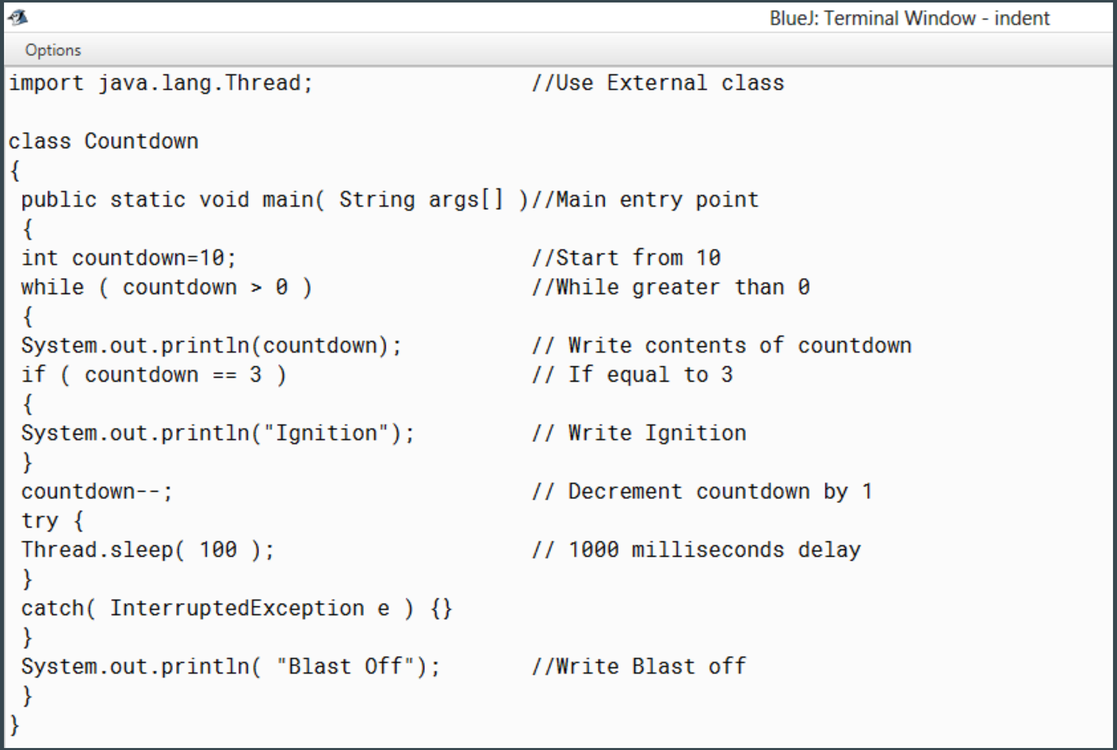
CI101 - Mini Project 1

By James Beck McGovern, Tyler Daniel, Khemsang Chemjong and C. S. Bartels

Our Program:

The task was to create a program which would indent a set of code, below is our result. As you can see, we were successful in our endeavour.



```
import java.lang.Thread;                 //Use External class

class Countdown
{
 public static void main( String args[] )//Main entry point
 {
 int countdown=10;                       //Start from 10
 while ( countdown > 0 )                 //While greater than 0
 {
 System.out.println(countdown);          // Write contents of countdown
 if ( countdown == 3 )                   // If equal to 3
 {
 System.out.println("Ignition");         // Write Ignition
 }
 countdown--;                            // Decrement countdown by 1
 try {
 Thread.sleep( 100 );                    // 1000 milliseconds delay
 }
 catch( InterruptedException e ) {}
 }
 System.out.println( "Blast Off");       //Write Blast off
 }
}
```

Why we believe that our program is deserving of a C grade:

We believe we have not only met ‘D’ grade criteria, but ‘C’ grade criteria as well:

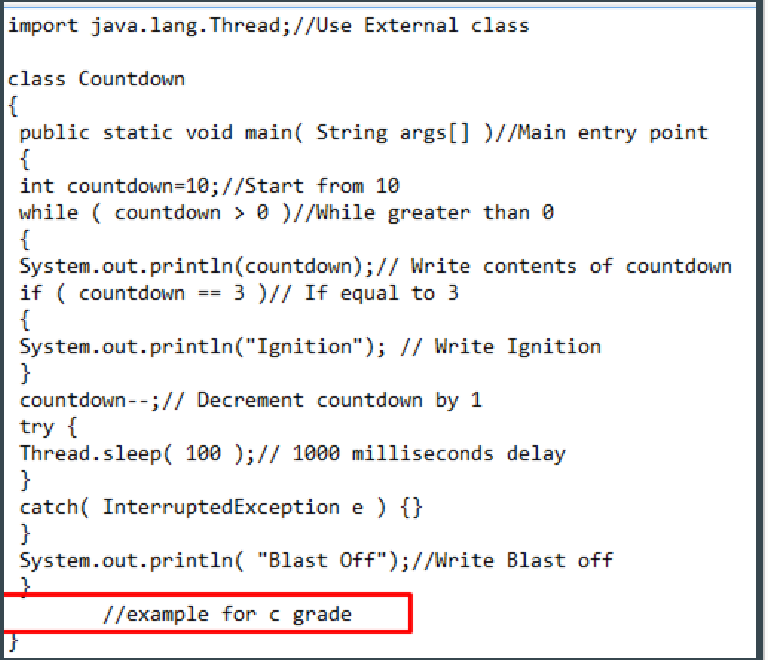
We have succeeded in, these tasks for ‘D’ grade:

- Indenting all ‘//’ comments in a program, so that all comments appear vertically underneath each other
- The contents of the line are not changed
- All non-significant spaces have been removed

We have succeeded in, these tasks for ‘C’ grade:

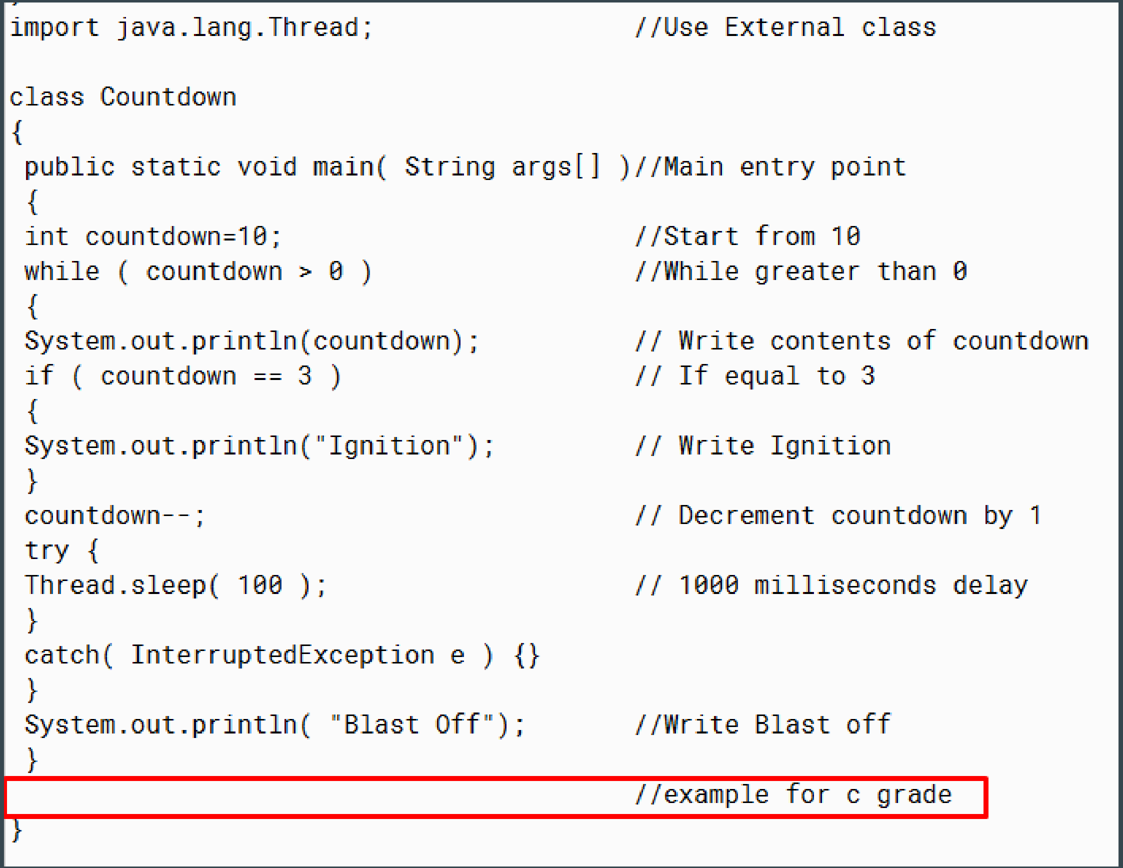
- Allowing blank java strings. i.e. Having a ‘//’ comment but no Java code before this.

Evidence for a C grade



```
import java.lang.Thread;//Use External class

class Countdown
{
 public static void main( String args[] )//Main entry point
 {
 int countdown=10;//Start from 10
 while ( countdown > 0 )//While greater than 0
 {
 System.out.println(countdown);// Write contents of countdown
 if ( countdown == 3 )// If equal to 3
 {
 System.out.println("Ignition"); // Write Ignition
 }
 countdown--;// Decrement countdown by 1
 try {
 Thread.sleep( 100 );// 1000 milliseconds delay
 }
 catch( InterruptedException e ) {}
 }
 System.out.println( "Blast Off");//Write Blast off
 }
        //example for c grade
}
```



```
import java.lang.Thread;                        //Use External class

class Countdown
{
 public static void main( String args[] )//Main entry point
 {
 int countdown=10;                              //Start from 10
 while ( countdown > 0 )                        //While greater than 0
 {
 System.out.println(countdown);                 // Write contents of countdown
 if ( countdown == 3 )                          // If equal to 3
 {
 System.out.println("Ignition");                // Write Ignition
 }
 countdown--;                                   // Decrement countdown by 1
 try {
 Thread.sleep( 100 );                           // 1000 milliseconds delay
 }
 catch( InterruptedException e ) {}
 }
 System.out.println( "Blast Off");              //Write Blast off
 }
                                                //example for c grade
}
```

Ensuring the program worked

Firstly, we created pseudo code for the main logic of the program to make sure that we had a concrete understanding of what we needed to achieve.

Once the code was implemented and each method required was complete we ensured the program worked by testing it after each addition/change was made.

We did this until we had the minimum outcome required and worked our way from there.

Testing was a significant part in this process as without it we wouldn’t know which areas to change.

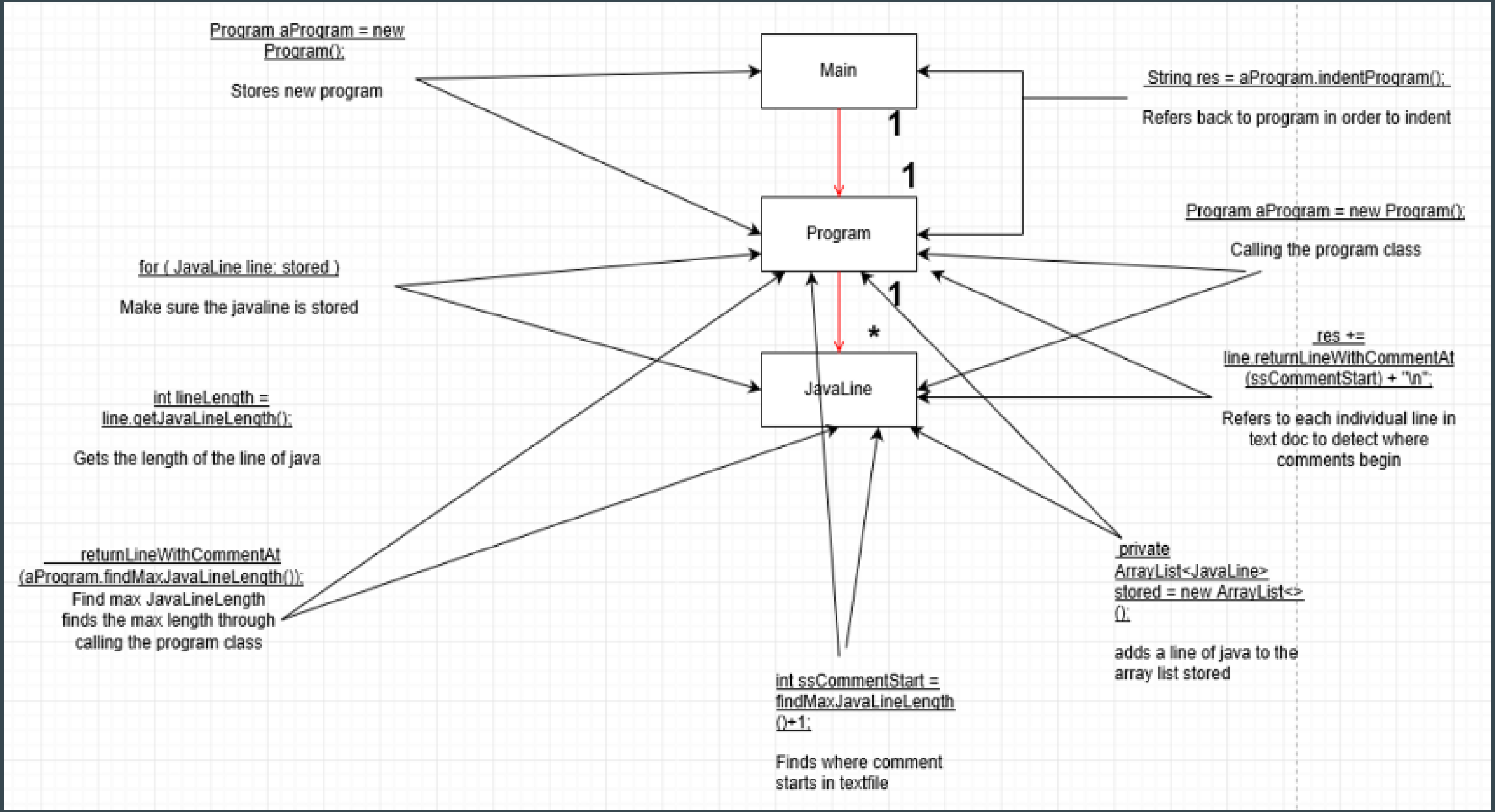
What did we learn?

We learnt many things during the creation of this program. The bulk of which was to make sure that we understand what is being asked of us before attempting to write the program. This way it makes writing the actual program a lot easier.

Furthermore, we learnt more about how class can call each other and how methods are linked.

Basing off this, an object oriented approach helped somewhat with the program as it allowed for different classes and methods to link together and make the individual code in each method shorter than it would otherwise be.

UML Diagram


Main
Program
JavaLine
1
1
1
*
Program aProgram = new Program();
Stores new program
String res = aProgram.indentProgram();
Refers back to program in order to indent
Program aProgram = new Program();
Calling the program class
for (JavaLine line: stored)
Make sure the javaline is stored
res += line.returnLineWithCommentAt (ssCommentStart) + "\n";
Refers to each individual line in text doc to detect where comments begin
int lineLength = line.getJavaLineLength();
Gets the length of the line of java
returnLineWithCommentAt (aProgram.findMaxJavaLineLength());
Find max JavaLineLength finds the max length through calling the program class
private ArrayList<JavaLine> stored = new ArrayList<> ();
adds a line of java to the array list stored
int ssCommentStart = findMaxJavaLineLength ()+1;
Finds where comment starts in textfile

What we could have done but did not do to improve our program to reach higher grades:

Unfortunately, we were unable to reach the ‘B’, ‘A’ and ‘A*’ grades, this is because we did not:

- Allow for the character pair ‘//’ to occur within a string and also within a ‘//’ comment
- Correctly indent the Java Programme so that the code inside each ‘{ }’ block is indented by two spaces and further adjust for a potential position change of the ‘//’.

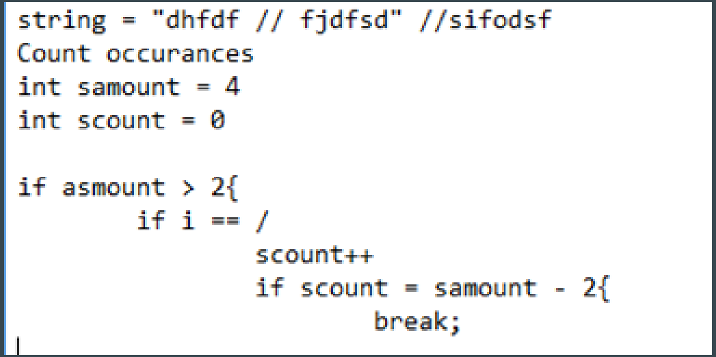
Suggestions of code to reach higher grades:

Although previously unable to reach grades higher than ‘C’ below are our ideas of which could have possibly fulfilled the requirements for ‘B’, ‘A’ and ‘A*’ grades.

Add character pairs throughout the string, presumably by using:

- Nested if statements
- Counts to identify how many ‘/’ used in the string

Potential Pseudo Code:



```
string = "dhfdf // fjdfsd" //sifodsf
Count occurances
int samount = 4
int scount = 0

if asmount > 2{
        if i == /
                scount++
                if scount = samount - 2{
                        break;
```

Suggestions of code to reach higher grades (2):

Indent the code by two spaces inside each ‘{ }’, presumably by doing:

- Using a switch statement
- Using ‘input java.util.regex’ which would identify specific characters inside the string, and would easily allow the addition of two spaces inside the ‘{ }’ block

Fin.