

Detailed guide to run Demo/Callables on localhost

1.0: Running the Demo on Windows

2.0: Running the Demo on Linux

3.0: Additional Resources

3.1: Callables Demo Breakdown

3.2: installing Git

3.3: installing just JRE

3.4: installing JDK

1.0 WINDOWS

The minimum requirement to run this demo is Java Runtime Environment (JRE) version 8 or higher. It is recommended to download the Java SE Development Kit (JDK) version 8 or higher. This project uses Java RMI registry, which is a java program available with the JDK. RMI registry is included in the project files for those that do not want to install the JDK. See the '3.3 Installing JRE' section or the '3.4 Installing JDK' section for instructions to install those. Remember that the JDK installation includes the JRE, so you do not need both.

This guide will be using Git bash on windows, which is included when you install Git (see '3.2 Installing Git' section). One could also download the zip file from github and run the demo in Windows Powershell. Be sure to run your bash as administrator if you receive permission warnings.

1.1 WITH JDK

First, clone the repository using 'git clone

<https://github.com/JonathanTF/DistributedExecutorService.git>'. Your main directory should look something like fig 1.

```
Jonathan@DESKTOP-AML7NU4 ~/DistributedExecutorService (master)
$ ls
Detailed_Demo_Callables.pdf  Oracle  WorkNode.jar  rundemo.sh  src  startreg_win.sh
LICENSE                     README.md  bin          rundemo_nojdk.sh  startreg_unix.sh
```

Fig. 1: main directory

To begin, you must start the RMRegistry in the bin directory so it has access to the RemoteMethods class.

```
Jonathan@DESKTOP-AML7NU4 ~/DistributedExecutorService (master)
$ cd bin/

Jonathan@DESKTOP-AML7NU4 ~/DistributedExecutorService/bin (master)
$ rmiregistry 5555
```

Fig. 2: starting RMI registry

For a more practical purpose, one would background (&) this process. From here we'll start a work node, which is the actual program that handles the distributed executor service's work. There is a .jar file in the main directory you can run that starts a work node:

```
Jonathan@DESKTOP-AML7NU4 ~/DistributedExecutorService (master)
$ java -jar WorkNode.jar localhost 5555 debug
```

Fig. 3: starting worknode

This starts a work node and connects to an RMI registry on localhost at port 5555 with debug enabled. It is also possible to start a work node from the bin directory using the command

- `java worknode.WorkNode <host> <port> <debug>`

Passing more than two arguments will enable debugging, which outputs a constant tick every second of how many requests the node is currently handling.

```
Jonathan@DESKTOP-AML7NU4 ~/DistributedExecutorService (master)
$ java -jar WorkNode.jar localhost 5555 debug
Debug Activated
~I have 0 Tasks~
Node ready: 3d3b97f6:15f8aea7917:-7fff
~I have 0 Tasks~
~I have 0 Tasks~
~I have 0 Tasks~
~I have 0 Tasks~
~I have 0 Tasks~
~I have 0 Tasks~
```

Fig. 4: running worknode

The node takes some time to initialize and then constantly outputs 0 tasks. To run the demo, we will open another terminal to connect on localhost. You can either run the rundemo.sh script from the main directory which has the following format

- `./rundemo.sh <Demo_To_Run> <host> <port> <arg3> <arg4> ...`

Or run the java program directly from the bin/ directory using

- `java Demo.<Demo_To_run> <host> <port> <arg3> <arg4> ...`

The rundemo.sh script will rebuild the source files and create the bin directory if it does not exist. For each demo you pass the host and port the RMI registry is on and any arguments that a Demo might take. A screenshot of running the script for the Callables demo next to the work node terminal is shown below.

```

Jonathan@DESKTOP-AML7NU4 ~/DistributedExecutorService (master)
$ ./rundemo.sh Callables localhost 5555
building Callables
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Running Callables with parameters localhost:5555...
[Ljava.lang.String;@4e50df2e

Jonathan@DESKTOP-AML7NU4 ~/DistributedExecutorService (master)
$ java -jar WorkNode.jar localhost 5555 debug
Debug Activated
~I have 0 Tasks~
Node ready: 2cabfd5:15f8afe631a:-7fff
~I have 0 Tasks~
~I have 0 Tasks~
~I have 0 Tasks~
Feeling Sleepy...
~I have 6 Tasks~
~I have 6 Tasks~
~I have 6 Tasks~
Hello World!
Feeling Sleepy...
~I have 5 Tasks~
~I have 5 Tasks~

```

Fig. 5: running Callables Demo

You have now successfully completed the demo! If there are exceptions thrown in regards to the connection to the RMI server, it is recommended to terminate all work nodes / running programs and restart the RMI registry. For a complete breakdown of what the Callables Demo does, see the '3.1 Callables Breakdown' section below.

1.2 JRE ONLY

Download the git repository as shown below.

```

Jonny@DESKTOP-K5LNIC4 MINGW64 ~/Desktop
$ git clone https://github.com/JonathanTF/DistributedExecutorService.git
Cloning into 'DistributedExecutorService'...
remote: Counting objects: 202, done.
remote: Compressing objects: 100% (138/138), done.
remote: Total 202 (delta 83), reused 174 (delta 60), pack-reused 0
Receiving objects: 100% (202/202), 1.63 MiB | 326.00 KiB/s, done.
Resolving deltas: 100% (83/83), done.

Jonny@DESKTOP-K5LNIC4 MINGW64 ~/Desktop
$ cd DistributedExecutorService/

Jonny@DESKTOP-K5LNIC4 MINGW64 ~/Desktop/DistributedExecutorService (master)
$ ls
bin/                               javac      rundemo.sh*      src/
Callables_InDepthDemoGuide.pdf    LICENSE    rundemo_nojdk.sh* startreg_unix.sh*
'Detailed guide to run Demo Callables.pdf' Oracle/    rundemo_nosdk_unix.sh* startreg_win.sh*
Detailed_Demo_Callables.pdf       README.md  rundemo_nosdk_win.sh* WorkNode.jar

```

Fig. 6: main directory

Run './startreg_win.sh <port>' to start an RMI registry that uses the provided registry file

```

Jonny@DESKTOP-K5LNIC4 MINGW64 ~/Desktop/DistributedExecutorService (master)
$ ./startreg_win.sh 5555
RMI registry started, hit Ctrl-C to terminate

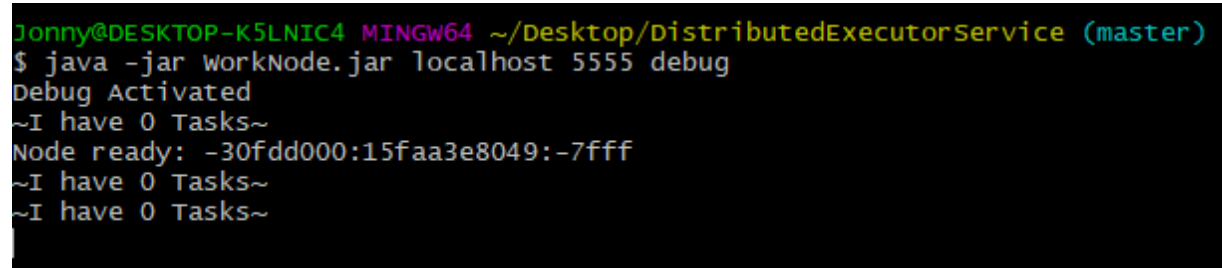
```

Fig. 7: start RMI registry – no JDK

Be sure to allow access if Windows Firewall gives you a warning. Start a worknode in a new terminal using the command:

- `java worknode.WorkNode <host> <port> <debug>`

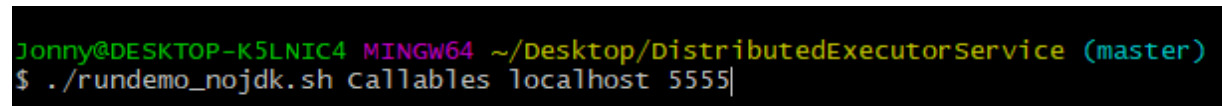
It takes a hostname and port number, as well as a debug option that prints extra information to the terminal.

A terminal window with a black background and green text. The prompt is 'Jonny@DESKTOP-K5LNIC4 MINGW64 ~/Desktop/DistributedExecutorService (master)'. The command entered is '\$ java -jar workNode.jar localhost 5555 debug'. The output is 'Debug Activated', '~I have 0 Tasks~', 'Node ready: -30fdd000:15faa3e8049:-7fff', '~I have 0 Tasks~', and '~I have 0 Tasks~'.

```
Jonny@DESKTOP-K5LNIC4 MINGW64 ~/Desktop/DistributedExecutorService (master)
$ java -jar workNode.jar localhost 5555 debug
Debug Activated
~I have 0 Tasks~
Node ready: -30fdd000:15faa3e8049:-7fff
~I have 0 Tasks~
~I have 0 Tasks~
```

Fig. 8: start a worknode

Then start a third terminal and run the command below to start the Callables Demo.

A terminal window with a black background and green text. The prompt is 'Jonny@DESKTOP-K5LNIC4 MINGW64 ~/Desktop/DistributedExecutorService (master)'. The command entered is '\$./rundemo_nojdk.sh callables localhost 5555'.

```
Jonny@DESKTOP-K5LNIC4 MINGW64 ~/Desktop/DistributedExecutorService (master)
$ ./rundemo_nojdk.sh callables localhost 5555
```

Fig. 9: run Callables Demo – no JDK

The following two figures show the output of the worknode and the Callables demo.

```

MINGW64/c/Users/Jonny/Desktop/DistributedExecutorService
Jonny@DESKTOP-K5LNIC4 MINGW64 ~/Desktop/DistributedExecutorService (master)
$ ./rundemo_nojdk.sh callables localhost 5555
Running Callables with parameters localhost:5555...
[Ljava.lang.String;@4e50df2e

MINGW64/c/Users/Jonny/Desktop/DistributedExecutorService
Jonny@DESKTOP-K5LNIC4 MINGW64 ~/Desktop/DistributedExecutorService (master)
$ java -jar workNode.jar localhost 5555 debug
Debug Activated
~I have 0 Tasks~
Node ready: 5a12edec:15faa408de3:-7fff
~I have 0 Tasks~
~I have 0 Tasks~
~I have 0 Tasks~
~I have 0 Tasks~
~I have 0 Tasks~
~I have 0 Tasks~
Feeling Sleepy...
~I have 6 Tasks~
~I have 6 Tasks~
~I have 6 Tasks~
~I have 6 Tasks~
Hello world!
Feeling Sleepy...

MINGW64/c/Users/Jonny/Desktop/DistributedExecutorService
Jonny@DESKTOP-K5LNIC4 MINGW64 ~/Desktop/DistributedExecutorService (master)
$ ./rundemo_nojdk.sh callables localhost 5555
Running Callables with parameters localhost:5555...
[Ljava.lang.String;@4e50df2e
On return we got I'm Done!I'm Done!I'm Done!I'm Done!I'm Done!I'm Done!
Done!

MINGW64/c/Users/Jonny/Desktop/DistributedExecutorService
Jonny@DESKTOP-K5LNIC4 MINGW64 ~/Desktop/DistributedExecutorService (master)
$ java -jar workNode.jar localhost 5555 debug
Debug Activated
~I have 4 Tasks~
Feeling Sleepy...
~I have 3 Tasks~
~I have 3 Tasks~
~I have 3 Tasks~
Hello world!
Feeling Sleepy...
~I have 2 Tasks~
~I have 2 Tasks~
~I have 2 Tasks~
~I have 2 Tasks~
Hello world!
Feeling Sleepy...
~I have 1 Tasks~
~I have 1 Tasks~
~I have 1 Tasks~
~I have 1 Tasks~
Hello world!
~I have 0 Tasks~
~I have 0 Tasks~
~I have 0 Tasks~

```

Fig. 10: screenshots of running Callables Demo

You have now successfully completed the demo! If there are exceptions thrown in regards to the connection to the RMI server, it is recommended to terminate all work nodes / running programs and restart the RMI registry. For a complete breakdown of what the Callables Demo does, see the '3.1 Callables Breakdown' section below.

2.0 LINUX

Install Git and JDK as recommended

```
~$ sudo apt install git
```

```
~$ sudo apt install openjdk-8-jdk
```

run `java -version` and `javac -version` to make sure everything was installed. If not, add the binaries to your Path. Next, start up the demo.

```
~$ git clone https://github.com/JonathanTF/DistributedExecutorService.git
```

```
~$ cd DistributedExecutorService/
```

You will need multiple terminals, or background the RMI registry and WorkNode

On one terminal:

```
~/DistributedExecutorService$ cd bin/
```

```
~/DistributedExecutorService/bin$ rmiregistry 5555
```

This starts an RMI registry on port 5555

On terminal two:

```
~/DistributedExecutorService$ java -jar WorkNode.jar localhost 5555 debug
```

This starts a worknode with debug enabled, which will tell it to print when it is ready and periodically print how many tasks that worknode is running.

On terminal three:

```
~/DistributedExecutorService$ ./rundemo.sh Callables localhost 5555
```

This builds and starts the Callables demo, which will use the rmiregistry found at “localhost:5555” to manage it’s worknodes. This should run and wait for the worknode to finish before returning. Look at the Windows section above for screenshots of the output of the Callables Demo. For a complete breakdown of what the Callables Demo does, see the ‘3.1 Callables Breakdown’ section below

3.0 ADDITIONAL RESOURCES

3.1 CALLABLES BREAKDOWN

in `src/demos/callables.java` the program first creates a distributed executor service, then creates a new callable `SleepyHelloWorldCall` thread. What this thread will do is sleep for 7 seconds and then return the string “I’m Done!”. We submit this thread to the executor service 6 times and save the returned futures, exactly like any other class that implements `Executor Service`. We then call `get()` on the futures, where each call blocks until the thread that is running on another terminal finishes. This execution can be seen in figure 5, where the work node goes from 0 tasks to 6. It prints that it is sleeping, then when it wakes up it prints “Hello World!” and the work node moves to 5 tasks. Each thread is returning a string when it completes, which can be seen in Figure 10 below. This is the same execution as Figure 5 but later in time when all the threads finished. The `Callables.java` program prints out the return value of each future (“I’m Done!”) before printing “Done!”.

```
Jonathan@DESKTOP-AML7NU4 ~/DistributedExecutorService (master)
$ ./rundemo.sh Callables localhost 5555
building Callables
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Running Callables with parameters localhost:5555...
[Ljava.lang.String;@4e50df2e
On return we got I'm Done!I'm Done!I'm Done!I'm Done!I
'm Done!
Done!
^ Feeling Sleepy...
~I have 3 Tasks~
~I have 3 Tasks~
~I have 3 Tasks~
Hello World!
~I have 3 Tasks~
Feeling Sleepy...
~I have 2 Tasks~
~I have 2 Tasks~
~I have 2 Tasks~
Hello World!
Feeling Sleepy...
~I have 1 Tasks~
~I have 1 Tasks~
~I have 1 Tasks~
~I have 1 Tasks~
Hello World!
~I have 0 Tasks~
~I have 0 Tasks~
~I have 0 Tasks~
~I have 0 Tasks~
~I have 0 Tasks~
~I have 0 Tasks~
```

Fig. 11: Callables Demo wrapping up

To get a larger picture of the distributed executor service, the figure on the next page shows the result of running the CallablesRNG Demo. This demo program starts a given number of threads that each return a random number from a work node. In this example there are 8 work nodes and the CallablesRNG is set to request 100 random numbers. These requests split as evenly as possible among the connected work nodes.

```
MINGW32/c/Users/Jonathan/DistributedEx...
Jonathan@DESKTOP-AML7NU4 ~/DistributedExecut
orService (master)
$ ./rundemo.sh CallablesRNG localhost 5555
100
building CallablesRNG
Note: Some input files use unchecked or unsa
fe operations.
Note: Recompile with -Xlint:unchecked for de
tails.
Running CallablesRNG with parameters Localho
st:5555...
[Ljava.lang.String;@4e50df2e
Results:
933301299 -2034394421 -1165371974 -11923355
69 -721538402 -275264663 -1514567632 -191262
1542 1409478371 1459011932 -378744248 130169
1760 -2144287850 -1647558930 -1958045867 115
5088405 -390198506 -1572766933 1966034367 20
50813573 -908891976 1745350486 1593221845 -2
29104879 -287048032 -2063879226 -1372833352
-2084817344 -1945612106 -501282382 -19936299
24 1204264133 -776128375 -1307654382 4255676
52 1086133520 669303797 707073554 1936016104
2082521647 -1147033447 -1491789905 84484742
0 1496332335 -639616770 -1147592514 -1789418
935 1529564042 1461616092 -271018240 -191299
2130 1855478164 790651850 804435754 12493278
0 1187132998 994201960 -582221803 688044895
1970128545 614609285 1352006602 495690081 -1
088640903 -1530621650 -1353791675 -190792427
8 -714818159 -494857877 -1520833773 -4188792
3 1573129032

MINGW32/c/Users/Jonathan/Distribut...
-I have 0 Tasks~
-I have 0 Tasks~
-I have 0 Tasks~
-I have 0 Tasks~
Your Random Number Is...
-I have 12 Tasks~
-1514567632!!!
-I have 12 Tasks~
Your Random Number Is...
1459011932!!!
-I have 11 Tasks~
Your Random Number Is...
-229104879!!!
-I have 10 Tasks~
Your Random Number Is...
-1745350486!!!
-I have 9 Tasks~
Your Random Number Is...
-501282382!!!
-I have 9 Tasks~
Your Random Number Is...
707073554!!!
-I have 8 Tasks~
Your Random Number Is...
-1147592514!!!
-I have 7 Tasks~
Your Random Number Is...
804435754!!!
-I have 5 Tasks~
Your Random Number Is...
1352006602!!!
-I have 5 Tasks~
Your Random Number Is...
-1520833773!!!
-I have 4 Tasks~
Your Random Number Is...

MINGW32/c/Users/Jonathan/Distribut...
Node ready: 56229ea6:15f8b31e8f3:-7fff
-I have 0 Tasks~
-I have 0 Tasks~
-I have 0 Tasks~
-I have 0 Tasks~
Your Random Number Is...
933301299!!!
-I have 13 Tasks~
Your Random Number Is...
-1165371974!!!
-I have 12 Tasks~
Your Random Number Is...
-390198506!!!
Your Random Number Is...
-I have 10 Tasks~
-2084817344!!!
-I have 10 Tasks~
Your Random Number Is...
1936016104!!!
-I have 9 Tasks~
Your Random Number Is...
-1147033447!!!
-I have 8 Tasks~
Your Random Number Is...
-1491789905!!!
-I have 7 Tasks~
1855478164!!!
Your Random Number Is...
-I have 6 Tasks~
495690081!!!
-I have 6 Tasks~
Your Random Number Is...
-1530621650!!!
-I have 5 Tasks~
Your Random Number Is...

MINGW32/c/Users/Jonathan/Distribut...
-I have 0 Tasks~
-I have 0 Tasks~
-I have 0 Tasks~
-I have 0 Tasks~
Your Random Number Is...
-1958045867!!!
-I have 13 Tasks~
Your Random Number Is...
1155088405!!!
Your Random Number Is...
-I have 11 Tasks~
-1572766933!!!
Your Random Number Is...
-1945612106!!!
-I have 10 Tasks~
Your Random Number Is...
2082521647!!!
-I have 9 Tasks~
Your Random Number Is...
-1491789905!!!
-I have 7 Tasks~
790651850!!!
Your Random Number Is...
-1088640903!!!
Your Random Number Is...
-1353791675!!!
-I have 5 Tasks~
Your Random Number Is...

MINGW32/c/Users/Jonathan/Distribut...
-I have 0 Tasks~
-I have 0 Tasks~
-I have 0 Tasks~
-I have 0 Tasks~
Your Random Number Is...
-275264663!!!
-I have 12 Tasks~
Your Random Number Is...
1409478371!!!
-I have 11 Tasks~
Your Random Number Is...
1593221845!!!
-I have 10 Tasks~
Your Random Number Is...
-287048032!!!
-I have 8 Tasks~
1086133520!!!
-I have 8 Tasks~
Your Random Number Is...
-1789418935!!!
-I have 7 Tasks~
Your Random Number Is...
1461616092!!!
-I have 6 Tasks~
Your Random Number Is...
-418879237!!!
-I have 4 Tasks~
Your Random Number Is...

MINGW32/c/Users/Jonathan/DistributedExec...
Debug Activated
-I have 0 Tasks~
Node ready: -29699207:15f8b31f6a9:-7fff
-I have 0 Tasks~
-I have 0 Tasks~
Your Random Number Is...
-2034394421!!!
-I have 12 Tasks~
Your Random Number Is...
-1192335569!!!
Your Random Number Is...
-I have 10 Tasks~
-908891976!!!
-I have 10 Tasks~
Your Random Number Is...
1204264133!!!
-I have 9 Tasks~
Your Random Number Is...
-1307654382!!!
-I have 8 Tasks~
Your Random Number Is...
-639616770!!!
-I have 6 Tasks~
1187132998!!!
-I have 6 Tasks~
Your Random Number Is...
-582221803!!!
-I have 5 Tasks~
Your Random Number Is...
-494857877!!!
-I have 4 Tasks~
Your Random Number Is...
```

Fig. 12: CallablesRNG Demo with 8 worknodes

3.2 INSTALLING GIT

Head to <https://git-scm.com/downloads> and download the Git for Windows. When installing, be sure to keep defaults. Select 'Use Git from the Windows Command Prompt' when asked about adjusting your PATH environment.

3.3 INSTALLING JRE

Be sure to uninstall any previous version of Java before upgrading. JRE 8 can be found at <http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html>.

Select Windows x64 Offline version, and hit the .exe. after it finishes downloading. Install with all default settings. Open up your bash and run 'java -version'. If you get a command-not-found error, continue on to the next section '3.4 Installing JDK' and skip to the second paragraph to see how to update your environment variables.

3.4 INSTALLING JDK

Be sure to uninstall any previous version of Java before upgrading. JDK 8 can found at <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>.

It is recommended to download Windows x64 version, not the x86 version.

Java SE Development Kit 8 Downloads
Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- [Java Developer Newsletter](#): From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- [Java Developer Day](#) hands-on workshops (free) and other events
- [Java Magazine](#)

JDK 8u151 checksum
JDK 8u152 checksum

Java SE Development Kit 8u151
You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.9 MB	jdk-8u151-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.85 MB	jdk-8u151-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.95 MB	jdk-8u151-linux-i586.rpm
Linux x86	183.73 MB	jdk-8u151-linux-i586.tar.gz
Linux x64	166.1 MB	jdk-8u151-linux-x64.rpm
Linux x64	180.95 MB	jdk-8u151-linux-x64.tar.gz
macOS	247.06 MB	jdk-8u151-macosx-x64.dmg
Solaris SPARC 64-bit	140.06 MB	jdk-8u151-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.32 MB	jdk-8u151-solaris-sparcv9.tar.gz
Solaris x64	140.65 MB	jdk-8u151-solaris-x64.tar.Z
Solaris x64	97 MB	jdk-8u151-solaris-x64.tar.gz
Windows x86	198.04 MB	jdk-8u151-windows-i586.exe
Windows x64	205.95 MB	jdk-8u151-windows-x64.exe

Fig. 13: screenshot of the JDK download page

Click the exe and allow the jdk to download with default settings. Afterwards Java should be set up, but check by pulling up your bash and running 'java -version' and 'javac -version'.

```
Jonny@DESKTOP-K5LNIC4 MINGW64 ~  
$ java -version  
java version "1.8.0_151"  
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)  
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)  
  
Jonny@DESKTOP-K5LNIC4 MINGW64 ~  
$ javac -version  
bash: javac: command not found
```

Fig. 14: checking with -version option

If one or both could not be found, you will need to add them to your path. Default settings should have placed the jdk and jre in C:\Program Files\Java

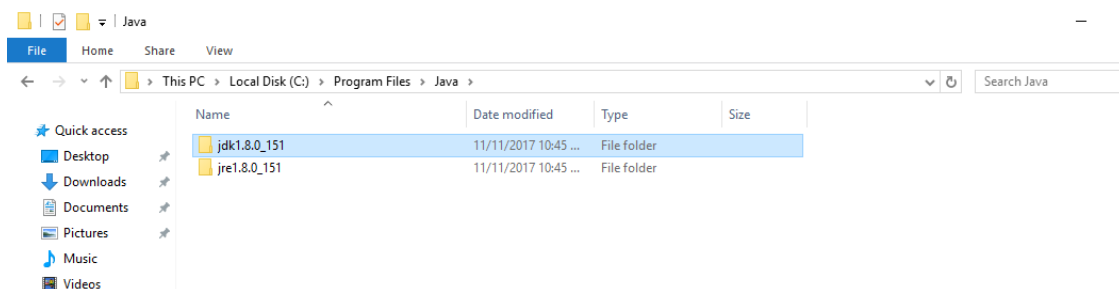
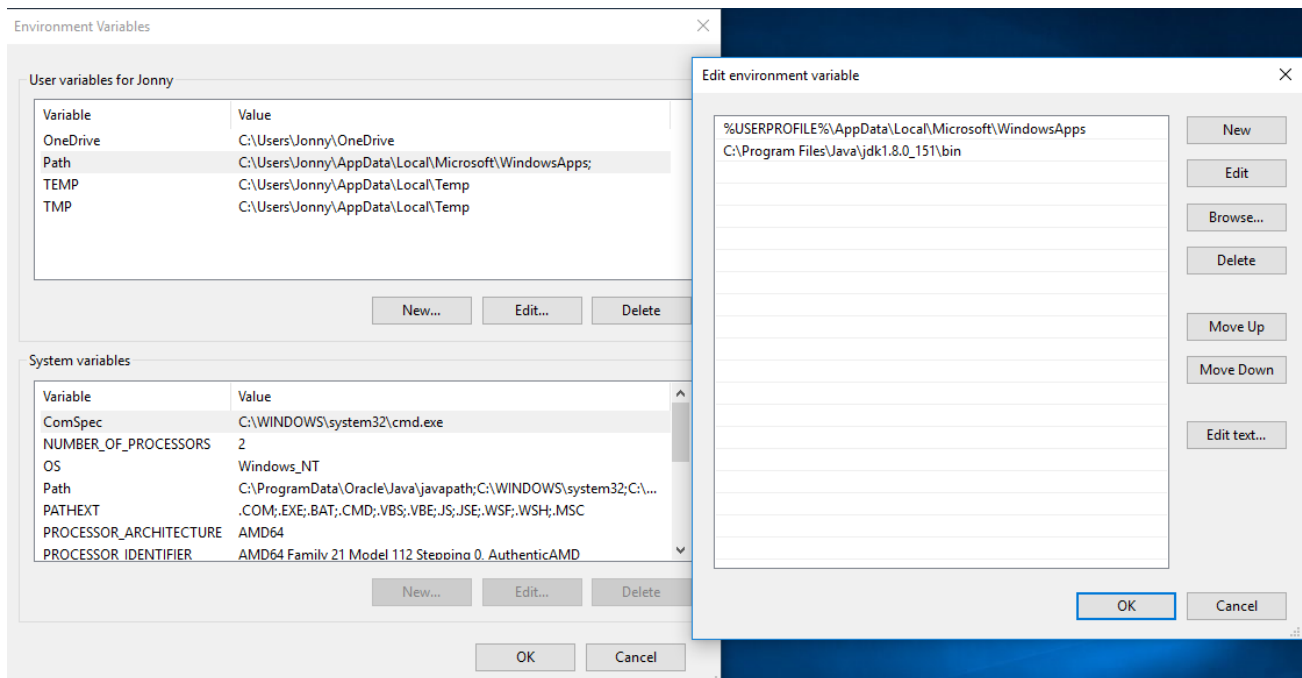


Fig. 15: Windows Java location

Go to the Windows search bar and search for “environment variables”. Click on the “Edit the environment variables for your account” to bring up the Environment Variables window. Locate the Path variable and hit edit. If your javac did not work, you will need to add ‘C:\Program Files\jdk1.8.0_151\bin’ (or wherever your JDK is located) to your path using the ‘New’ button. If your java did not work, you will need to also add ‘C:\Program Files\jre1.8.0_151\bin’ (or wherever your JRE is located). Hit ok twice to confirm changes. You will need to close and reopen your bash to see the changes.



```
Jonny@DESKTOP-K5LNIC4 MINGW64 ~  
$ javac -version  
javac 1.8.0_151  
  
Jonny@DESKTOP-K5LNIC4 MINGW64 ~  
$ java -version  
java version "1.8.0_151"  
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)  
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)
```

Fig. 16: Windows Environment Variables Window