Lab 1 – Output to the console

Exercise 1

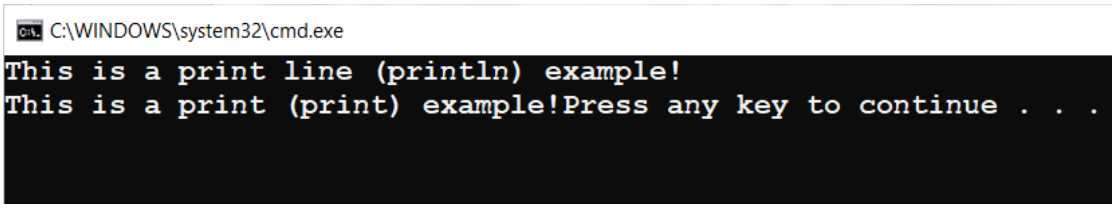
Create a java program in a file called "Exercise1.java" that outputs text to the screen.

1. Ensure that the filename is called "Exercise1.java"

2. Input the following code:

```
1  public class Exercise1
2  {
3      public static void main(String[] args)
4      {
5          System.out.println("This is a print line (println) example!");
6          System.out.print("This is a print (print) example!");
7      }
8  }
```

3. Your output should be similar to as shown below:



Exercise 2

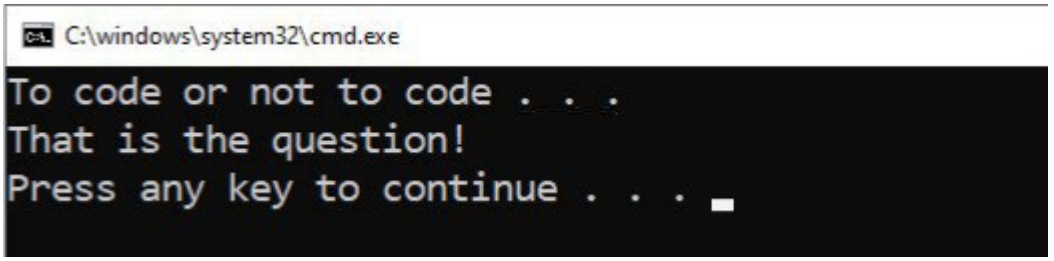
Create a java program in a file called “Exercise2.java” that outputs text to the screen.

4. Ensure that the filename is called “Exercise2.java”

5. Input the following code:

```
1  public class Exercise2
2  {
3      public static void main(String[] args)
4      {
5          System.out.println("To code or not to code . . . ");
6          System.out.println("That is the question!");
7      }
8  }
```

6. Compile and view the output. The result should be similar to as shown:



7. Amend your code so that it uses System.out.print (remove the “**ln**” after print, see below):

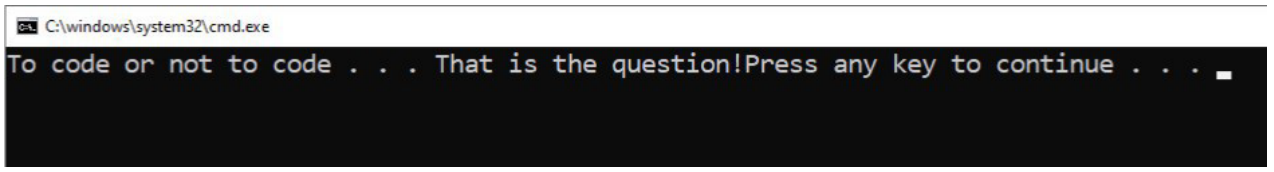
From this:

```
System.out.println("To code or not to code . . . ");
System.out.println("That is the question!");
```

To this:

```
System.out.print("To code or not to code . . . ");
System.out.print("That is the question!");
```

Compile and run your program after making this change. You should note that the 2 lines now output on the console on a single line:



What's the difference between .print() and .println()?

- println("Hello") outputs the text "Hello" and moves the cursor to a new line
- print("Hello") instead outputs just the text "Hello", but does not move the cursor to a new line. Hence, subsequent printing instructions will print on the same line. The println() method can also be used without parameters or any text inside, to position the cursoron the next line.

Exercise 3

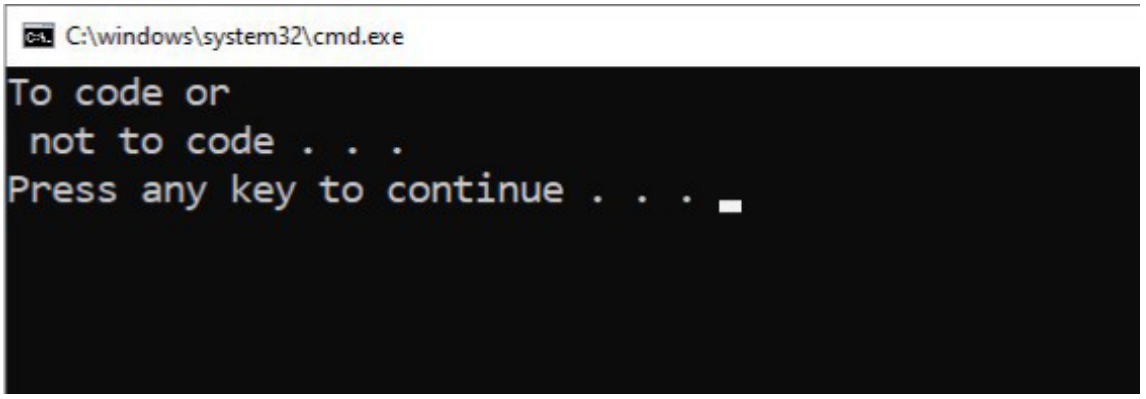
Create a java program in a file called "Exercise3.java". In this exercise you will begin to use some of the special characters (known as escape sequences) shown below.

Escape Sequence	Description
\t	Inserts a tab in the text at this point.
\n	Inserts a newline in the text at this point.

1. Ensure that the filename is called "exercise3.java". Write a program that includes the following line of code:

```
System.out.println("To code or\n not to code . . . ");
```

   Compile and run your code. The output should display on 2 lines, where the "\n" was used to break the single line into 2:

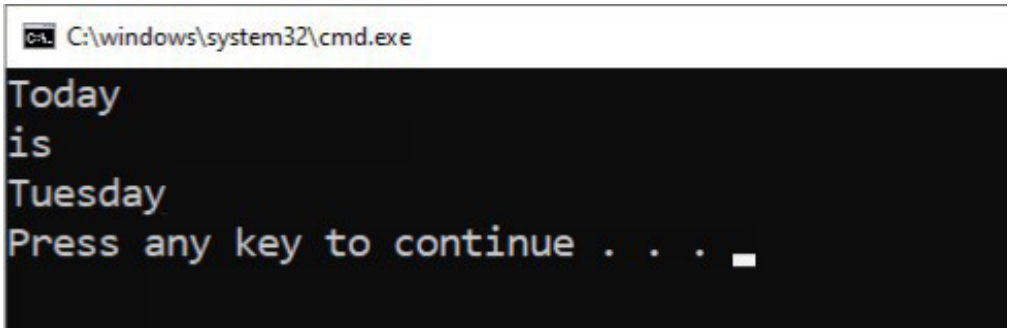


Note that the escape sequence "\n" will force what follows output on to a new line.

Exercise 4

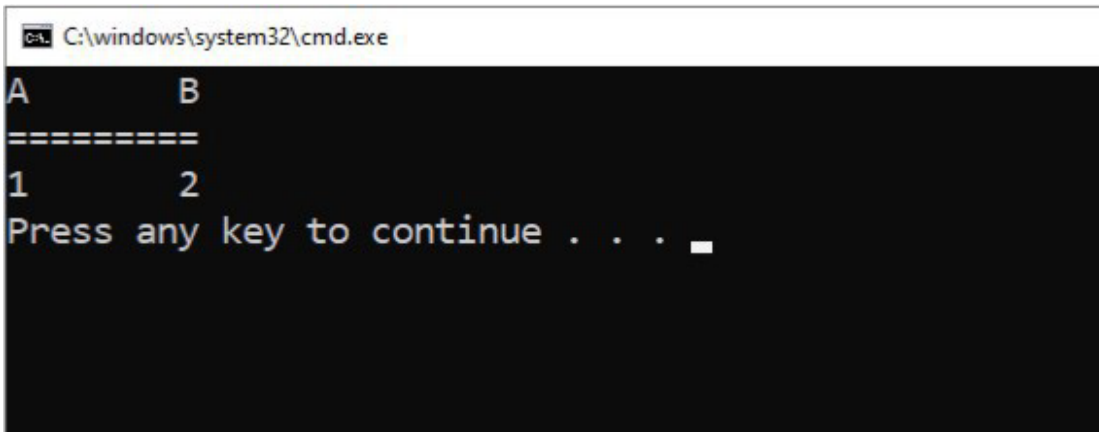
Create a java program in a file called "Exercise4.java" *using escape sequences* as required.

The following output should be produced using only one line (statement):



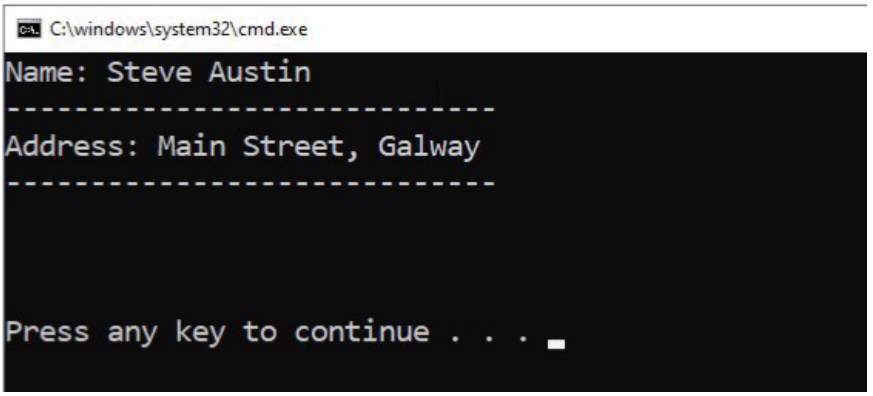
Exercise 5

Create a java program in a file called "Exercise5.java" which uses one statement to print multiple lines of data. You may consider using a tab escape sequence (\t). The following output should be produced using only one line (statement):

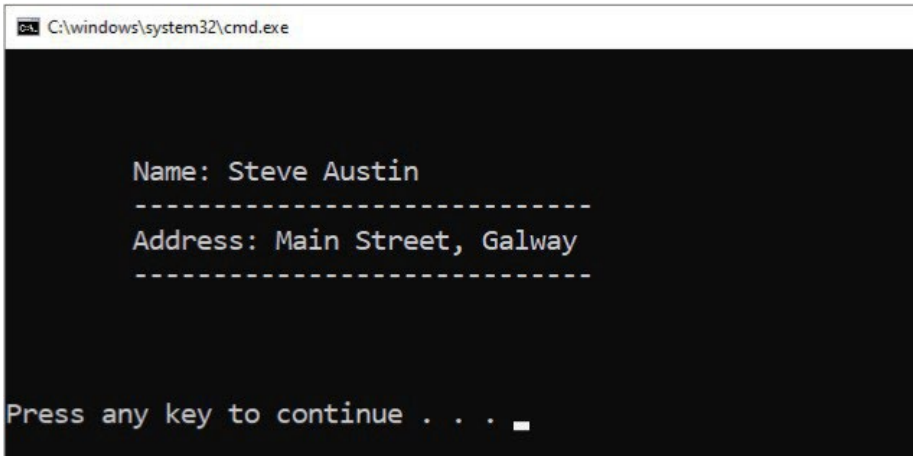


Exercise 6

Create a java program in a file called "Exercise6.java" which uses exactly 2 statements to print multiple lines of data. Your program should produce the following output:

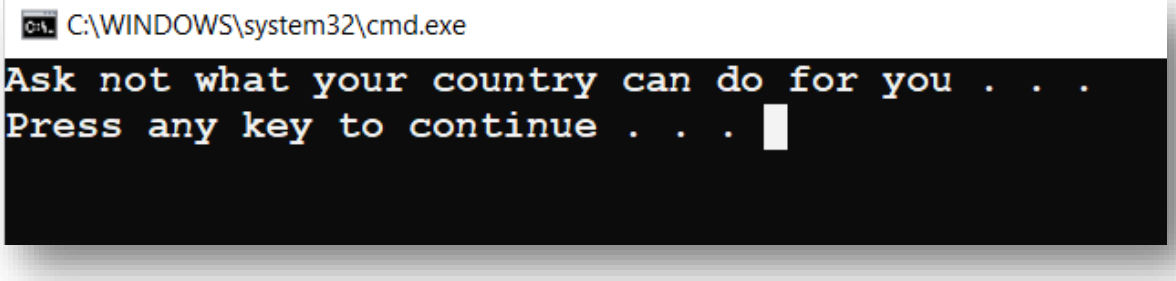


Amend your code so that the following output is produced (still using only 2 statements):



Exercise 7

Create a java program in a file called "Exercise7.java" which uses exactly 7 statements to produce the output shown below:



Exercise 8

You have been provided with the following Java code. Input the code as shown,compile and run. Make sure that the program executes successfully.

```
1  public class MyJavaProgram {
2      public static void main(String[] args){
3          System.out.println("Looking for trouble!");
4      }
5  }
```

In this exercise you will remove parts of the code and recompile. The purpose of this isto become familiar with error messages that you may come across from time to time inthe development environment, so that you see what error messages the compiler produces.

Sometimes the compiler can be helpful and tell you exactly what is wrong, and all youhave to do is fix it. But sometimes the error messages are misleading. Over time you will develop a sense for when you can trust the compiler and when you have to figurethings out yourself.

Using the program “MyJavaProgram” above, try out each of the following errors.

After you make each change, compile the program, read the error message (if there is one), and then fix the error.

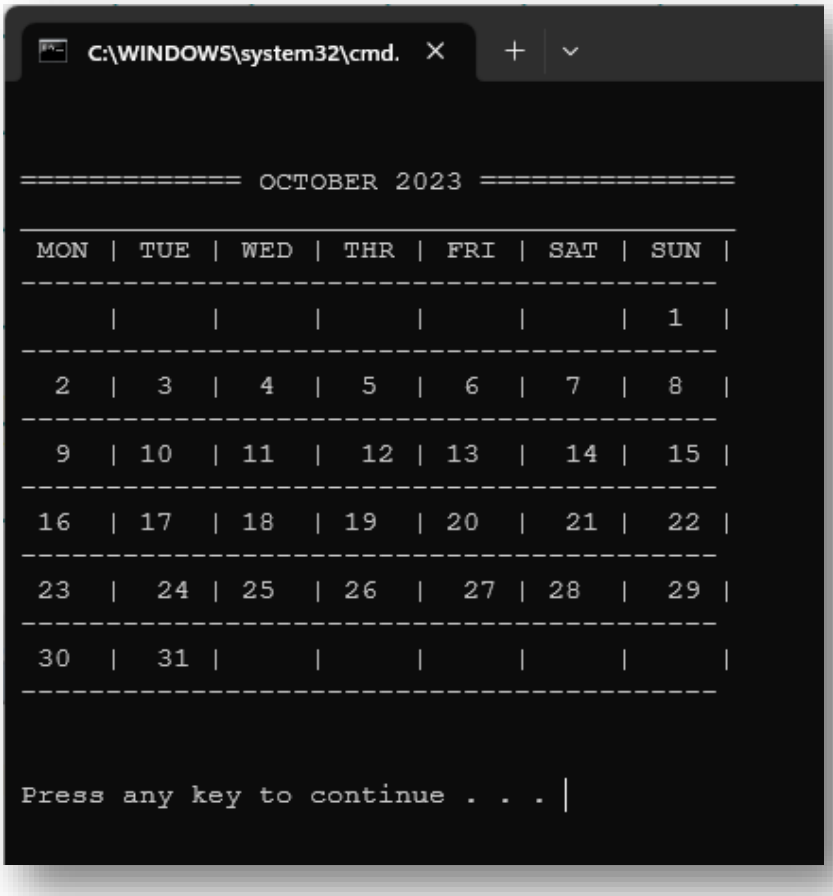
Try the following;

1. Remove one of the opening curly braces.
2. Remove one of the closing curly braces.
3. Instead of main, write mian.
4. Remove the word static.
5. Remove the word System.
6. Replace println with Println (with a capital P).
7. Replace println with print.
8. Delete one bracket.
9. Add an extra bracket.

Exercise 9

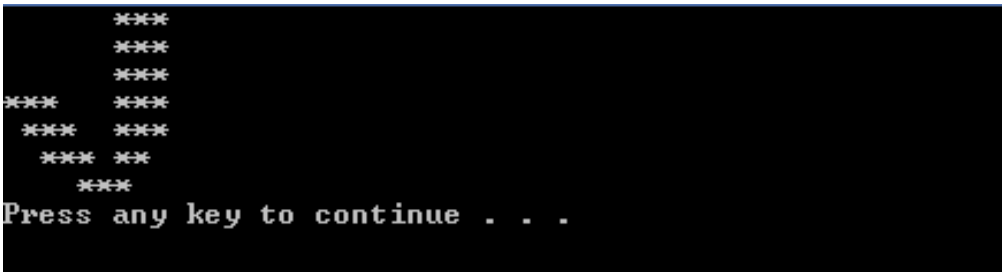
Create a java program in a file called “October.java”.

Your program should produce the following output:



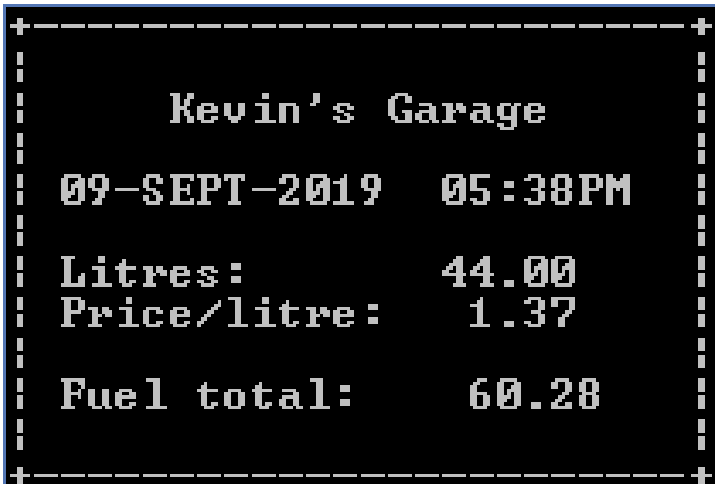
Exercise 10

Create a java program called “J”. This program should print to the screen a large J, made up of the character “*”. You can use print or println for this exercise, whichever you prefer. An example of the output is as follows:



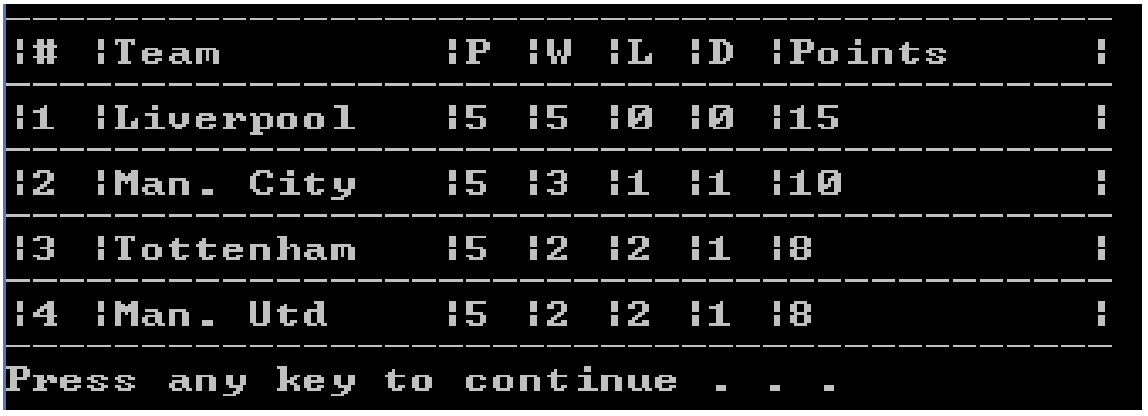
Exercise 11

Write a program called PetrolReceipt.java, which prints out a receipt for petrol to the screen, similar to the following:



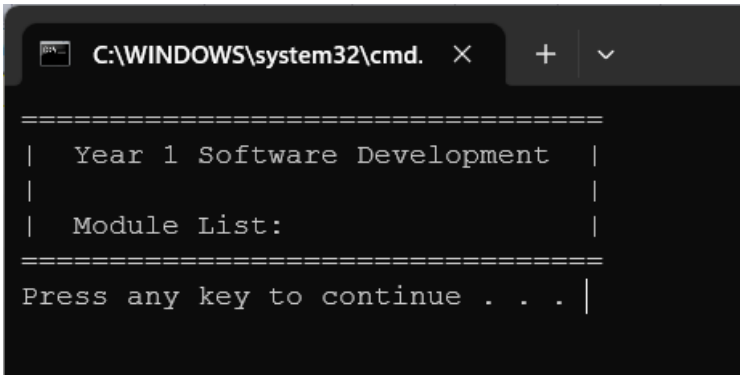
Exercise 12

Write a program called "EPL.java", which prints out the following league table to the screen:

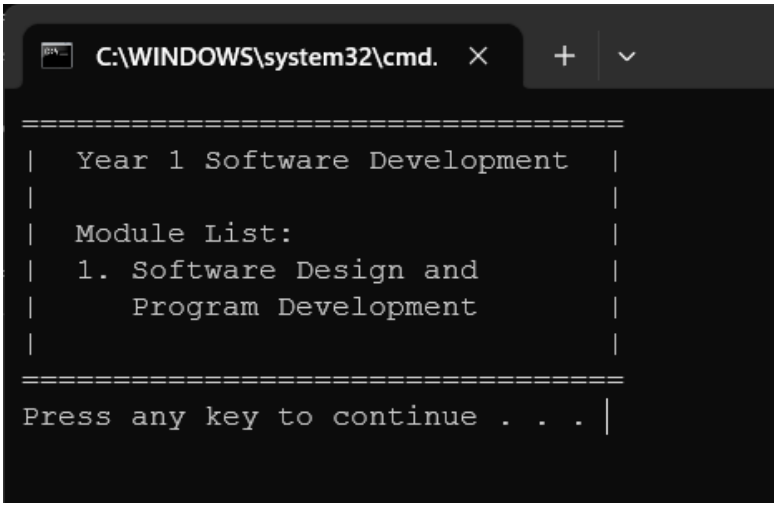


Exercise 13

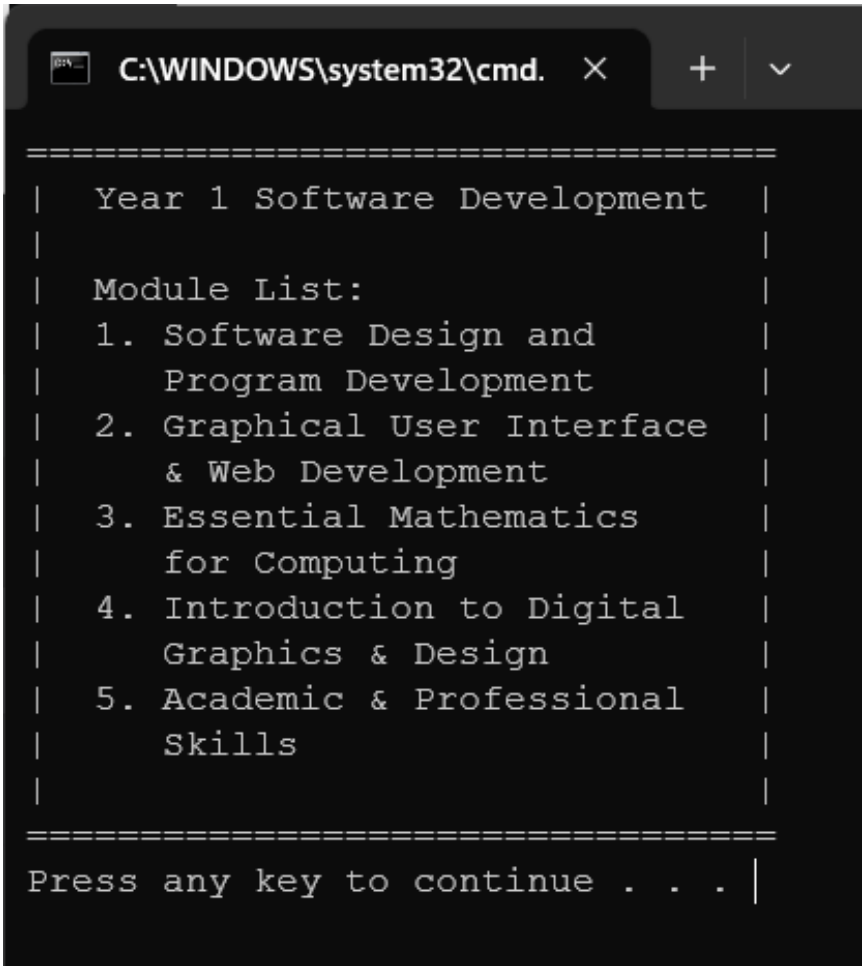
Create a java program in a file called “JavaProgram13.java” that outputs the following to the console screen.



Add additional code so that the output contains the list of modules, starting with SDPD:

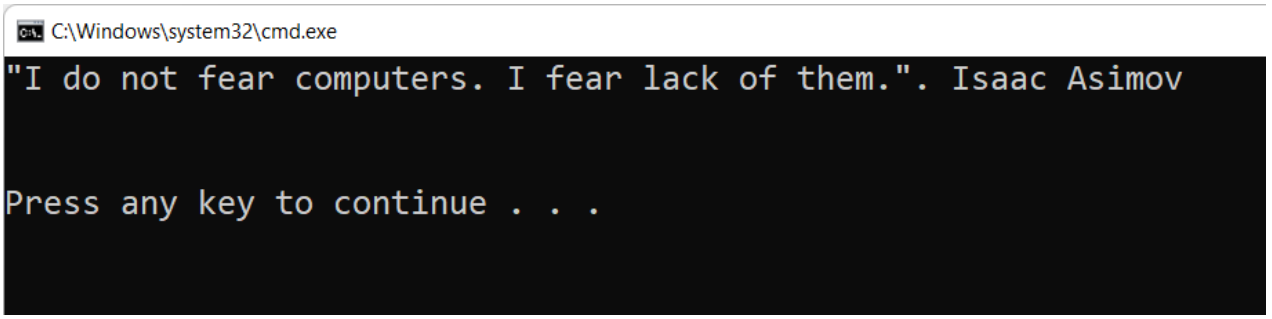


Amend your code so that the output shows all modules, similar to as shown here:



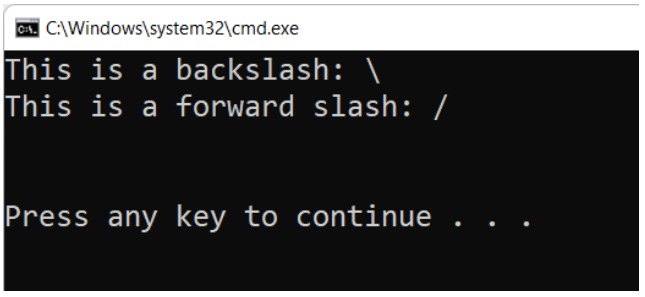
Exercise 14

Write a program called Exercise14.java, which prints the following line of text, (including the quotation marks), similar to as shown below. You can use an escape character to produce this output.



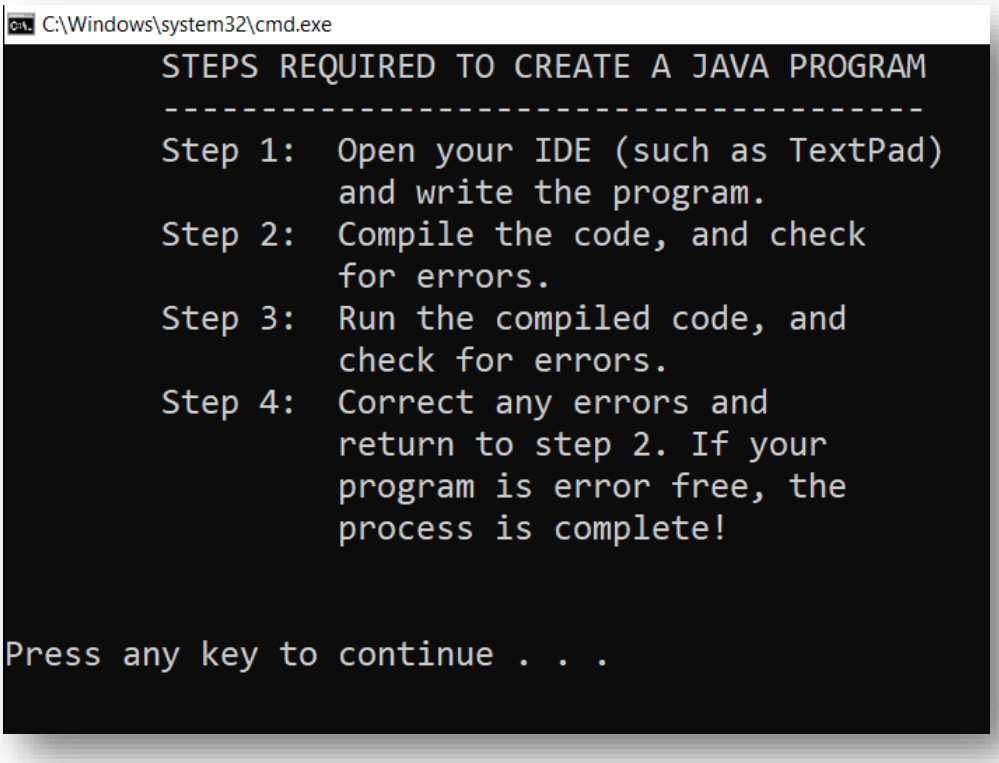
Exercise 15

Write a program called Exercise15.java, which prints the following lines of text, as shown below. You can use an escape character to produce this output.



Exercise 16

Write a program called Exercise16.java, which prints the following lines of text, as shown below. You should only use the *print* method (**not** *println*) to produce this output. Consider using tabs to indent the output (\t).



Exercise 17

Comments can be used to explain Java code, and to make it more readable. It can also be used to prevent execution when testing alternative code. A comment can provide information to a person reviewing code, but it has no effect on the program.

Single-line comments start with two forward slashes (//). Any text between // and the end of the line is ignored by Java (will not be executed). This example uses a single-line comment before a line of code:

```
1  // This is a comment - this will not appear anywhere when the program executes
2  public class Hello
3  {
4     public static void main(String[] args)
5     {
6         System.out.println("Hello World");
7     }
8  }
```

Add a comment that contains your name and student ID into <u>all your exercises</u> from today. <u>Recompile each program to ensure that they still run after adding the comment.</u>

Example:

```
1  // Your name and student ID here
2  public class Hello
3  {
4     public static void main(String[] args)
5     {
6         System.out.println("Hello World");
7     }
8  }
```