



Java Lab

Week 10 – Write and Append to a file

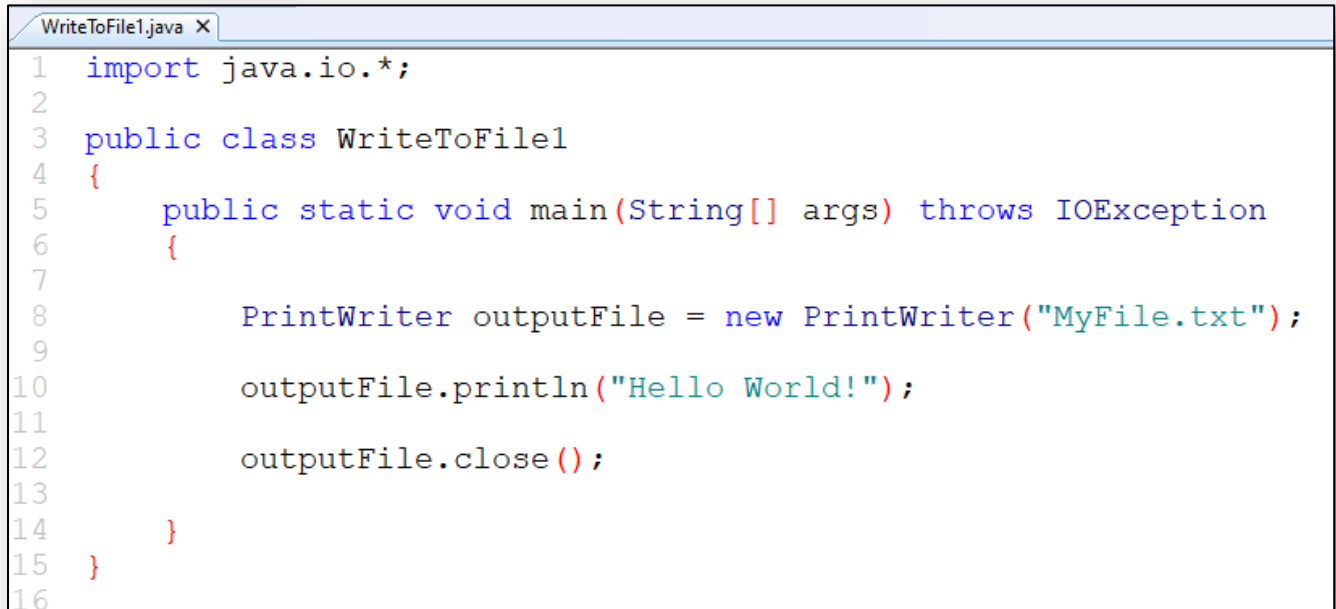
IMPORTANT! Save all your work to a safe location such as oneDrive.

Create a folder for SDPD into which you will save all your work for this module, arranged how you wish. Ideally you should create a folder each week for your lab exercises. Note that you should create a separate file for each exercise.

Exercise 1

Goal: Create a program in Java that outputs information to a file.

Create a new Java program called WriteToFile1. Follow the code shown below to write a single line of text to a file called MyFile.txt:

A screenshot of a Java IDE window titled 'WriteToFile1.java'. The code is as follows:

```
1  import java.io.*;
2
3  public class WriteToFile1
4  {
5      public static void main(String[] args) throws IOException
6      {
7
8          PrintWriter outputFile = new PrintWriter("MyFile.txt");
9
10         outputFile.println("Hello World!");
11
12         outputFile.close();
13
14     }
15 }
16
```

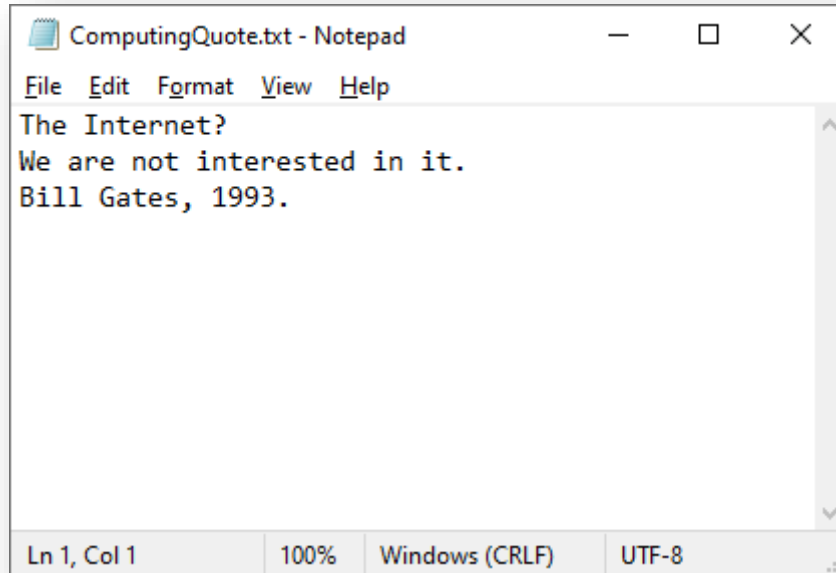
Note that this file will be created in the same folder as your java file.

Don't forget to include the `throws IOException` otherwise your program won't run.

Exercise 2

Goal: Create a program in Java that writes to a file

Create a new Java program called WriteToFile2. The program should create file called ComputingQuote.txt that contains the output shown below. This should use 3 *println* statements.

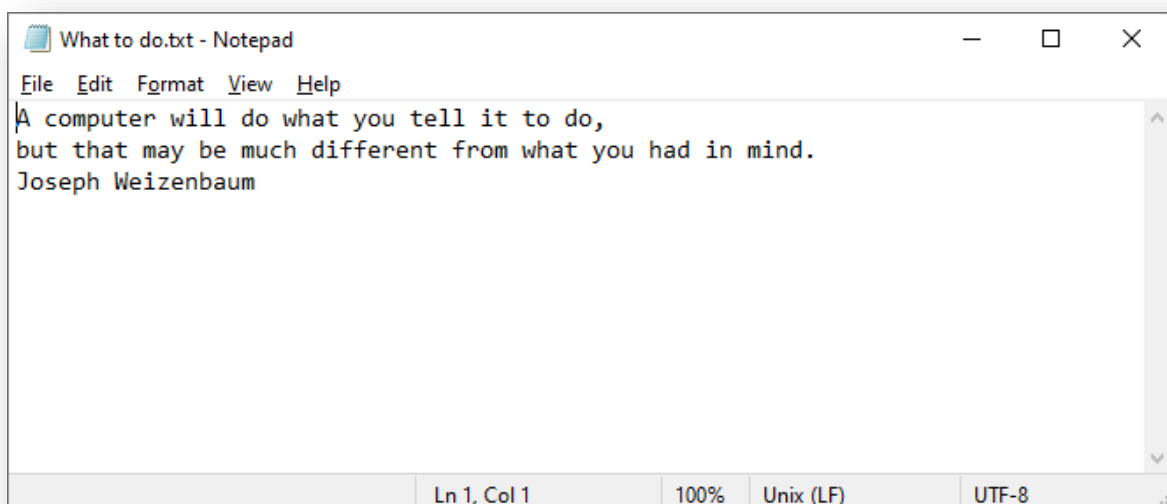


```
ComputingQuote.txt - Notepad
File Edit Format View Help
The Internet?
We are not interested in it.
Bill Gates, 1993.
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Exercise 3

Goal: Create a program in Java that writes to a file

Create a new Java program called WriteToFile3. The program should create file called *What to do.txt* that contains the output shown below. This should use a single *print* statement.

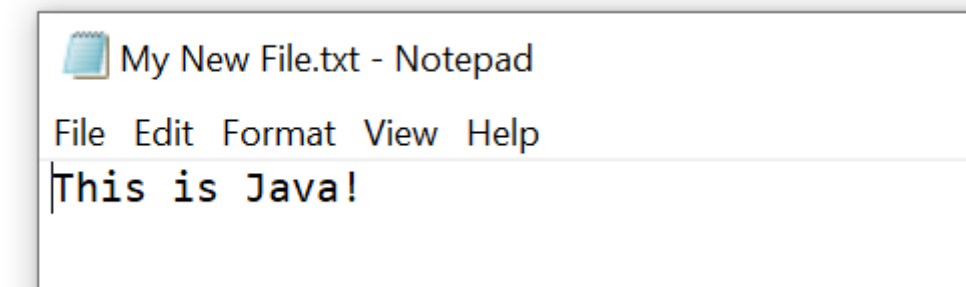
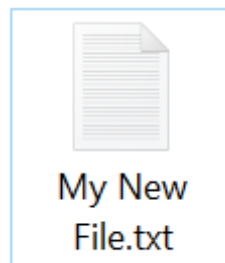
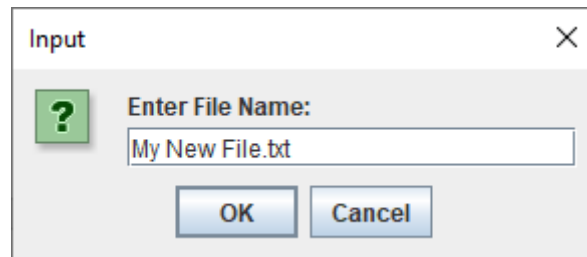


```
What to do.txt - Notepad
File Edit Format View Help
A computer will do what you tell it to do,
but that may be much different from what you had in mind.
Joseph Weizenbaum
Ln 1, Col 1 100% Unix (LF) UTF-8
```

Exercise 4

Goal: Create a program in Java that writes to a file

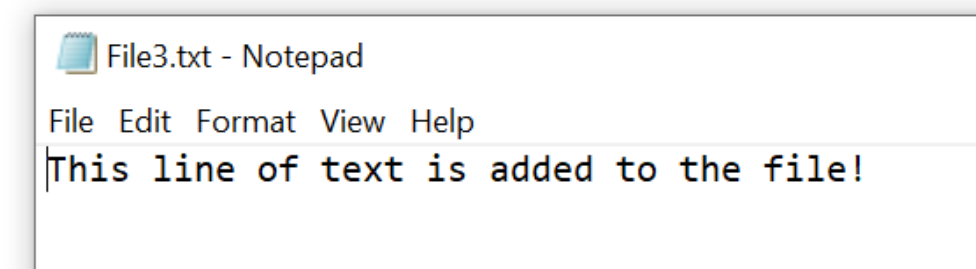
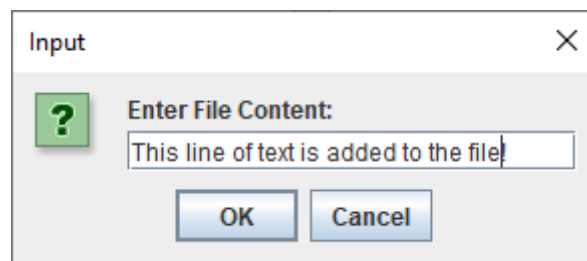
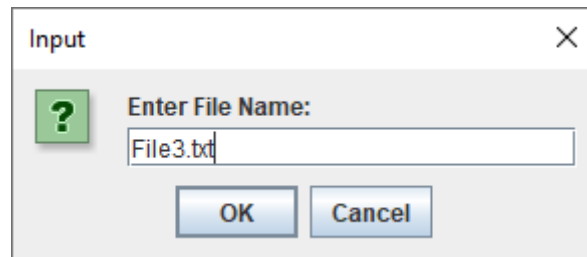
Create a new Java program called WriteToFile4. The program should prompt the user for the *file name* using JOptionPane and then output the sentence “This is Java!” to the file:



Exercise 5

Goal: Create a program in Java that writes to a file

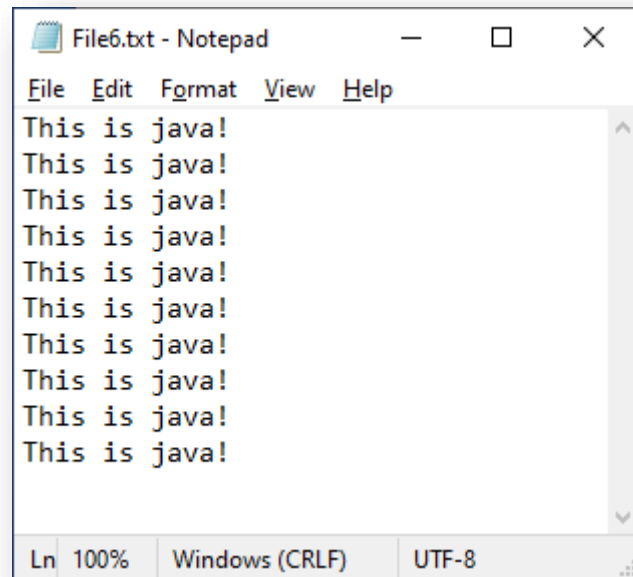
Create a new Java program called WriteToFile5. The program should prompt the user for the file name , and then prompt the user for the content of the file, as shown below:



Exercise 6

Goal: Create a program in Java that writes to a file

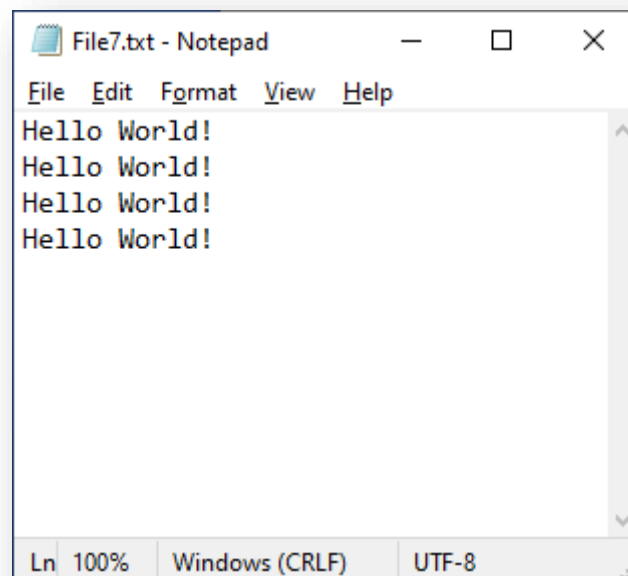
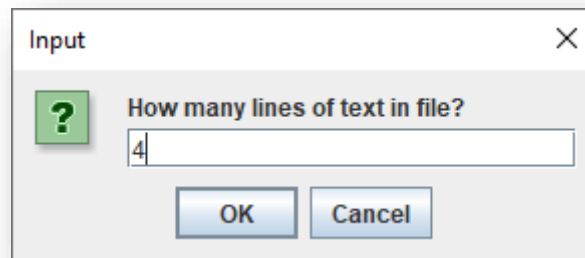
Create a new Java program called WriteToFile6. Using a for loop, the program should output 10 lines similar to as shown below:



Exercise 7

Goal: Create a program in Java that writes to a file

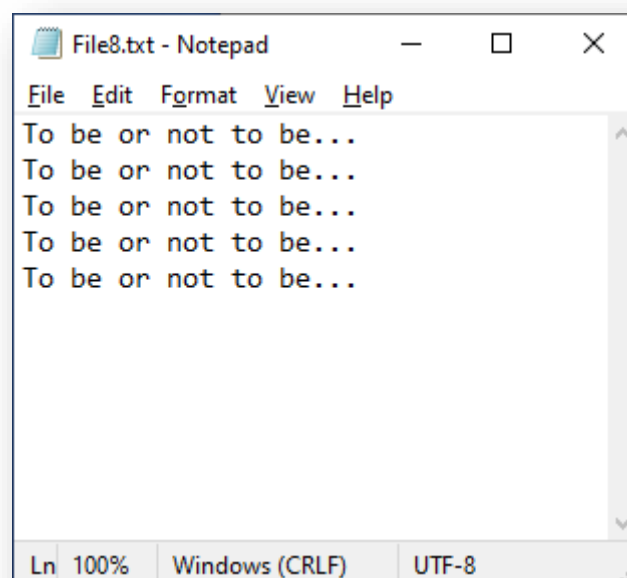
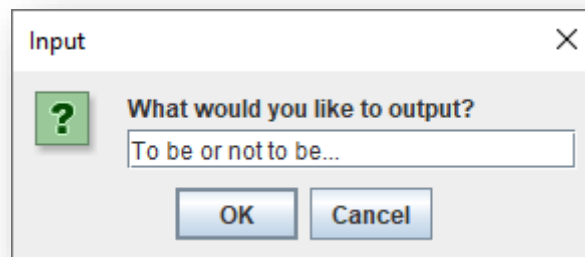
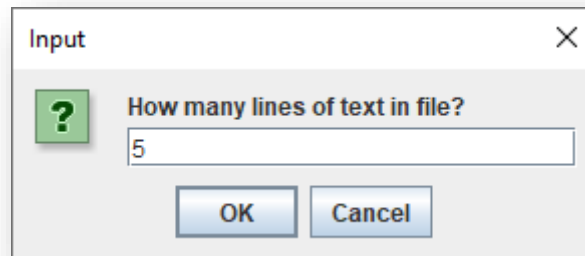
Create a new Java program called WriteToFile7. The program should prompt the user for how many times the line “Hello World” will be printed into a file called *file7.txt*. See below for output:



Exercise 8

Goal: Create a program in Java that writes to a file

Create a new Java program called WriteToFile8. The program should prompt the user for how many lines will be printed into the file, and what text will be output. See below for output:



Exercise 9

Goal: Create a program in Java that appends to an existing file

Create a new program in Java called WriteToFile9. This file should append (add) information to an existing file, rather than overwrite the existing file content – you will need to use *FileWriter* for this. You can use the file created in exercise 1 (*MyFile.txt*) to test this.

```
WriteToFile9.java X
1 // Import IO for writing to file
2 import java.io.*;
3
4 public class WriteToFile9
5 {
6     public static void main(String[] args) throws IOException
7     {
8         // Create instance of filewriter - this will use the file "MyFile.txt"
9         // This will allow to append to existing file
10        FileWriter fwriter = new FileWriter("MyFile.txt", true);
11
12        // Create an instance of PrintWriter
13        // referencing the name given for filewriter above (fwriter)
14        PrintWriter outputFile = new PrintWriter(fwriter);
15
16        // Line for appending (joining) to existing file
17        outputFile.println("This is using filewriter!");
18
19        // Close the file
20        outputFile.close();
21
22    }
23 }
```

MyFile.txt - Notepad

File Edit Format View Help

Hello World!This is using filewriter!

Ln 2, Col 1 100% Windows (CRLF) UTF-8

MyFile.txt - Notepad

File Edit Format View Help

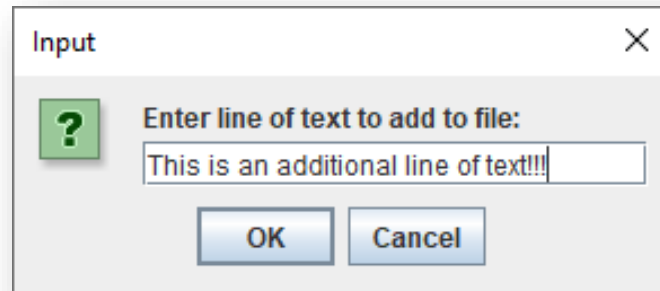
Hello World!This is using filewriter!
This is using filewriter!
This is using filewriter!
This is using filewriter!

Ln 1, Col 1 100% Windows (CRLF) UTF-8

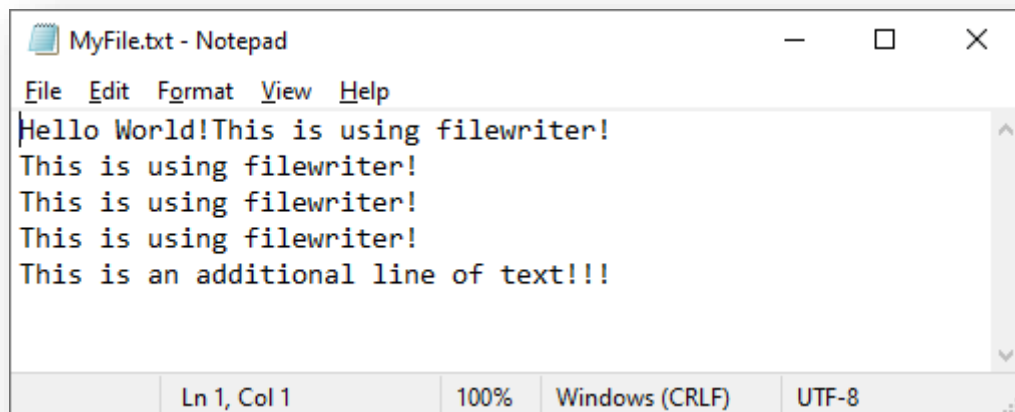
Exercise 10

Goal: Create a program in Java that appends to an existing file

Create a new program in Java called WriteToFile10. This file should append (add) information to an existing file, rather than overwrite the existing file content. Use the file from the previous exercise called "MyFile.txt" to test this.



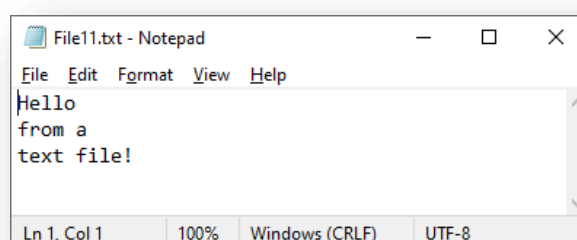
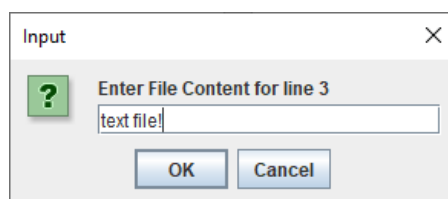
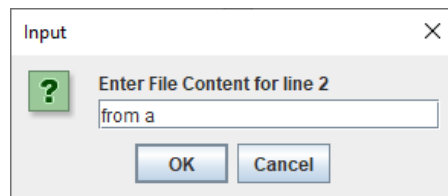
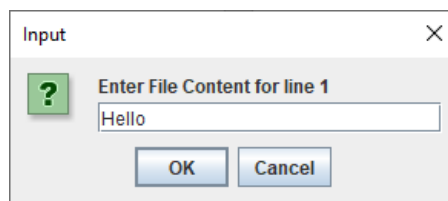
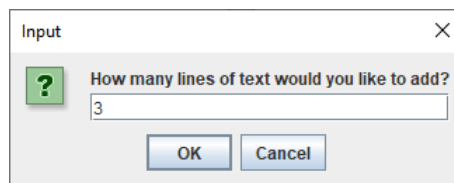
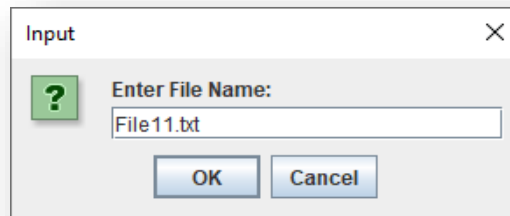
This line of text should now be present at the end of the file *MyFile.txt*



Exercise 11

Goal: Create a program in Java that prompts the user to add multiple lines of text

Create a new program in Java called WriteToFile11. This file should append information to a file, rather than overwrite the existing content. The program should ask the user how many lines of text they want to add to the file, then they will be presented with a prompt for each line to be added:

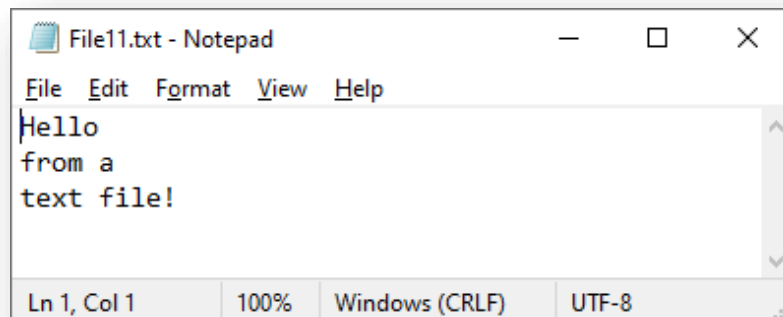


Exercise 12

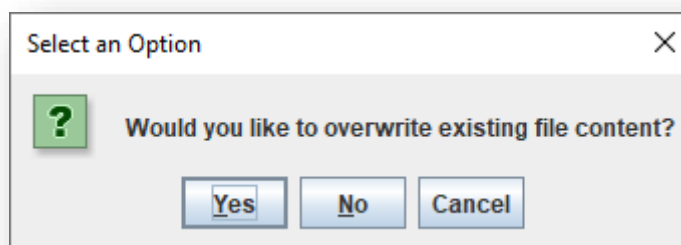
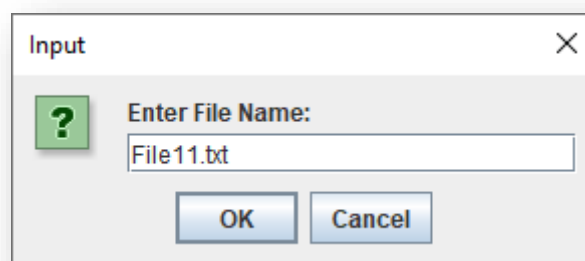
Goal: Create a program in Java that prompts the user to add multiple lines of text

Create a new program in Java called WriteToFile12. Using the file written to in the previous exercise for testing, write a program that prompts the user to specify a filename (file11.txt), and then prompts them with the question whether they would like to overwrite the existing content (yes) or append to the file (no), as shown in the example below:

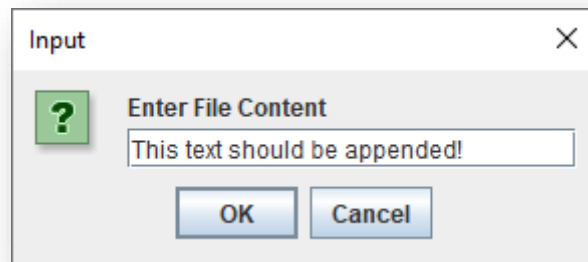
File11.txt prior to program running:



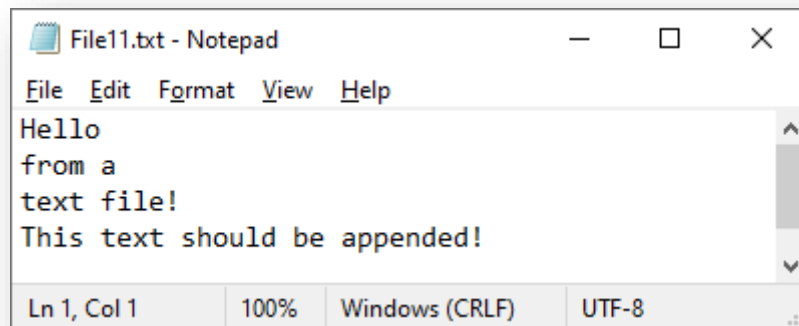
Program runs:



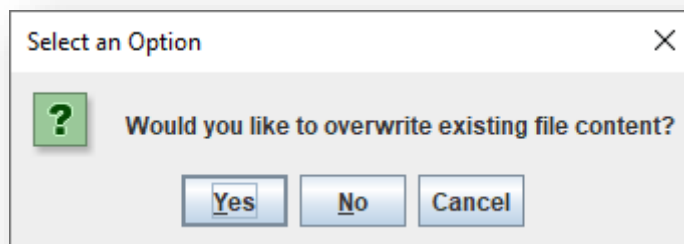
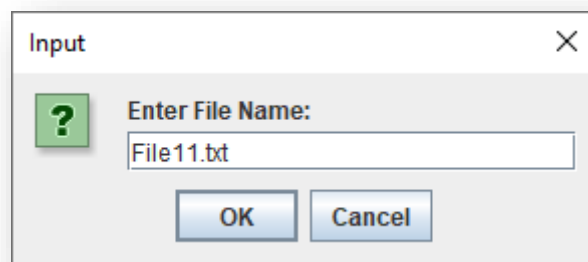
Choose no



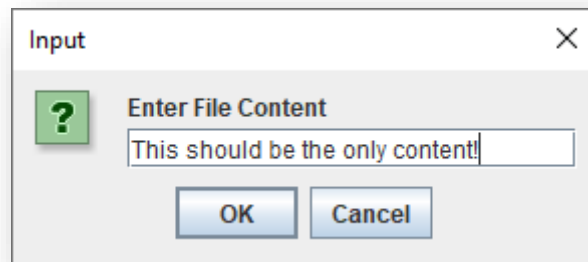
File after program runs:



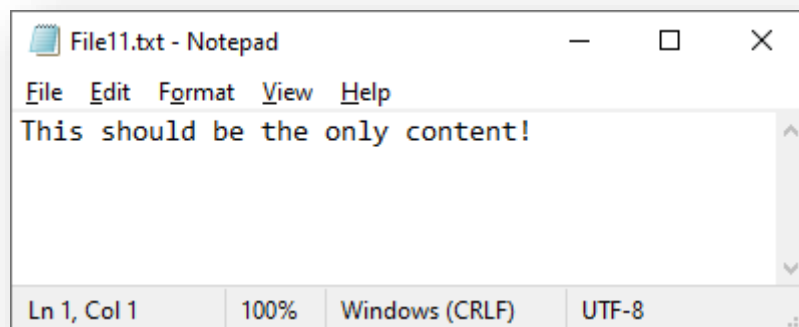
Re run the program, this time choosing yes when prompted to overwrite existing file content :



Choose Yes



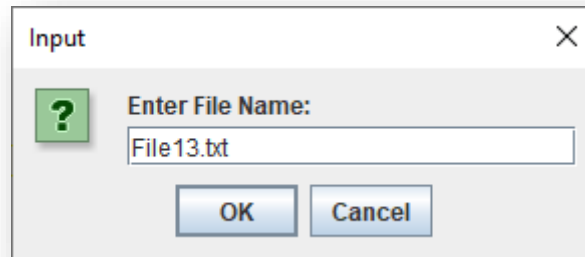
File should now be similar to as shown:



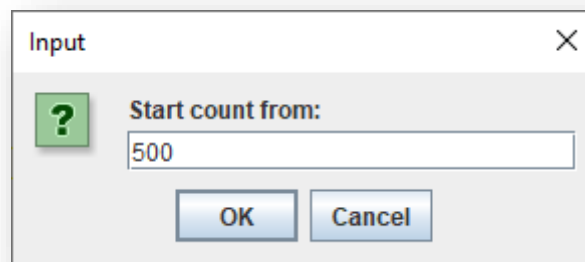
Exercise 13

Goal: Create a program in Java that prompts the user to add multiple lines of text

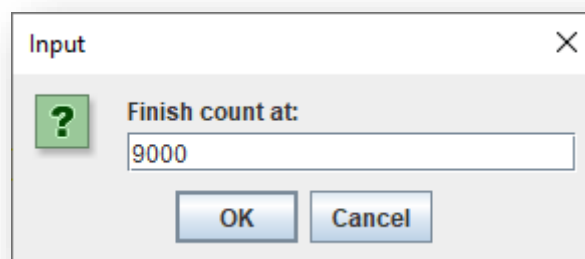
Create a new program in Java called WriteToFile7. The program should prompt the user to specify a starting number and finishing number. The program will count to screen and output the information to a file. the file name should also be provided by the user.



An input dialog box titled "Input" with a close button (X) in the top right corner. It contains a green square icon with a white question mark on the left. To the right of the icon is the label "Enter File Name:". Below the label is a text input field containing the text "File13.txt". At the bottom of the dialog are two buttons: "OK" and "Cancel".



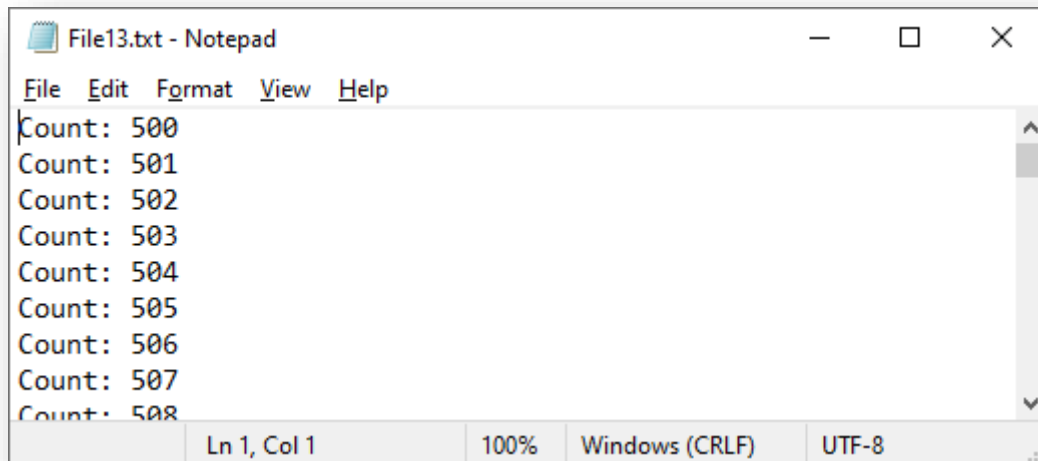
An input dialog box titled "Input" with a close button (X) in the top right corner. It contains a green square icon with a white question mark on the left. To the right of the icon is the label "Start count from:". Below the label is a text input field containing the text "500". At the bottom of the dialog are two buttons: "OK" and "Cancel".



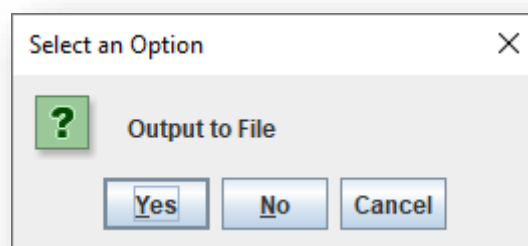
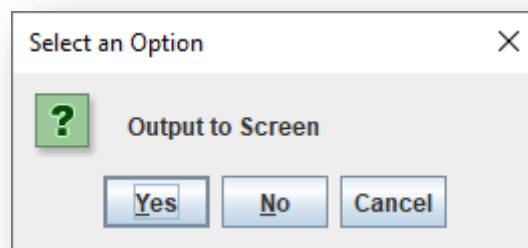
An input dialog box titled "Input" with a close button (X) in the top right corner. It contains a green square icon with a white question mark on the left. To the right of the icon is the label "Finish count at:". Below the label is a text input field containing the text "9000". At the bottom of the dialog are two buttons: "OK" and "Cancel".

Program outputs to **both** console and file:

```
C:\WINDOWS\system32\cmd.exe
Count: 8987
Count: 8988
Count: 8989
Count: 8990
Count: 8991
Count: 8992
Count: 8993
Count: 8994
Count: 8995
Count: 8996
Count: 8997
Count: 8998
Count: 8999
Count: 9000
Press any key to continue . . .
```



Amend your program so that the user is prompted on whether they want to output to the screen and the file.

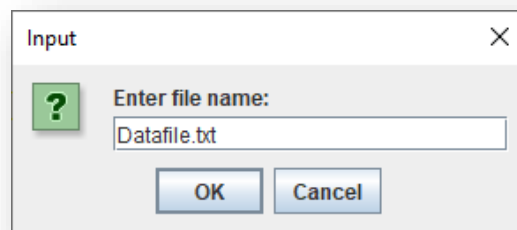


Exercise 14

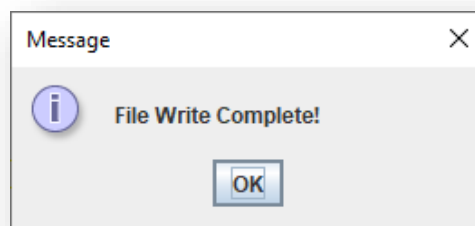
Goal: Create a program in Java that prompts the user to enter a filename, and the file is saved into a folder.

Create a new program in Java called WriteToFile14. The program should save a file using a file name based on information provided by the user. Before beginning, create a folder where you are creating your programs called "Data". The files created in this exercise will be saved into this folder.

Your program should prompt the user to enter a file name, as shown:

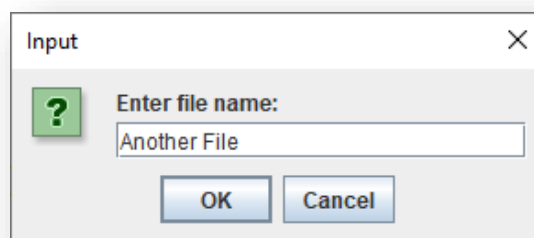


After writing the file, the program should produce a popup to show that the file has been created:

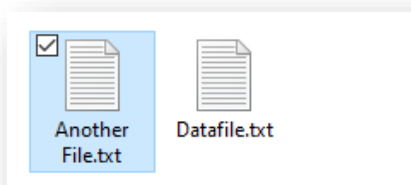


Check the "Data" folder you created to ensure that this is where the file was created.

Amend your program so that the user can enter a file name without a file extension, for example:



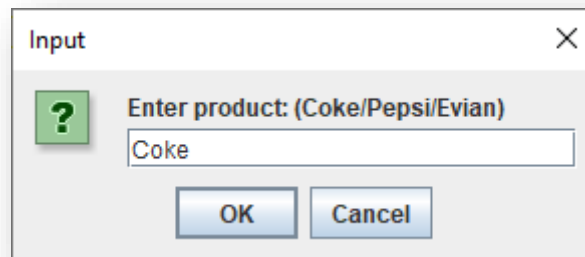
Entering "Another File" as the file name will produce a file called "Another File.txt":



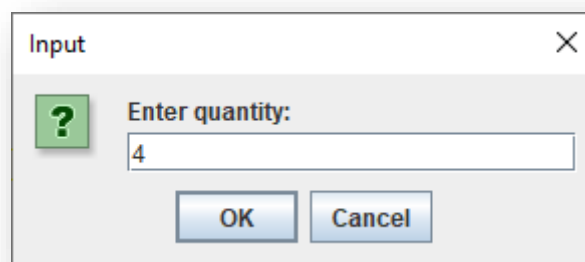
Exercise 15

Goal: Create a program in Java a shop selling 3 products, and produce a receipt based on data entered.

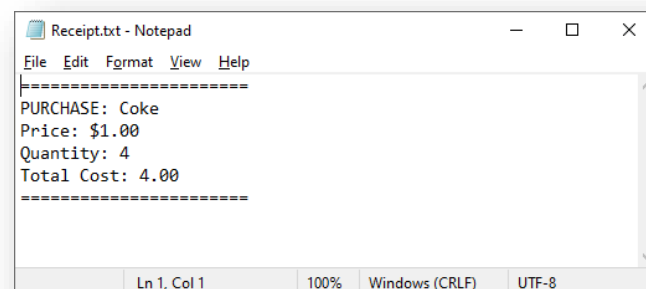
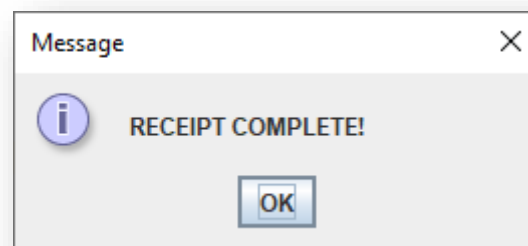
Create a new program in Java called WriteToFile15. The program should prompt the user to enter a product – Coke, Pepsi, or Evian.



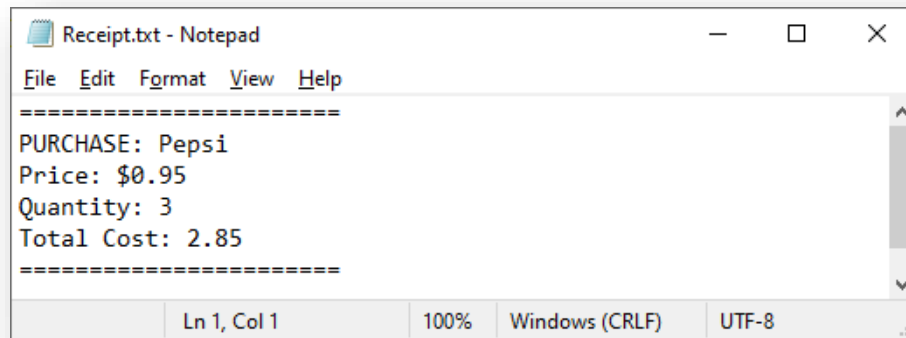
The program should then prompt the user to enter a quantity:



Finally, the program displays a popup as shown:



The prices for the products are as follows - Coke: \$1.00 – Pepsi: \$0.95 – Evian: \$0.75

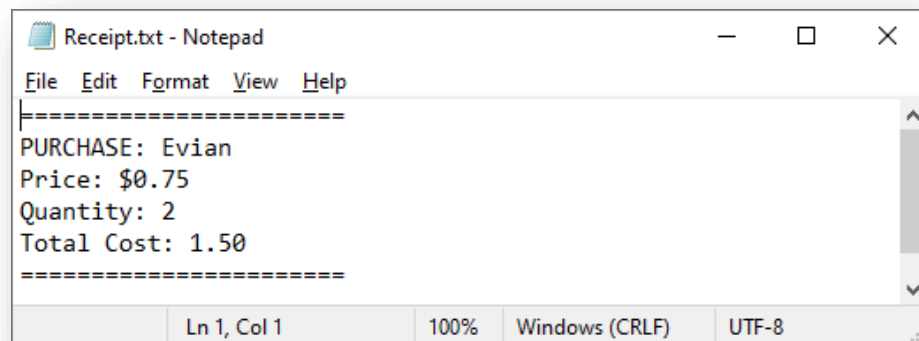


Receipt.txt - Notepad

File Edit Format View Help

```
=====
PURCHASE: Pepsi
Price: $0.95
Quantity: 3
Total Cost: 2.85
=====
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8



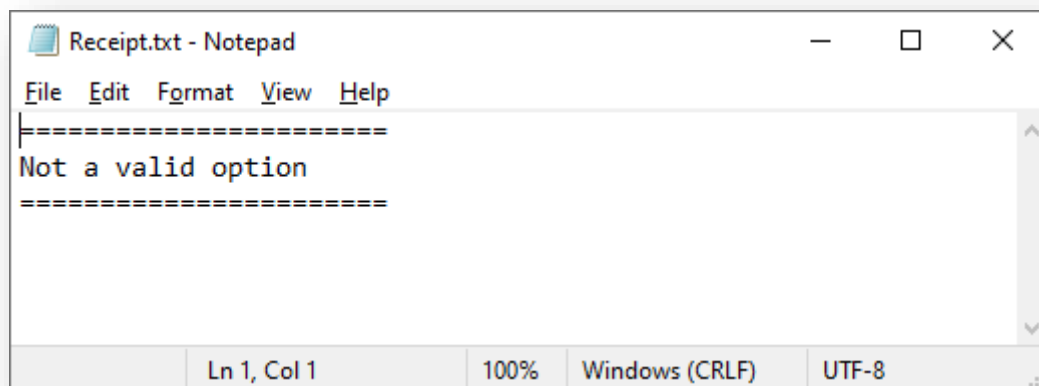
Receipt.txt - Notepad

File Edit Format View Help

```
=====
PURCHASE: Evian
Price: $0.75
Quantity: 2
Total Cost: 1.50
=====
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8

If no valid product is entered, the receipt has the following output:



Receipt.txt - Notepad

File Edit Format View Help

```
=====
Not a valid option
=====
```

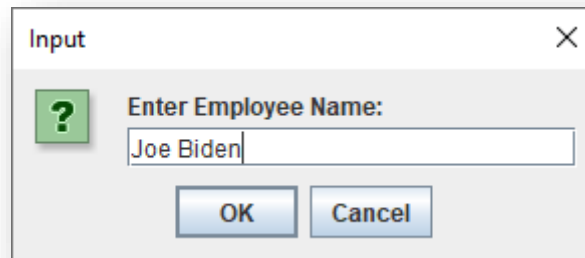
Ln 1, Col 1 100% Windows (CRLF) UTF-8

Exercise 16

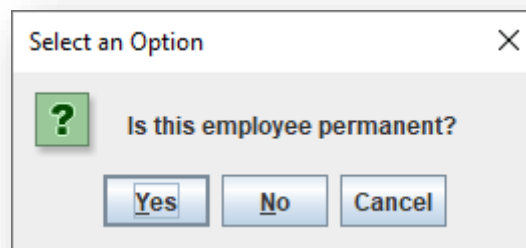
Goal: Create a payroll program in Java and produce a payslip based on data entered.

Create a new program in Java called WriteToFile16. The program should prompt the user to enter payroll information as shown below.

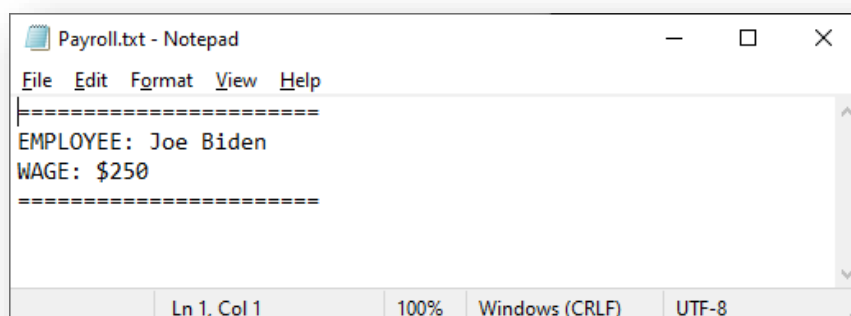
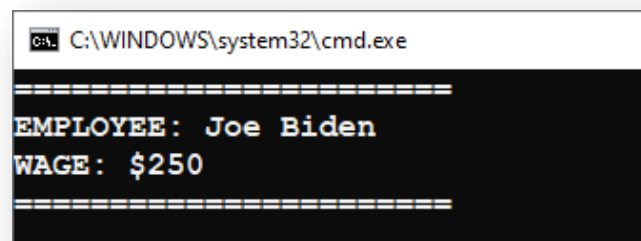
The program should first prompt the user to enter an employee name:



Then prompt the user on whether the user is permanent:

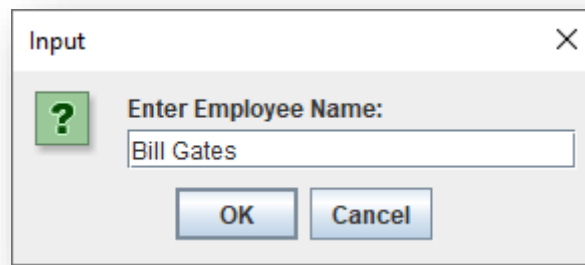


If the user click “Yes”, then the following is output to the console **and** output to a file called *payroll.txt*:

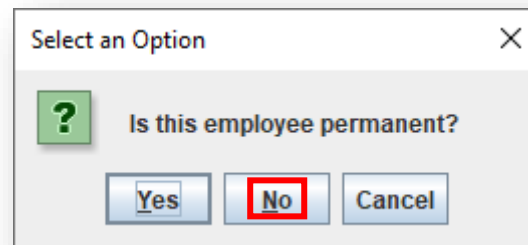


If the user clicks no, then the wage is \$125 dollars

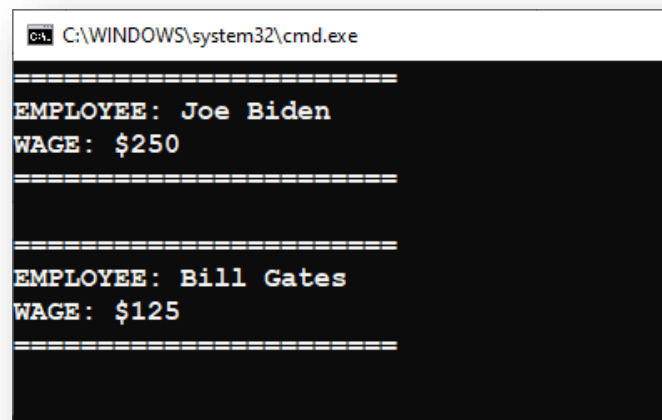
Amend your program so that when entering employee data, it prompts the user if they would like to add additional user data after each entry, eg:



A Windows-style dialog box titled "Input" with a close button (X) in the top right corner. It features a green square icon with a white question mark on the left. The text "Enter Employee Name:" is followed by a text input field containing "Bill Gates". At the bottom are "OK" and "Cancel" buttons.



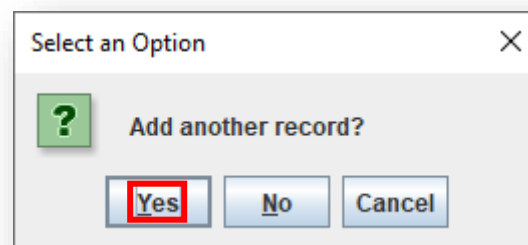
A Windows-style dialog box titled "Select an Option" with a close button (X) in the top right corner. It features a green square icon with a white question mark on the left. The text "Is this employee permanent?" is followed by three buttons: "Yes", "No", and "Cancel". The "No" button is highlighted with a red rectangular border.



```
C:\WINDOWS\system32\cmd.exe

=====
EMPLOYEE: Joe Biden
WAGE: $250
=====

=====
EMPLOYEE: Bill Gates
WAGE: $125
=====
```



A Windows-style dialog box titled "Select an Option" with a close button (X) in the top right corner. It features a green square icon with a white question mark on the left. The text "Add another record?" is followed by three buttons: "Yes", "No", and "Cancel". The "Yes" button is highlighted with a red rectangular border.

Input

Enter Employee Name:

Grace Hopper

OK Cancel

Select an Option

Is this employee permanent?

Yes No Cancel

Select an Option

Add another record?

Yes No Cancel

Sample Console output:

```
C:\WINDOWS\system32\cmd.exe

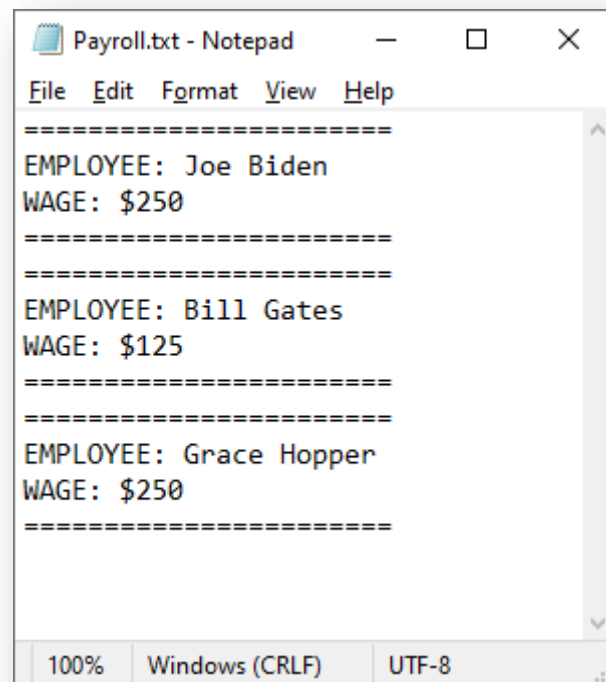
=====
EMPLOYEE: Joe Biden
WAGE: $250
=====

=====
EMPLOYEE: Bill Gates
WAGE: $125
=====

=====
EMPLOYEE: Grace Hopper
WAGE: $250
=====

Press any key to continue . . .
```

Sample payroll file output:



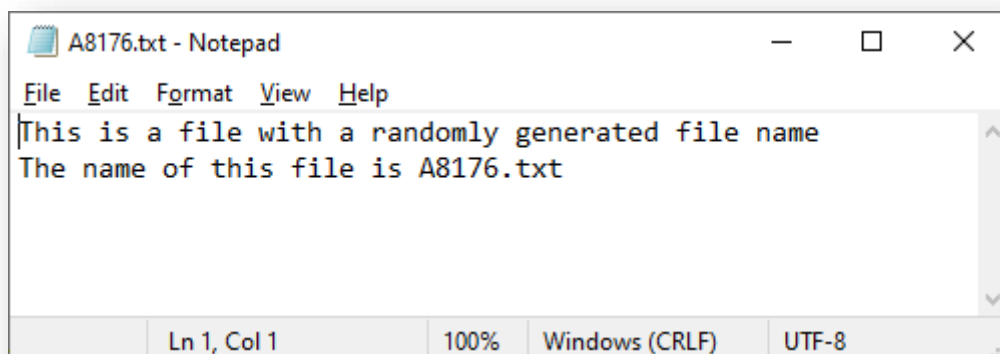
```
Payroll.txt - Notepad
File Edit Format View Help
=====
EMPLOYEE: Joe Biden
WAGE: $250
=====
EMPLOYEE: Bill Gates
WAGE: $125
=====
EMPLOYEE: Grace Hopper
WAGE: $250
=====
100% Windows (CRLF) UTF-8
```

Exercise 17

Goal: Create a program in Java that creates a file with a random file name.

Create a new program in Java called WriteToFile17. The program should generate a file with a random file name. The file name should have the format as following: AXXXX Where XXXX is a random number between 1000 and 9999, for example: A1234.txt

The file content should be as follows:



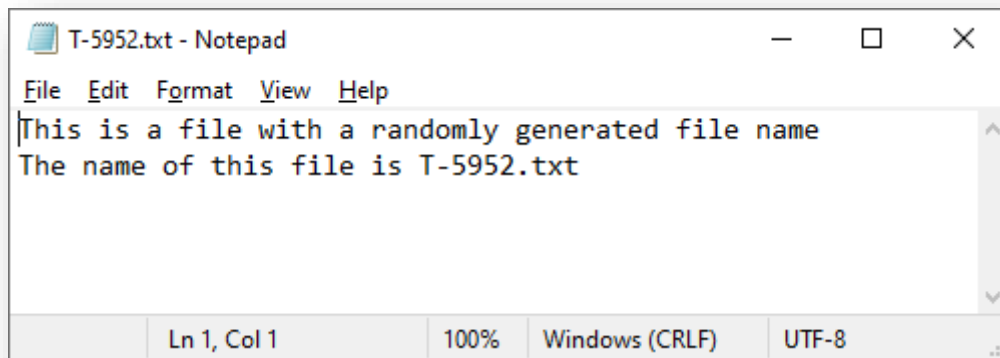
```
A8176.txt - Notepad
File Edit Format View Help
This is a file with a randomly generated file name
The name of this file is A8176.txt
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Exercise 18

Goal: Create a program in Java that creates a file with a random file name.

Create a new program in Java called WriteToFile18. The program should generate a file with a random file name. The file name should have the format as following: N-XXXX where N is either the letter R,S or T, and XXXX is a random number between 5500 and 6000, for example: T-5678.txt


The file content should be as follows:



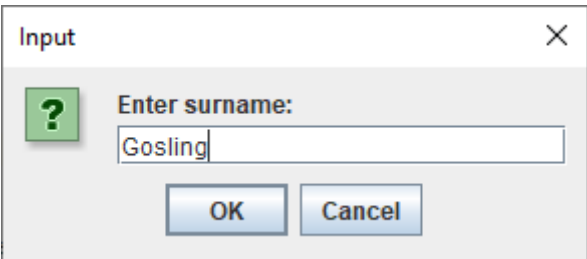
Exercise 19

Goal: Create a program in Java that prompts the user to data, and a report is generated based on this data.

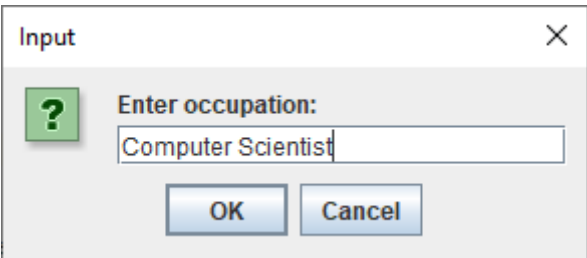
Create a new program in Java called WriteToFile14. The program should generate a report to file based on information provided by the user.



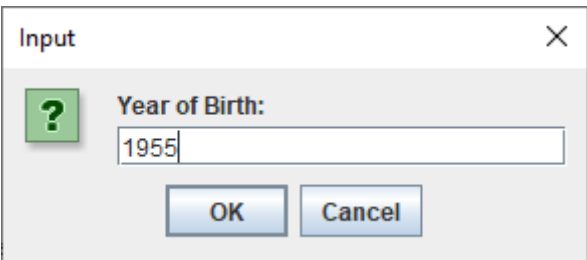
Input dialog box titled "Input" with a close button (X). It contains a green question mark icon, the label "Enter first name:", a text input field containing "James", and "OK" and "Cancel" buttons.



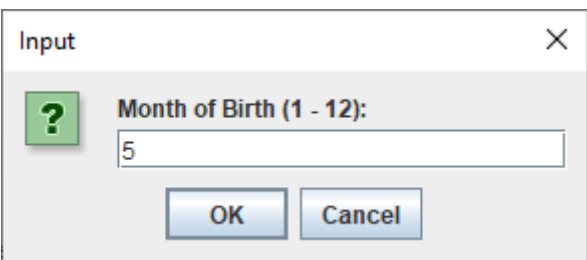
Input dialog box titled "Input" with a close button (X). It contains a green question mark icon, the label "Enter surname:", a text input field containing "Gosling", and "OK" and "Cancel" buttons.



Input dialog box titled "Input" with a close button (X). It contains a green question mark icon, the label "Enter occupation:", a text input field containing "Computer Scientist", and "OK" and "Cancel" buttons.




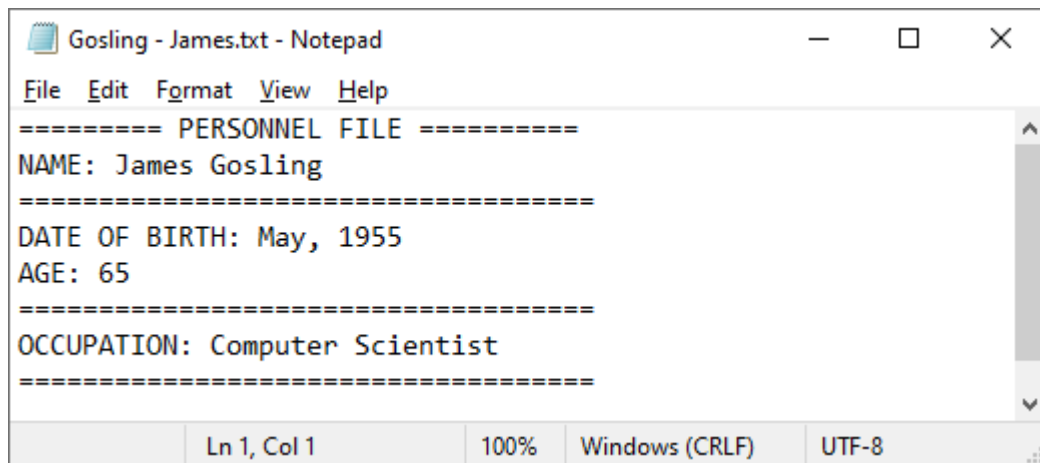
Input dialog box titled "Input" with a close button (X). It contains a green question mark icon, the label "Year of Birth:", a text input field containing "1955", and "OK" and "Cancel" buttons.



Input dialog box titled "Input" with a close button (X). It contains a green question mark icon, the label "Month of Birth (1 - 12):", a text input field containing "5", and "OK" and "Cancel" buttons.

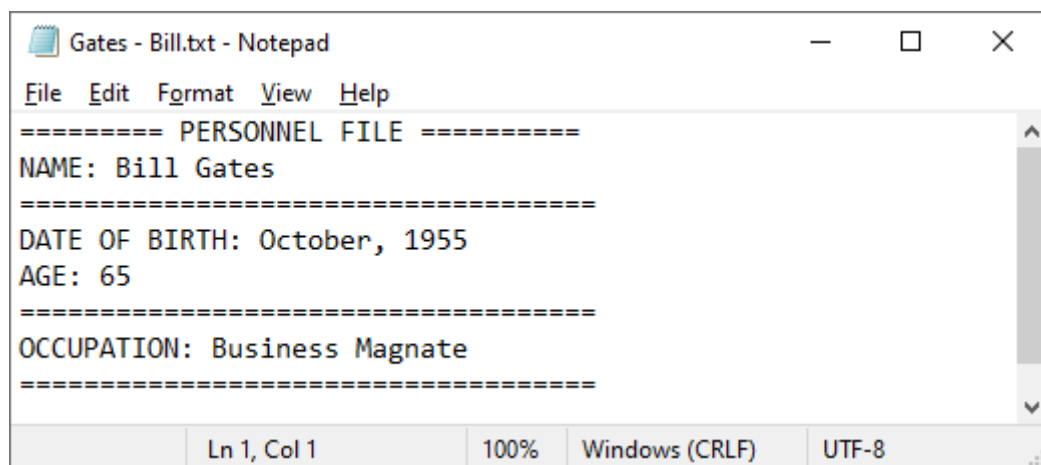


 Gosling - James.txt

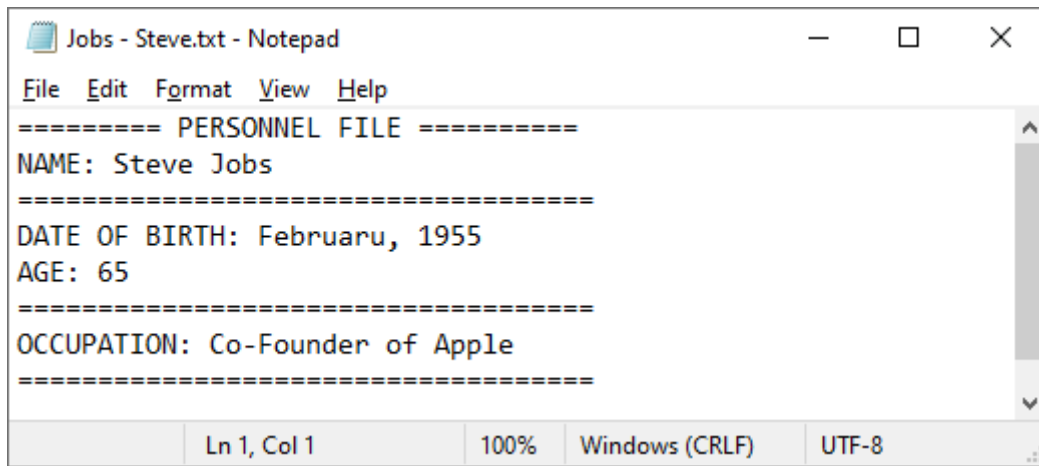


```
Gosling - James.txt - Notepad
File Edit Format View Help
===== PERSONNEL FILE =====
NAME: James Gosling
=====
DATE OF BIRTH: May, 1955
AGE: 65
=====
OCCUPATION: Computer Scientist
=====
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

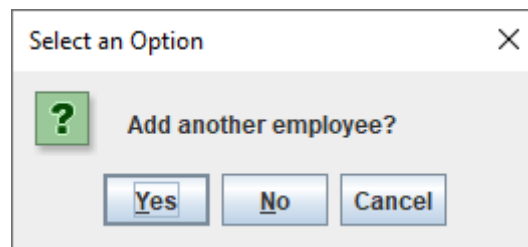
Test your program with the following data – each input should generate a different file:



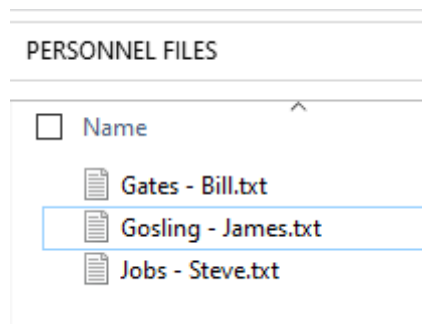
```
Gates - Bill.txt - Notepad
File Edit Format View Help
===== PERSONNEL FILE =====
NAME: Bill Gates
=====
DATE OF BIRTH: October, 1955
AGE: 65
=====
OCCUPATION: Business Magnate
=====
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```



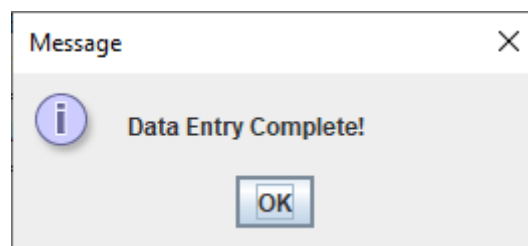
Amend your program so that at the end of each cycle, the user is prompted to enter another personnel entry by clicking yes to continue or no to end entry (see below). Change your code so that the files are automatically saved to a folder called “PERSONNEL FILES”.



The files should be saved into the “PERSONNEL FILES” folder:



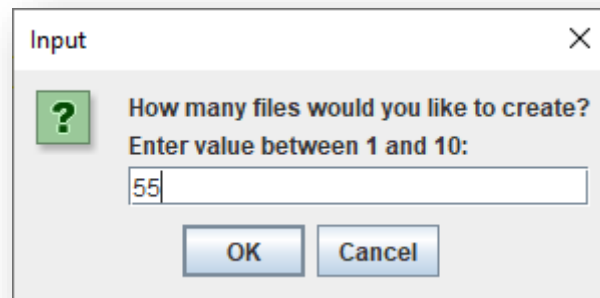
On completion of all entries, the following message is displayed:



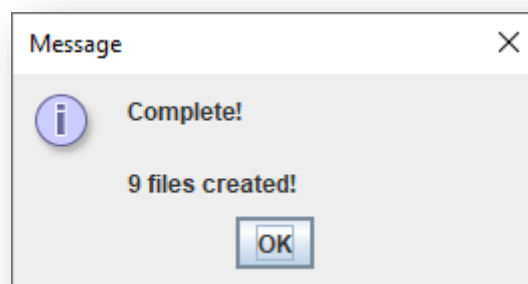
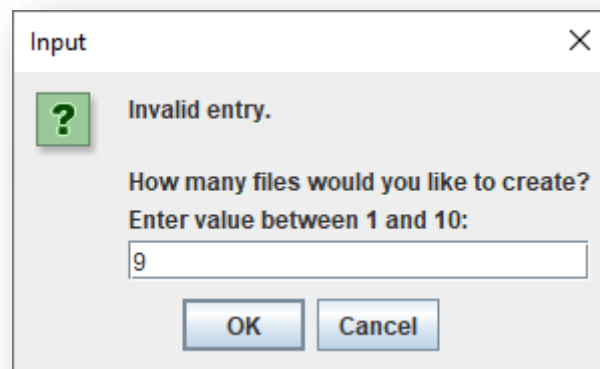
Exercise 20

Goal: Create a program in Java that prompts the user to enter number of files to be created.

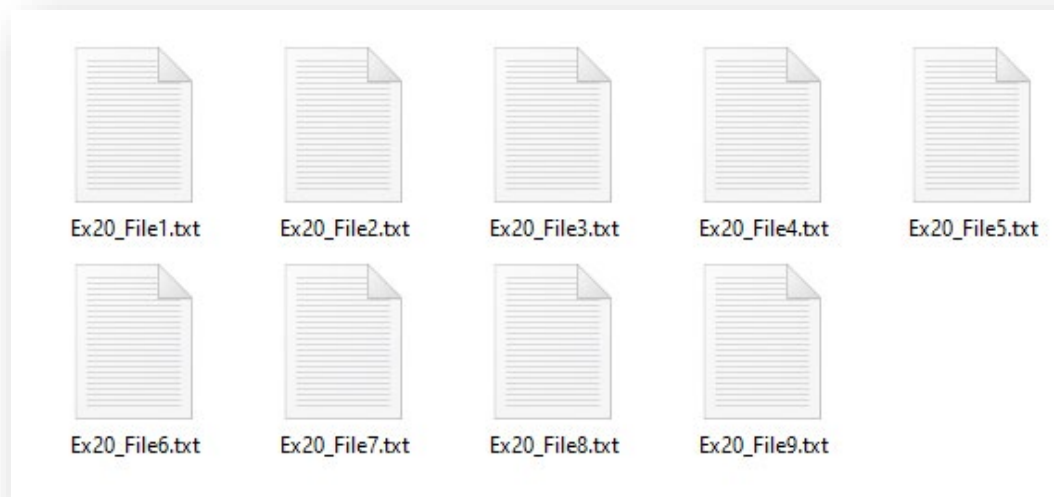
Create a new program in Java called WriteToFile20. The program should prompt the user to specify the number of files to be created.



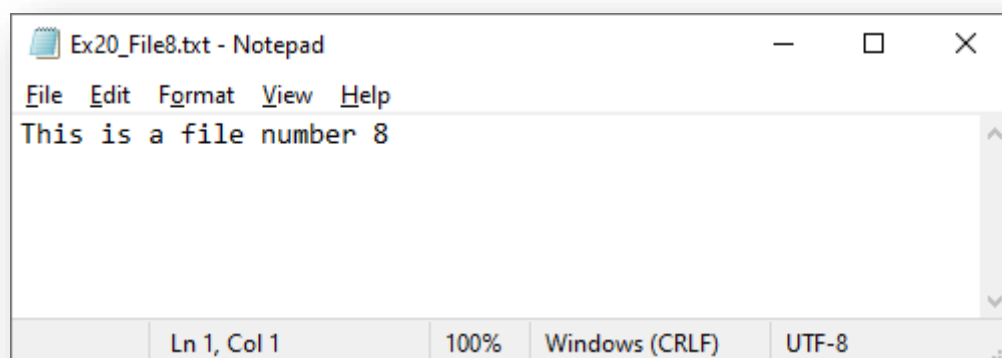
The only valid numbers accepted are between 1 and 10. While the user enters any value lower or higher, they are presented with the following prompt:



The program should create the correct number of files, with the output and filename similar to as shown below:



Filename should be in the format of Ex20_FileX, where X is from 1 to the number of files being created.
The file content should be similar to as shown here:



The program should then generate a report to file based on information provided by the user.