging info
  -g:none                    Generate no debugging info
  -g:{lines,vars,source}     Generate only some debugging info
  -nowarn                    Generate no warnings
  -verbose                   Output messages about what the compiler is doing
  -deprecation               Output source locations where deprecated APIs are used
  -classpath <path>          Specify where to find user class files and annotation processors
  -cp <path>                 Specify where to find user class files and annotation processors
  -sourcepath <path>         Specify where to find input source files
  -bootclasspath <path>      Override location of bootstrap class files
  -extdirs <dirs>            Override location of installed extensions
  -endorseddirs <dirs>       Override location of endorsed standards path
  -proc:{none,only}          Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process
  -processorpath <path>      Specify where to find annotation processors
  -parameters                Generate metadata for reflection on method parameters
  -d <directory>             Specify where to place generated class files
  -s <directory>             Specify where to place generated source files
  -h <directory>             Specify where to place generated native header files
  -implicit:{none,class}     Specify whether or not to generate class files for implicitly referenced files
  -encoding <encoding>       Specify character encoding used by source files
  -source <release>          Provide source compatibility with specified release
  -target <release>          Generate class files for specific VM version
  -profile <profile>         Check that API used is available in the specified profile
  -version                   Version information
  -help                      Print a synopsis of standard options
  -Akey[=value]              Options to pass to annotation processors
  -X                         Print a synopsis of nonstandard options
  -J<flag>                   Pass <flag> directly to the runtime system
  -Werror                    Terminate compilation if warnings occur
  @<filename>                Read options and filenames from file
```

Figure 8: All set up and ready to go.