CS620c

What happens when you write a custom made method named writeString

Joe Duffin

Callan Building Room 2.108

Email: Joseph.Duffin@nuim.ie

Another method with two parameters.

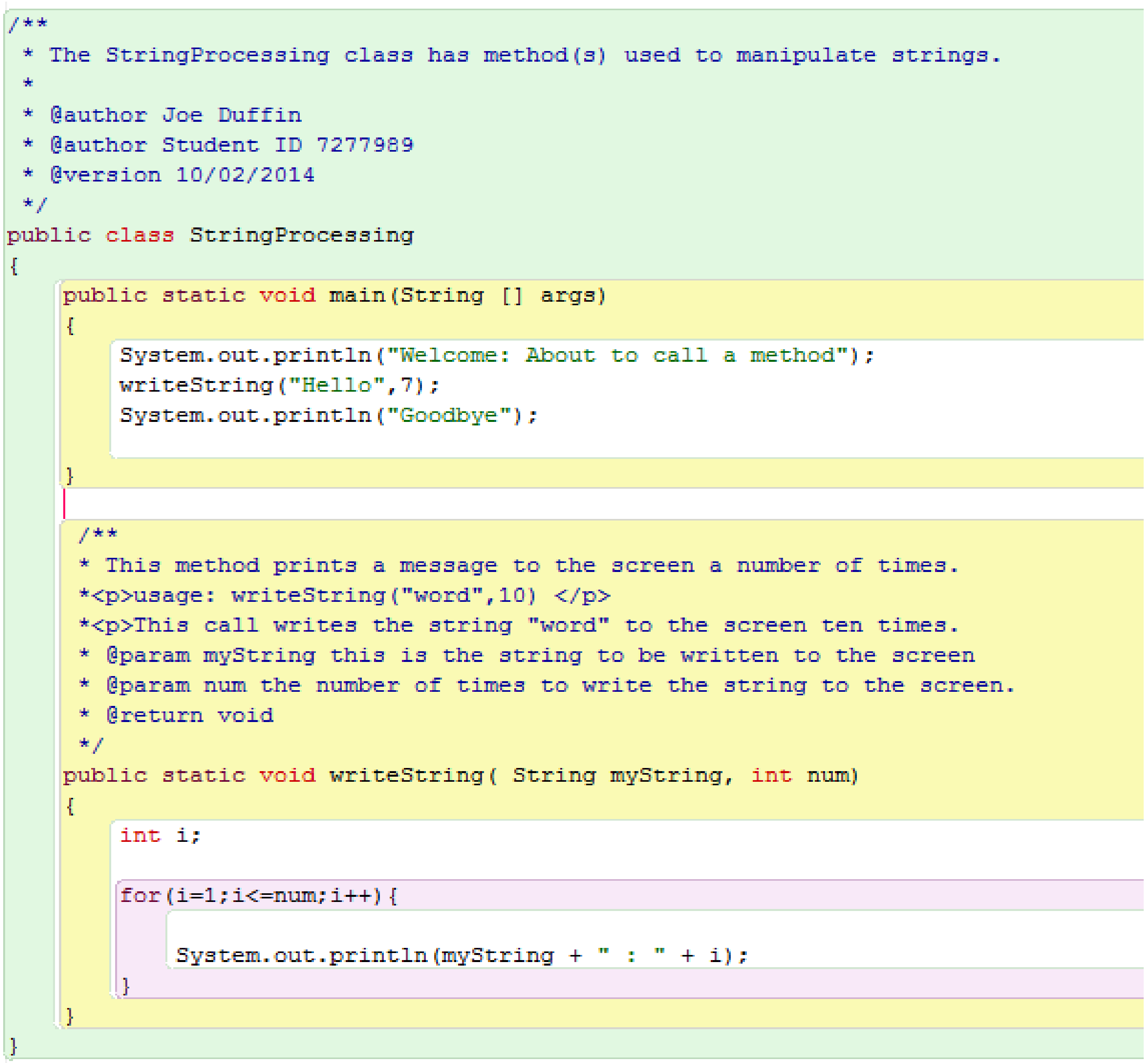
Q1: Write a method that takes a string and a value and then writes the string to the screen a number of times defined by the value. For example if the String was “hello” and the value was 7 then the method would write out “hello” seven times.

From the information in the question above we can tell that the method would have a **signature** something like this below:

```
public static void writeString( String myString, int num)
```

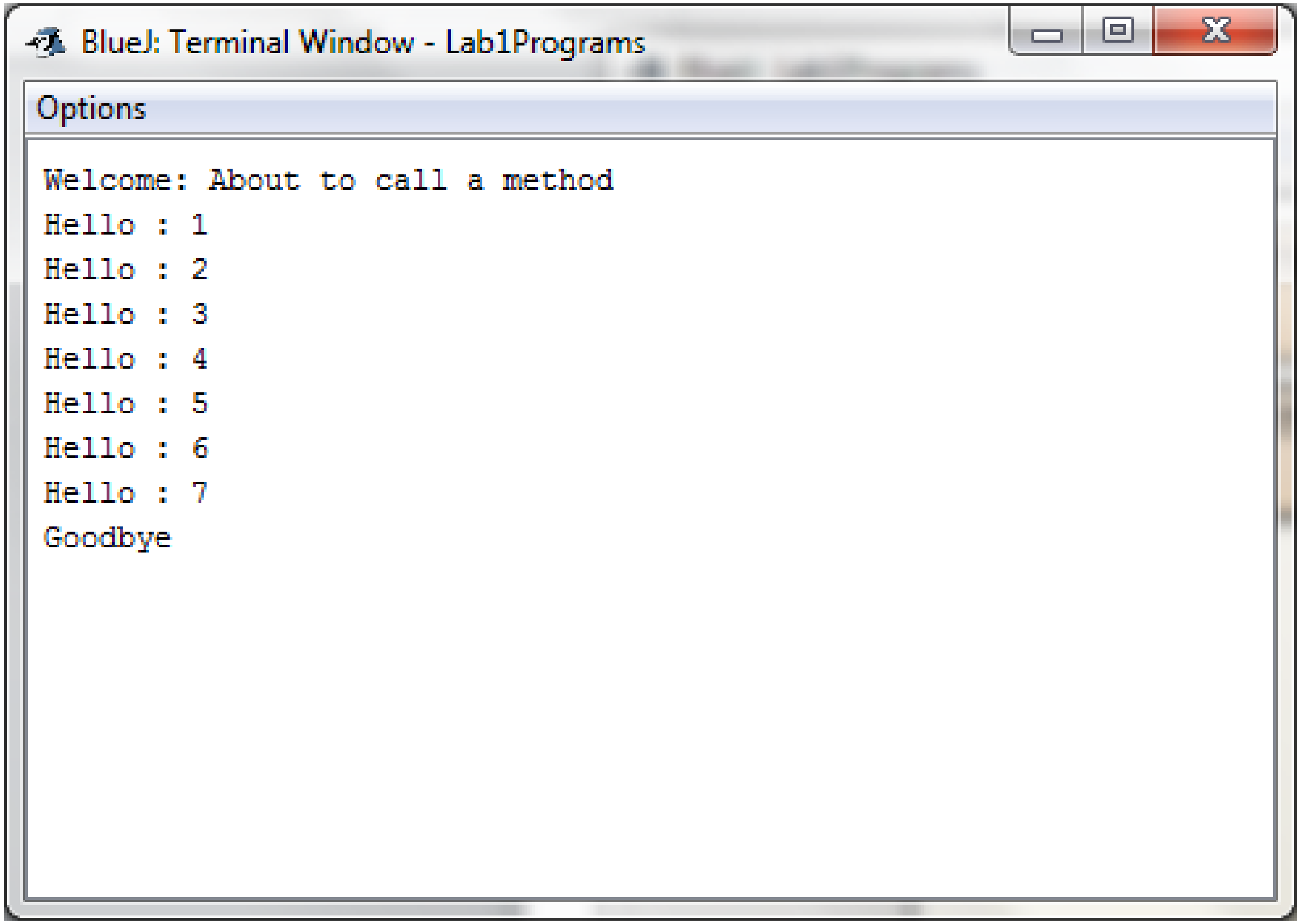
The body of the code would take the values for myString and num and use them to write myString to the screen a number of times defined by num.

The solution to this is on the next page.



```java
/**
 * The StringProcessing class has method(s) used to manipulate strings.
 *
 * @author Joe Duffin
 * @author Student ID 7277989
 * @version 10/02/2014
 */
public class StringProcessing
{
    public static void main(String [] args)
    {
        System.out.println("Welcome: About to call a method");
        writeString("Hello",7);
        System.out.println("Goodbye");

    }

     /**
     * This method prints a message to the screen a number of times.
     *<p>usage: writeString("word",10) </p>
     *<p>This call writes the string "word" to the screen ten times.
     * @param myString this is the string to be written to the screen
     * @param num the number of times to write the string to the screen.
     * @return void
     */
    public static void writeString( String myString, int num)
    {
        int i;

        for(i=1;i<=num;i++){

            System.out.println(myString + " : " + i);
        }
    }
}
```

Output of the StringProcessing class when its main method is run.



```
Welcome: About to call a method
Hello : 1
Hello : 2
Hello : 3
Hello : 4
Hello : 5
Hello : 6
Hello : 7
Goodbye
```

What happens when we call the method writeString?

1 A memory resource called a **stack frame or run time stack** is created for the call to the method writeString("hello", 7)

2 Memory space is set aside within this **stack frame** for the formal parameters **String myString** and **int num (**and the local variable **int i** step 5**)**

3 The **return point or address** is stored in the **stack frame**. This is the point at which the program continues to execute after the method is finished running.

4 The values in the actual parameter s ("hello", 7) are **<u>copied</u>** into the formal parameters **"hello"** is copied into **myString** and **7** is copied into **num.**

5 Space is set aside for the local variable **int i**

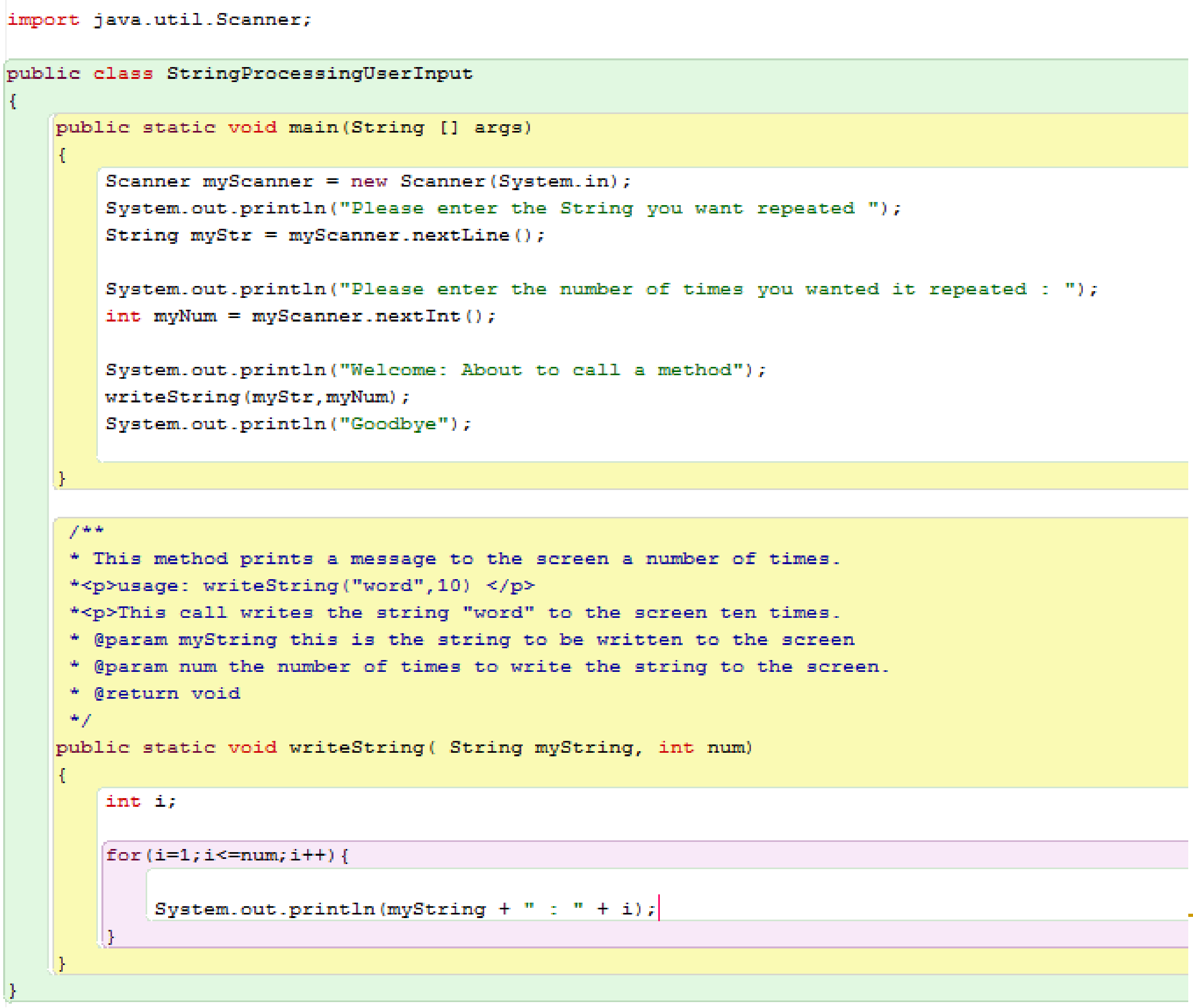
6 The code statements which were written in the body of the method are now executed, in this case a for loop prints to the word "hello" to the screen 7 times.

7 When the loop is finished executing the flow of control of the program returns to the point at which the method **writeString** was called using the **return address** stored in the **stack frame**.

8 Finally the stack frame is **"destroyed"** or returned to the system to be reused. The Formal parameters **String myString** and **int num** as well as the local variable **int i** are no longer available in your program

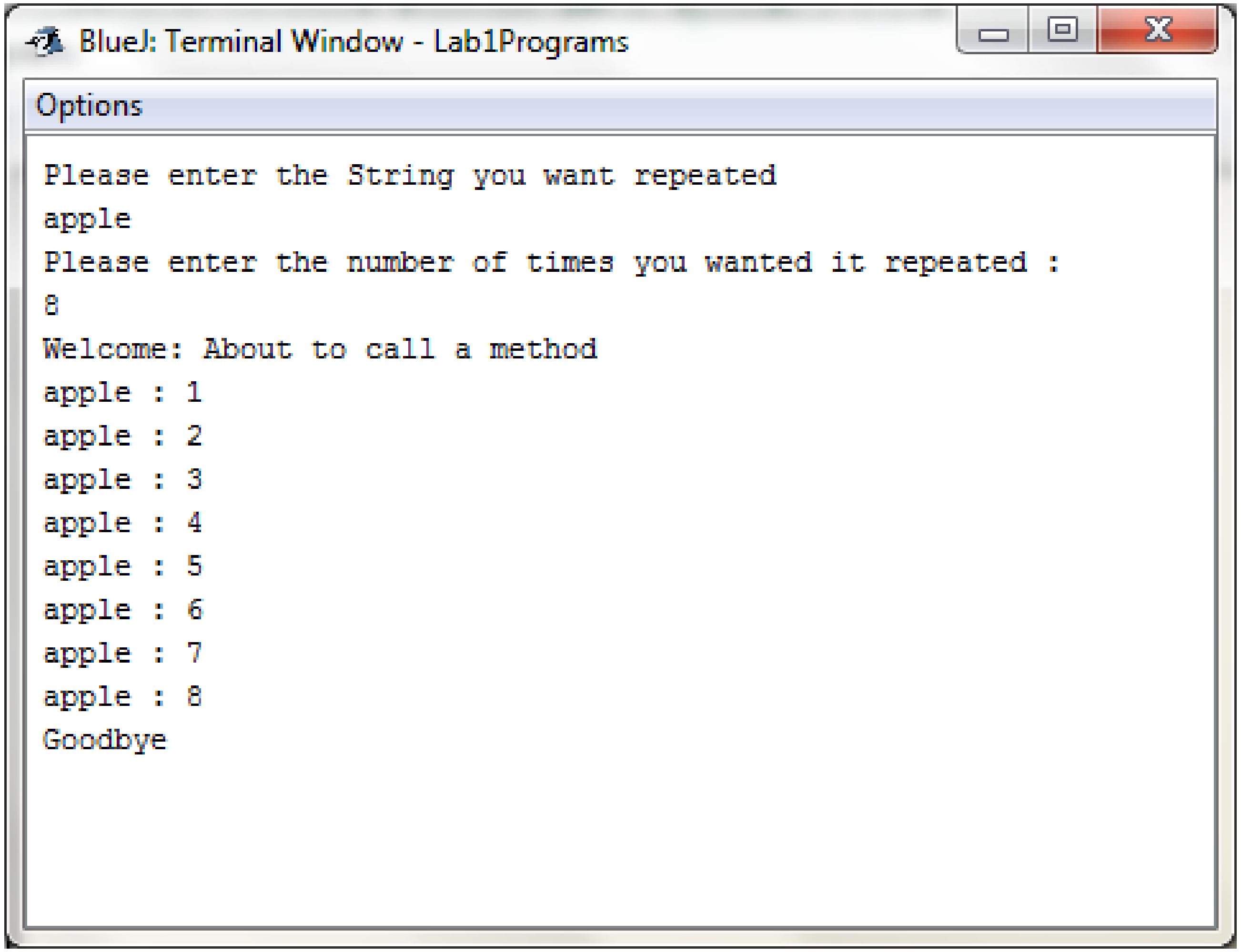
Your program then continues executing beyond the method call onto the next line of code.

Adapting the program StringProcessing to take user input.



```
import java.util.Scanner;

public class StringProcessingUserInput
{
    public static void main(String [] args)
    {
        Scanner myScanner = new Scanner(System.in);
        System.out.println("Please enter the String you want repeated ");
        String myStr = myScanner.nextLine();

        System.out.println("Please enter the number of times you wanted it repeated : ");
        int myNum = myScanner.nextInt();

        System.out.println("Welcome: About to call a method");
        writeString(myStr,myNum);
        System.out.println("Goodbye");

    }

     /**
     * This method prints a message to the screen a number of times.
     *<p>usage: writeString("word",10) </p>
     *<p>This call writes the string "word" to the screen ten times.
     * @param myString this is the string to be written to the screen
     * @param num the number of times to write the string to the screen.
     * @return void
     */
    public static void writeString( String myString, int num)
    {
        int i;

        for(i=1;i<=num;i++){

            System.out.println(myString + " : " + i);
        }
    }
}
```

Output of StringProcessingUserInput.



```
Please enter the String you want repeated
apple
Please enter the number of times you wanted it repeated :
8
Welcome: About to call a method
apple : 1
apple : 2
apple : 3
apple : 4
apple : 5
apple : 6
apple : 7
apple : 8
Goodbye
```

Recommendation

It would help you get a very good understanding of methods and how they are executed (or run) in your program if you write a description similar to the previous page (about the writeString method), for between 5 and 10 of your own methods that you write in this module.

Exercise for you

Q1: Write a method named **squareNum** that takes a single integer parameter and returns an integer value. This method calculates the square of the number passed to it. i.e. squareNum(2) → 4 , squareNum(5) → 25

From the information in the question above we can tell that the method would have a **signature** something like this below:

```
public static int squareNum( int number )
```

How many formal parameter does this method take?
What is the return type of this method?
Does the method have a meaningful name?
Place your finger on the formal parameter list of this method.

When you have written the method squareNum use it in your main method to calculate the square value of all numbers from 1 to 20 and print these to the screen. **Tip 1**: use a loop **Tip 2:** do **NOT** alter the body of your method.

Write out what happens when the your method **squareNum** is called in the main method of your program. Use the example of the writeString method to guide you.