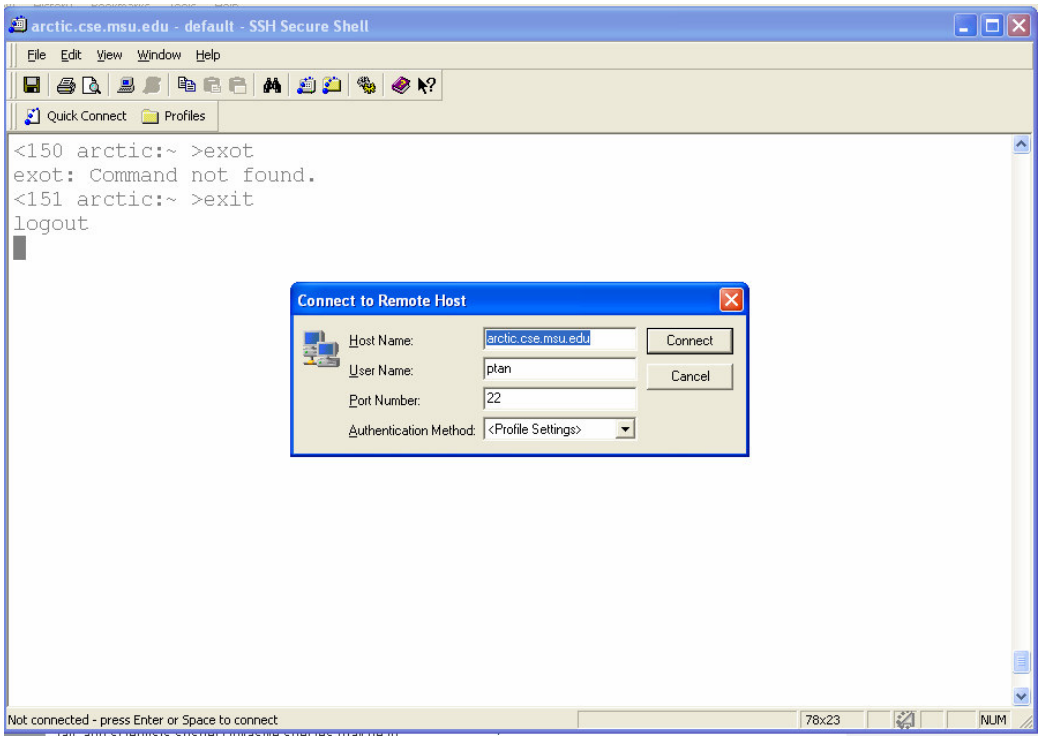
Getting Started with Java Programming on Linux

Here is a step-by-step process for writing a Java program on Linux.

STEP 1: Connect to one of the CSE machines (example: pacific.cse.msu.edu)



STEP 2: Check to make sure java is installed on the machine

```
<5 hadoop:~ >where java
/usr/bin/java
/usr/bin/X11/java
```

STEP 3: Make sure the environment variables are setup correctly

```
<122 hadoop:~ >echo $CLASSPATH
/usr/lib/jvm/java-6-openjdk/lib:.
<123 hadoop:~ >
<123 hadoop:~ >echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games:/usr/local/hadoop/bin:.
```

If the PATH and CLASSPATH variables are not set, you need to set them first before you proceed further. Read the instructions provided on the following web page http://docs.oracle.com/javase/tutorial/essential/environment/paths.html on how to set up the variables.

STEP 4: Open a text editor (vi, pico, xemacs, etc), type in your code, and save the file as <filename>.java. Make sure the filename has an extension .java. The following is an example of a Java program to count the number of words given an input document. First, let's take a look at the data file itself.

```
<48 hadoop:~ >cat document.txt
This is a sample document to be used as an input document file for a
Java program written to count the number of times each word appears
in the document.

<49 hadoop:~ >cat wordCount.java
import java.io.*;
import java.util.*;

public class wordCount  {
    public static void main(String [] args) throws IOException {

        Map<String,Integer> counts = new HashMap<String, Integer>();
        FileReader input = new FileReader(args[0]);
        BufferedReader bufRead = new BufferedReader(input);
        String cline = null;

        while ( (cline = bufRead.readLine()) != null) {
             StringTokenizer itr = new StringTokenizer(cline);

             while (itr.hasMoreTokens()) {
                String term = itr.nextToken();
                Integer freq = counts.get(term);

                if (freq == null) {
                        counts.put(term, 1);
                }
                else {
                        counts.put(term, freq + 1);
                }
            }
        }
        input.close();

        Iterator iter = counts.keySet().iterator();
        while (iter.hasNext()) {
            String key = iter.next().toString();
            String value = counts.get(key).toString();
            System.out.println(key + "\t" +  value);
        }
    }
}
```

STEP 5: To compile the java code, type javac followed by the name of the source file.

```
<51 hadoop:~ >javac wordCount.java
```

After compilation, a *.class file will be created.

```
<63 hadoop:~ >ls
wordCount.class  wordCount.java
```

STEP 6: To execute the program, java followed by the name of the file (without its extension) and its command line arguments.

```
<55 hadoop:~ >java wordCount document.txt
to      2
count   1
for     1
used    1
This    1
Java    1
appears 1
of      1
document        2
times   1
file    1
written 1
be      1
sample  1
is      1
document.       1
a       2
as      1
number  1
the     2
in      1
input   1
program 1
an      1
word    1
each    1
```