OCA 6: Flow Control and Exceptions

Exercises

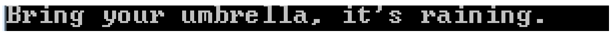
Q1 – JAT6Ex1

Create a class named **JAT6Ex1.java**.

In the main method, create a variable named, *isRaining* (a boolean) and assign it a value of true.

Evaluate the value stored in the variable, using an **if-else statement**. The program should display one of the following messages:

- “Bring your umbrella, it’s raining."
- “No need for an umbrella, it’s dry”.



```
Bring your umbrella, it's raining.
```

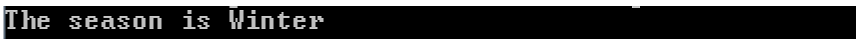
Q2 – JAT6Ex2

Create a class named **JAT6Ex2.java.**

Write a program, which displays the name of the season to match a given temperature. In the main method, create a variable named, *currentTemperature* (a float) and assign it a value of 6.00f.

Use an **if-else if-else statement** to evaluate the value stored in the variable. Display the name of the appropriate season.

Temperature	Season
>=16	Summer
11-15	Spring
7-10	Autumn
<7	Winter



```
The season is Winter
```

Please Turn Over

Q3 – JAT6Ex3

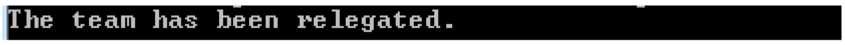
Create a class named **JAT6Ex3.java.**

In the main method, create a variable named, *pointsWon* (an integer). This variable is used to store the number of points won by your favourite football team in the Premier League. Assign a value of 14 to the variable.

Use an **if-else if-else statement** to evaluate the value stored in the variable.

Display an appropriate message.

pointsWon	Message
>=65	The team has qualified for the Champions League.
57-64	The team has qualified for the Europa League.
50-56	The team has finished in mid-table.
40-49	The team has finished in the lower-half of the table but has avoided relegation.
<40	The team has been relegated.



```
The team has been relegated.
```

Q4 – JAT6Ex4

Create a class named **JAT6Ex4.java.**

Each month, a store offers a loyalty gift to customers, depending on the number of loyalty points they have earned in the month.

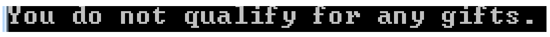
Create a program, which determines whether a customer has qualified for a loyalty gift and if so, which one.

In the main method, create a variable named *noLoyaltyPointsEarned* (an integer) and assign it a value of 149.

Use an **if-else if-else statement** to evaluate the value stored in the variable.

Display an appropriate message.

Points Target Reached	Loyalty Gift
<150	No gift.
151 - 250	35 Euro off your next shopping bill
251 - 350	50 Euro off your next shopping bill
>350	75 Euro off your next shopping bill



```
You do not qualify for any gifts.
```

Please Turn Over

Q5 – JAT6Ex5

Create a class named **JAT6Ex5.java.**

A motor tax office operates a queue system to allow members of the public to consult with officials.

A member of the public is given a ticket number and a position in the queue.

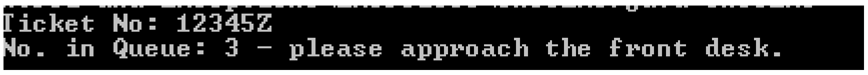
In the main method, create the following two variables.

byte positionInQueue = 3;

String ticketNo = "12345Z";

Test the value stored in the variable positionInQueue. If the number stored is from 1-3 (inclusive), a message should be displayed (as shown below), instructing the waiting person to proceed to the front desk. Otherwise, the person should be told to remain seated.

Solve the problem using a **switch statement**.



```
Ticket No: 12345Z
No. in Queue: 3 - please approach the front desk.
```

Q6 – JAT6Ex6

Create a class named **JAT6Ex6.java.**

A steak is a piece of meat which can be cooked in one of several different ways.

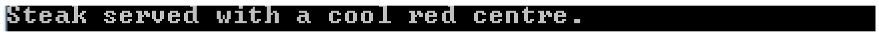
Style	**Description**
Rare	Cool RED Centre
Medium Rare	Hot RED Centre
Medium	Pink Throughout
Well Done	Brown and well-cooked throughout.

Create a Java program which uses an enum to store the following constants, RARE, MEDIUM_RARE, MEDIUM, WELL_DONE.

In the main method, create a variable of the specified enum type, and assign it a value of RARE.

Evaluate the value stored in the variable and display an appropriate description to the user.

Solve the problem using a **switch statement**.



```
Steak served with a cool red centre.
```

Please Turn Over

Q7 – JAT6Ex7

Create a class named **JAT6Ex7.java.**

Create a program, which displays a welcome message to a user to match their locale.

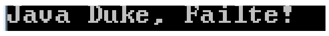
Locale	Welcome Message
Ireland	Failte!
France	Bienvenue!
Germany	Willkommen!
Spain	Bienvenida!

In the main method, create the following variables.

- locale (a string). Assign it the value, “Ireland”.
- user (a string). Assign it the value, “Java Duke”.

Use a **switch statement** to evaluate the value stored in the variable, locale.

Display an appropriate message to the user.



```
Java Duke, Failte!
```

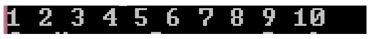
Q8 – JAT6Ex8

Create a class named **JAT6Ex8.java.**

In the main method, create an array named *myBytes*. The size of the array is 10 and it should store byte literal values.

Use a for loop to populate the array with the values, 1-10 (inclusive).

Use a separate for loop to display the contents of the array in the console.

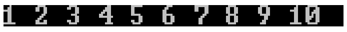


```
1 2 3 4 5 6 7 8 9 10
```

Q9 – JAT6Ex9

Create a class named **JAT6Ex9.java.**

Re-write Q8 using while loops.



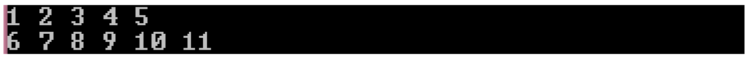
```
1 2 3 4 5 6 7 8 9 10
```

Please Turn Over

Q10 – JAT6Ex10

Create a class named **JAT6Ex10.java.**

The following numbers should be stored in a two-dimensional array of type int.



```
1 2 3 4 5
6 7 8 9 10 11
```

The two-dimensional array itself should store two one-dimensional arrays.

The two-dimensional array should be declared, constructed and initialized on one line.

The first one dimensional array should store the numbers 1 through 5.

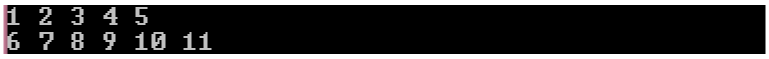
The second one dimensional array should store the numbers 6 through 11.

Display the contents of the two-dimensional array, using for loops.

Q11 – JAT6Ex11

Create a class named **JAT6Ex11.java.**

Re-write Q10 using while loops.



```
1 2 3 4 5
6 7 8 9 10 11
```

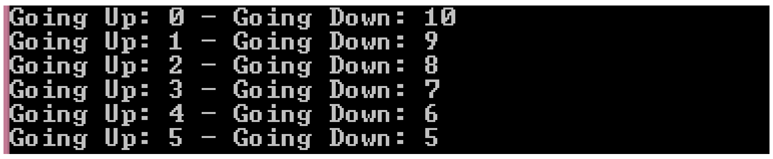
Q12 – JAT6Ex12

Create a class named **JAT6Ex12.java.**

The following output was produced using a **single for loop**. Two variables were initialised in the first part of the loop. (GoingUp and GoingDown).

One variable is increasing in value while the other is decreasing. When the two variables have the same value, the loop should exit.

Create the for loop to produce the specified output.

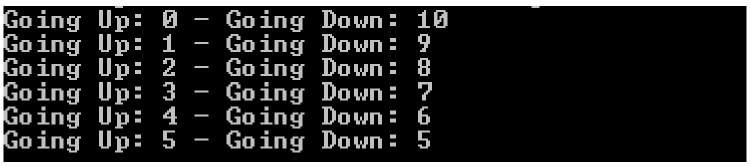


```
Going Up: 0 - Going Down: 10
Going Up: 1 - Going Down: 9
Going Up: 2 - Going Down: 8
Going Up: 3 - Going Down: 7
Going Up: 4 - Going Down: 6
Going Up: 5 - Going Down: 5
```

Q13 – JAT6Ex13

Create a class named **JAT6Ex13.java.**

Can you re-write Q12 using a **single while loop**.



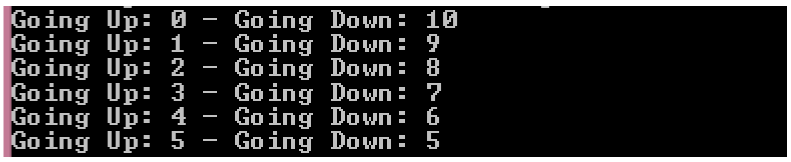
```
Going Up: 0 - Going Down: 10
Going Up: 1 - Going Down: 9
Going Up: 2 - Going Down: 8
Going Up: 3 - Going Down: 7
Going Up: 4 - Going Down: 6
Going Up: 5 - Going Down: 5
```

Please Turn Over

Q14 – JAT6Ex14

Create a class named **JAT6Ex14.java.**

Can you re-write Q12 using a **do while loop**.



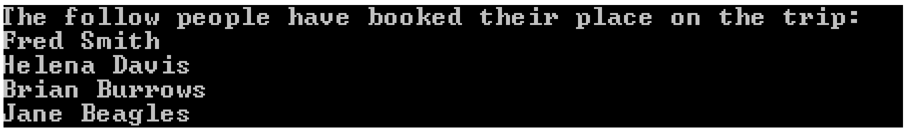
```
Going Up: 0 - Going Down: 10
Going Up: 1 - Going Down: 9
Going Up: 2 - Going Down: 8
Going Up: 3 - Going Down: 7
Going Up: 4 - Going Down: 6
Going Up: 5 - Going Down: 5
```

Q15 – JAT6Ex15

Create a class named **JAT6Ex15.java.**

In this program, you are asked to store the names of people going on a camping trip. An arraylist (of type String) should be used to store the names.

An **enhanced for loop** should be used to display the contents of the arraylist.



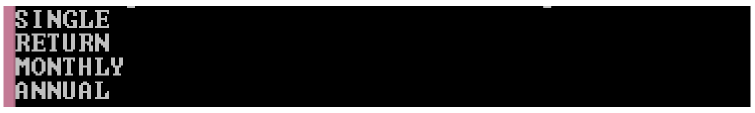
```
The follow people have booked their place on the trip:
Fred Smith
Helena Davis
Brian Burrows
Jane Beagles
```

Q16 – JAT6Ex16

Create a class named **JAT6Ex16.java**

Create an enum named *TicketType* to store the following constant values (SINGLE,RETURN,MONTHLY,ANNUAL).

Use an **enhanced for loop** to display the values stored in the enum.



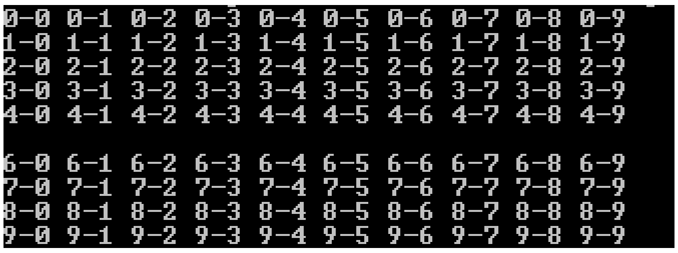
```
SINGLE
RETURN
MONTHLY
ANNUAL
```

Please Turn Over

Q17 – JAT6Ex17

Create a class named **JAT6Ex17.java**

Create the following output using **nested for loops** (an outer loop which contains an inner loop).

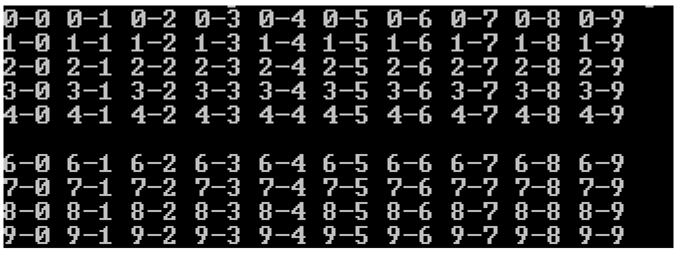


```
0-0 0-1 0-2 0-3 0-4 0-5 0-6 0-7 0-8 0-9
1-0 1-1 1-2 1-3 1-4 1-5 1-6 1-7 1-8 1-9
2-0 2-1 2-2 2-3 2-4 2-5 2-6 2-7 2-8 2-9
3-0 3-1 3-2 3-3 3-4 3-5 3-6 3-7 3-8 3-9
4-0 4-1 4-2 4-3 4-4 4-5 4-6 4-7 4-8 4-9

6-0 6-1 6-2 6-3 6-4 6-5 6-6 6-7 6-8 6-9
7-0 7-1 7-2 7-3 7-4 7-5 7-6 7-7 7-8 7-9
8-0 8-1 8-2 8-3 8-4 8-5 8-6 8-7 8-8 8-9
9-0 9-1 9-2 9-3 9-4 9-5 9-6 9-7 9-8 9-9
```

Q18 – JAT6Ex18

Create a class named **JAT6Ex18.java**

Re-write Q17 using **nested while loops** (an outer loop which contains an inner loop).

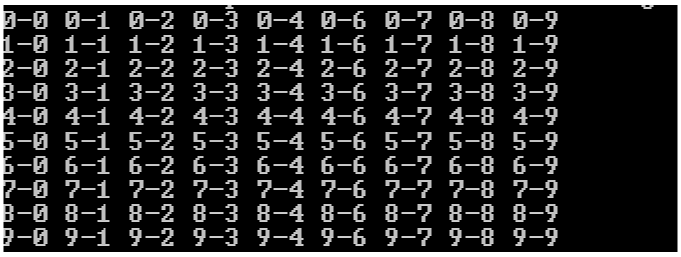


```
0-0 0-1 0-2 0-3 0-4 0-5 0-6 0-7 0-8 0-9
1-0 1-1 1-2 1-3 1-4 1-5 1-6 1-7 1-8 1-9
2-0 2-1 2-2 2-3 2-4 2-5 2-6 2-7 2-8 2-9
3-0 3-1 3-2 3-3 3-4 3-5 3-6 3-7 3-8 3-9
4-0 4-1 4-2 4-3 4-4 4-5 4-6 4-7 4-8 4-9

6-0 6-1 6-2 6-3 6-4 6-5 6-6 6-7 6-8 6-9
7-0 7-1 7-2 7-3 7-4 7-5 7-6 7-7 7-8 7-9
8-0 8-1 8-2 8-3 8-4 8-5 8-6 8-7 8-8 8-9
9-0 9-1 9-2 9-3 9-4 9-5 9-6 9-7 9-8 9-9
```

Q19 – JAT6Ex19

Create a class named **JAT6Ex19.java**

Create the following output using **nested for loops** (an outer loop which contains an inner loop).



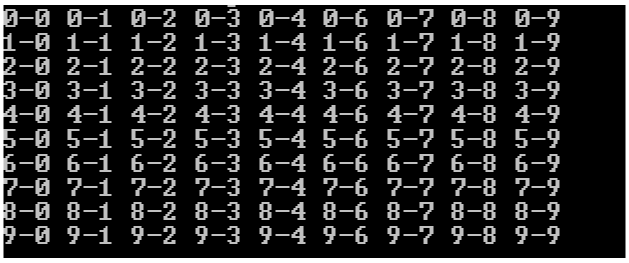
```
0-0 0-1 0-2 0-3 0-4 0-6 0-7 0-8 0-9
1-0 1-1 1-2 1-3 1-4 1-6 1-7 1-8 1-9
2-0 2-1 2-2 2-3 2-4 2-6 2-7 2-8 2-9
3-0 3-1 3-2 3-3 3-4 3-6 3-7 3-8 3-9
4-0 4-1 4-2 4-3 4-4 4-6 4-7 4-8 4-9
5-0 5-1 5-2 5-3 5-4 5-6 5-7 5-8 5-9
6-0 6-1 6-2 6-3 6-4 6-6 6-7 6-8 6-9
7-0 7-1 7-2 7-3 7-4 7-6 7-7 7-8 7-9
8-0 8-1 8-2 8-3 8-4 8-6 8-7 8-8 8-9
9-0 9-1 9-2 9-3 9-4 9-6 9-7 9-8 9-9
```

Please Turn Over

Q20 – JAT6Ex20

Create a class named **JAT6Ex20.java**

Re-write Q19 using **nested while loops** (an outer loop which contains an inner loop).



```
0-0 0-1 0-2 0-3 0-4 0-6 0-7 0-8 0-9
1-0 1-1 1-2 1-3 1-4 1-6 1-7 1-8 1-9
2-0 2-1 2-2 2-3 2-4 2-6 2-7 2-8 2-9
3-0 3-1 3-2 3-3 3-4 3-6 3-7 3-8 3-9
4-0 4-1 4-2 4-3 4-4 4-6 4-7 4-8 4-9
5-0 5-1 5-2 5-3 5-4 5-6 5-7 5-8 5-9
6-0 6-1 6-2 6-3 6-4 6-6 6-7 6-8 6-9
7-0 7-1 7-2 7-3 7-4 7-6 7-7 7-8 7-9
8-0 8-1 8-2 8-3 8-4 8-6 8-7 8-8 8-9
9-0 9-1 9-2 9-3 9-4 9-6 9-7 9-8 9-9
```

Q21 – JAT6Ex21

Create a class named **JAT6Ex21.java**

In this program, you will ask a user for their password. The passwords should be stored in a two-dimensional array (of type String).

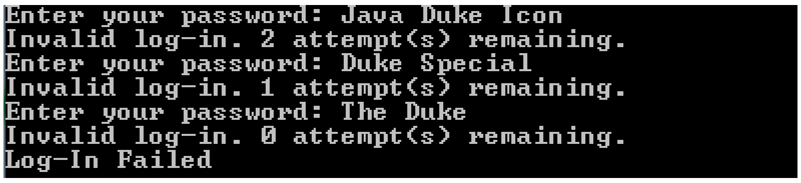
JavaDuke	JavaBean	TheNullPointer
StackOverflow	VirtualMachine	LossyPrecision

A user is given **three attempts** to enter a correct password.

In case of an invalid entry, the number of attempts remaining should be displayed to the user.

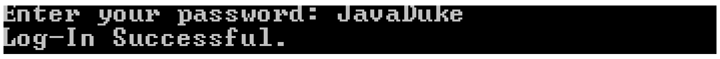
If the password is found, the search should immediately end.

A Scanner object should be used to facilitate user input.



```
Enter your password: Java Duke Icon
Invalid log-in. 2 attempt(s) remaining.
Enter your password: Duke Special
Invalid log-in. 1 attempt(s) remaining.
Enter your password: The Duke
Invalid log-in. 0 attempt(s) remaining.
Log-In Failed
```

The user enters three incorrect passwords.



```
Enter your password: JavaDuke
Log-In Successful.
```

The user enters a correct password on the first attempt. The program should not continue to search the array.

Please Turn Over

Q22 – JAT6Ex22

Create the following program, which attempts to parse a String to an integer.

```
public class JAT6Ex22{
   public static void main(String args[]) {
      String noOne = "one";
      int noTwo;
      noTwo = Integer.parseInt(noOne);
  }// main
}// class
```

If the String cannot be parsed successfully, an exception will be generated. What is its name?

Is it a checked or unchecked exception?

Does the Java compiler force you to handle or declare this type of exception?

Create a class named **JAT6Ex22.java**

Q23 – JAT6Ex23

Write a program, which asks the user to enter a number. Use a Scanner object to facilitate user input.

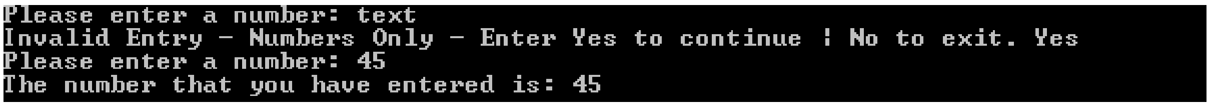
The user might input an incorrect data type. An exception will be thrown. What is its name?

Is it a checked or an unchecked exception?

Does the Java compiler force you to handle or declare this type of exception?

Include a try-catch block to catch this exception should it occur.

If an invalid entry is made, the user should be given the opportunity to re-enter correct data (unlimited attempts).



```
Please enter a number: text
Invalid Entry - Numbers Only - Enter Yes to continue ! No to exit. Yes
Please enter a number: 45
The number that you have entered is: 45
```

Create a class named **JAT6Ex23.java**

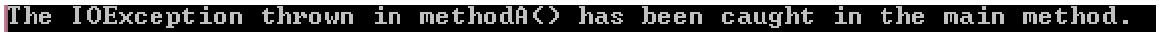
Please Turn Over

Q24 – JAT6Ex24

Create a class named **JAT6Ex24.java**. The class should contain the following two methods:

- the main method
- void methodA()

The main method should call methodA(). MethodA() should throw a java.io.IOException (use the **throw** keyword). methodA() should declare the exception and let the calling method (main) handle it using a try-catch block. The message, 'The IOException object thrown in methodA() has been caught in the main method', should be included in the catch block.



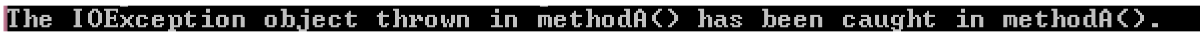
```
The IOException thrown in methodA() has been caught in the main method.
```

Q25 – JAT6Ex25

Create a class named **JAT6Ex25.java**.

Re-write the previous question (JAT6Ex24) so that methodA() handles (and does not declare) the IOException. The message, 'The IOException object thrown in methodA() has been caught in methodA()', should be included in the catch block.

The main method should call methodA().



```
The IOException object thrown in methodA() has been caught in methodA().
```

Q26 – JAT6Ex26

Create a class named **JAT6Ex26.java.**

In this program, you are asked to read in a line of text from a text file. The text file should be named, 'Java Blog.txt', and contain the following line of text:

- Duke is the name of the Java mascot.

The following methods should be created in the class:

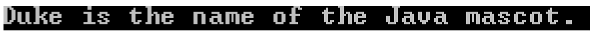
- the main method
- void readFromTextFile()

The main method should call the readFromTextFile() method. The readFromTextFile() method should read in the line of text and subsequently display it in the console. The readFromTextFile() method should handle any exception(s) generated using a try-catch-finally block.

Make sure that all file resource objects (FileReader and BufferedReader) are closed on completion of the file reading process. The main method should not handle or declare any exceptions.

The following objects should be used as part of the file reading process. You may wish to review the Java API for information on each one.

File
FileReader
BufferedReader



```
Duke is the name of the Java mascot.
```

Please Turn Over

Q27 – JAT6Ex27

Create a class named **JAT6Ex27.java.**

In this program, you are asked to write the following piece of text to a text file:

- I am preparing to sit the Java Associate Exam!

The following methods should be created in the class:

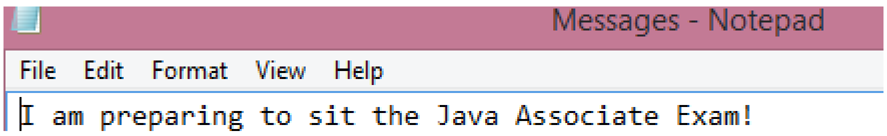
- the main method
- void writeToTextFile()

The main method should call the writeToTextFile() method. The writeToTextFile() method should write the specified piece of text to the file. The following message should also be displayed in the console, 'Message successfully written to text file.' The writeToTextFile() method should handle any exception(s) generated using a try-catch-finally block.

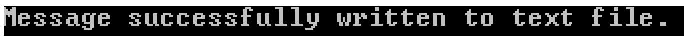
Make sure that all file resource objects (FileWriter, BufferedWriter, PrintWriter) are closed on completion of the file writing process. The main method should not handle or declare any exceptions.

The following objects should be used as part of the file writing process. You may wish to review the Java API for information on each one.

File
FileWriter
BufferedWriter
PrintWriter



```
I am preparing to sit the Java Associate Exam!
```



```
Message successfully written to text file.
```

Please Turn Over

Q28 – JAT6Ex28

Create a class named **JAT6Ex28.java.**

In the program, model the following call stack. Methods a() through e() have a void return type and are passed no arguments.

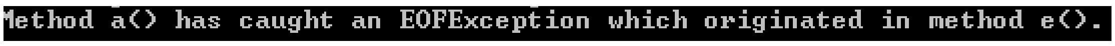
e()
d()
c()
b()
a()
main()

Method e() at the top of the stack, should declare that it throws an **EOFException** (end of file). Is this a checked or an unchecked exception?

In method e(), using the Java keyword **throw**, throw an exception of type EOFException.

Nominate method a() as the method that will handle any exceptions, which are propagated down the call stack.

In method a(), catch the exception and display the following message in the console - 'Method a() has caught an EOFException, which originated in method e().'



```
Method a() has caught an EOFException which originated in method e().
```

Please Turn Over

Q29 – JAT6Ex29

In this exercise, you are asked to create a program, which asks the user to enter the name of their favourite food. The program will determine whether the food is considered to be unhealthy.

Step 1:

Create an enum named *UnhealthyFoods*. The enum should be stored in a separate class file. The constants to be stored in the enum are {CHIPS, FRIES, BURGERS, BURGER, PIZZAS, PIZZA, SAUSAGES, PIES, PIE, CRISPS}.

Step 2:

Create a custom exception class named, *UnhealthyFoodException*. Make it a checked exception.

If thrown, this exception should return the following String, "That's an unhealthy food!"

Step 3:

Create a class named **JAT6Ex29.java.**

Create the main method and incorporate the following logic.

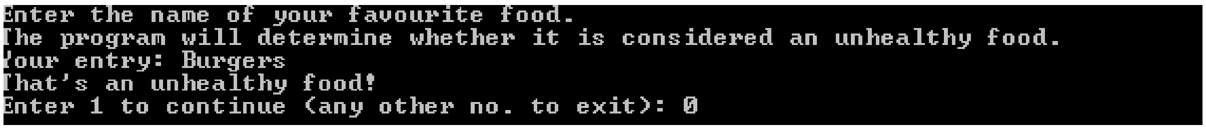
The user should be prompted to enter the name of their favourite food (using a Scanner object). The program should then determine whether the food entered is considered to be an UnhealthyFood (as per the enum). If considered unhealthy, an *UnhealthyFoodException* should be thrown and handled.

If the food type is considered to be a healthy food, the following message should be displayed:

- The program does not consider <<insert user entry>> to be an unhealthy food.

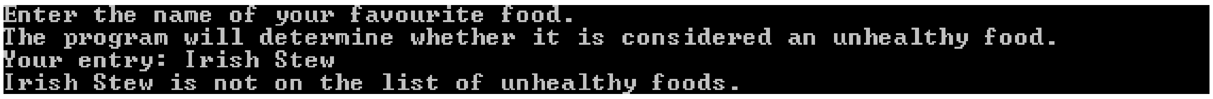
The user should be given the option of continuing (enter another food name) or exiting.

Program Demo 1: The user enters a value on the list of unhealthy foods.



```
Enter the name of your favourite food.
The program will determine whether it is considered an unhealthy food.
Your entry: Burgers
That's an unhealthy food!
Enter 1 to continue (any other no. to exit): 0
```

Program Run 2: The user enters a value not on the list of unhealthy foods.



```
Enter the name of your favourite food.
The program will determine whether it is considered an unhealthy food.
Your entry: Irish Stew
Irish Stew is not on the list of unhealthy foods.
```

Create a folder named JAT6Ex29 to store all class files.

End of Exercises

A programming challenge is outlined overleaf

Programming Challenge (Optional Question)

The following question is somewhat complex and should be seen as a programming challenge. You may wish to attempt it.

Duke's 2000 Calorie Daily Diet

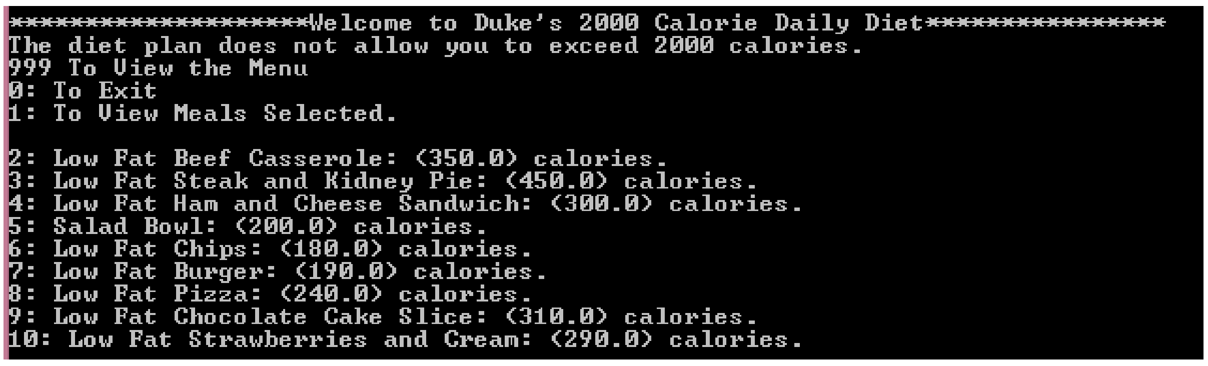
Overview:

Duke's 2000 Calorie Daily Diet, packs in as much good nutrition as possible, while trying to keep it simple, tasty and realistic. The user selects as many meals as they like, but cannot exceed 2000 calories.

The Application:

The program should incorporate the following functionality.

- The user should be presented with the following menu.



```
*********************Welcome to Duke's 2000 Calorie Daily Diet****************
The diet plan does not allow you to exceed 2000 calories.
999 To View the Menu
0: To Exit
1: To View Meals Selected.

2: Low Fat Beef Casserole: (350.0) calories.
3: Low Fat Steak and Kidney Pie: (450.0) calories.
4: Low Fat Ham and Cheese Sandwich: (300.0) calories.
5: Salad Bowl: (200.0) calories.
6: Low Fat Chips: (180.0) calories.
7: Low Fat Burger: (190.0) calories.
8: Low Fat Pizza: (240.0) calories.
9: Low Fat Chocolate Cake Slice: (310.0) calories.
10: Low Fat Strawberries and Cream: (290.0) calories.
```

- The user should be able to select an item from the menu. The program should be able to validate the user entry (in case they enter text instead of a number or a number outside of the specified menu range).

- The user should be able to select as many meals as they like (but only one at a time). The program should not allow the user to select meals where the accumulated calorie count would exceed 2000.

- The user can select the same meal many times.

- The program must be able to store the various meals entered by the user.

- If the user enters 1, they should be able to view a list of all the meals they have selected so far.

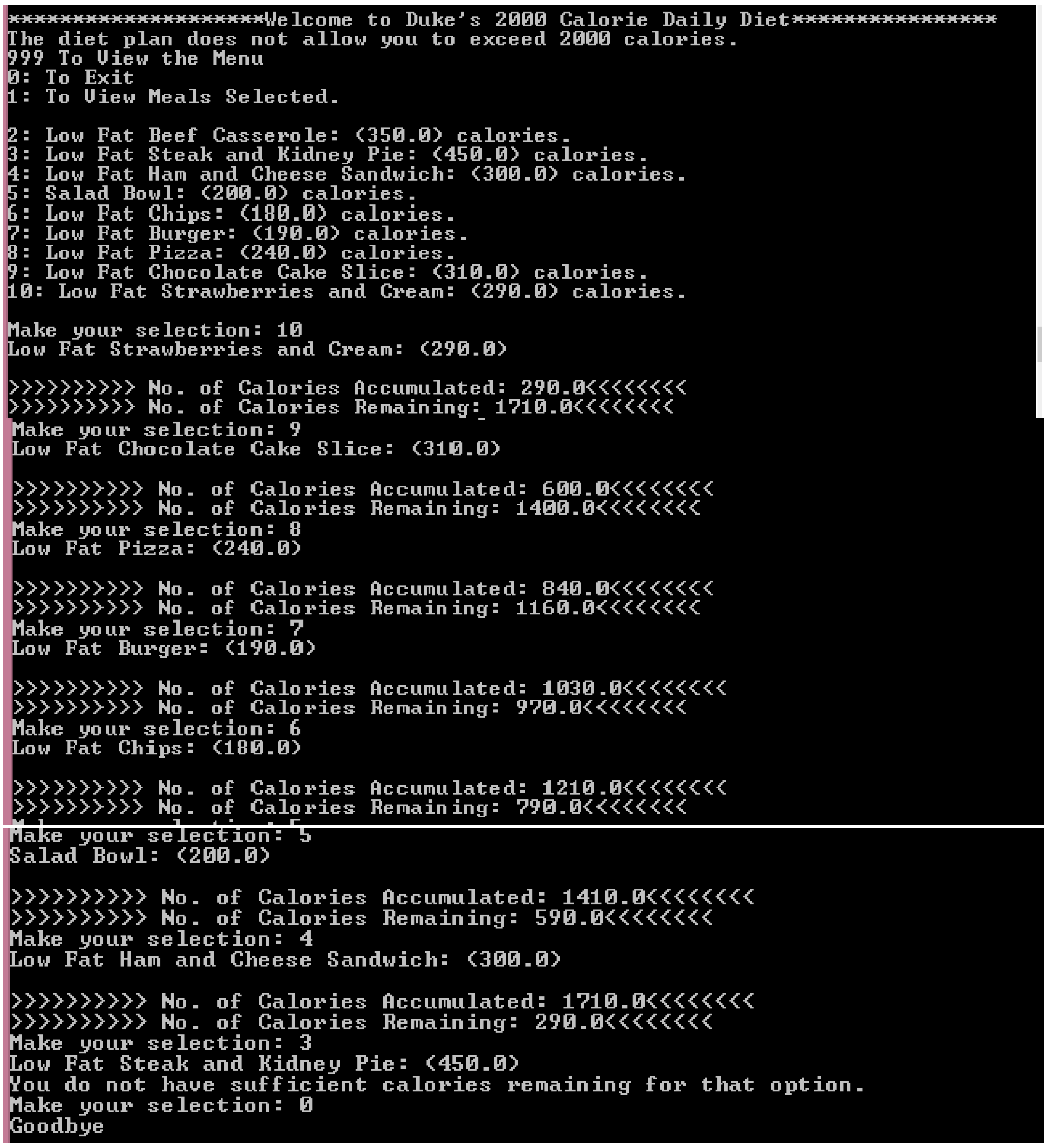
- The user cannot change their selections once entered.

- If the user enters 999, the main menu should be displayed.

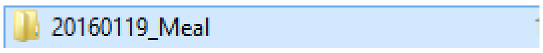
- If the user enters 0, they wish to exit the application. Before existing, the program must first determine whether any meals have been selected.
    - If no meals have been selected, the program should simply display the message, “Goodbye”.
    - If the user has made a selection / selections, they should be asked if they wish to save their entries to a text file.
    - If they wish to save, a folder should be created (in your project folder). The name of the folder should include today’s date. The program should be able to dynamically create the date. An example of the folder name would be:
        - 20160119_Meal
    - A text file should be created and stored inside the previously created folder. It should store the user’s meal selections. The name of the text file should also include today’s date. The program should be able to dynamically create the date. An example of the text file name would be:
        - 20160119_MealDetails

Please turn over

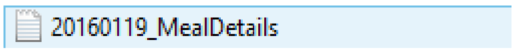
Here is a Program Demo:



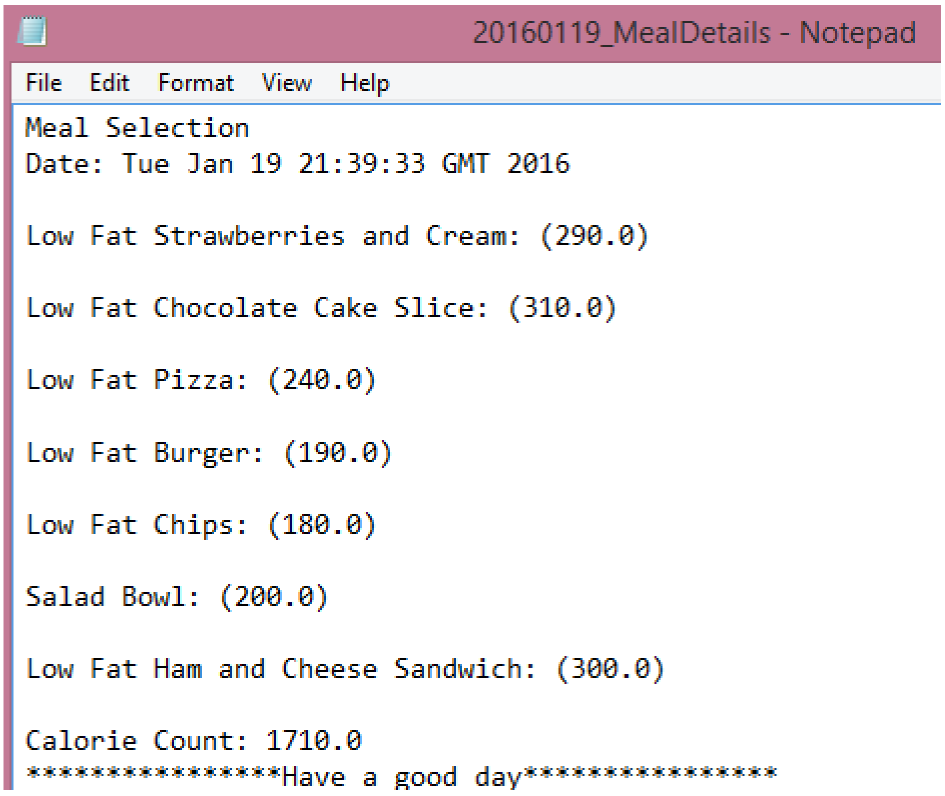
```
********************Welcome to Duke's 2000 Calorie Daily Diet****************
The diet plan does not allow you to exceed 2000 calories.
999 To View the Menu
0: To Exit
1: To View Meals Selected.

2: Low Fat Beef Casserole: (350.0) calories.
3: Low Fat Steak and Kidney Pie: (450.0) calories.
4: Low Fat Ham and Cheese Sandwich: (300.0) calories.
5: Salad Bowl: (200.0) calories.
6: Low Fat Chips: (180.0) calories.
7: Low Fat Burger: (190.0) calories.
8: Low Fat Pizza: (240.0) calories.
9: Low Fat Chocolate Cake Slice: (310.0) calories.
10: Low Fat Strawberries and Cream: (290.0) calories.

Make your selection: 10
Low Fat Strawberries and Cream: (290.0)

>>>>>>>>>> No. of Calories Accumulated: 290.0<<<<<<<<
>>>>>>>>>> No. of Calories Remaining: 1710.0<<<<<<<<
Make your selection: 9
Low Fat Chocolate Cake Slice: (310.0)

>>>>>>>>>> No. of Calories Accumulated: 600.0<<<<<<<<
>>>>>>>>>> No. of Calories Remaining: 1400.0<<<<<<<<
Make your selection: 8
Low Fat Pizza: (240.0)

>>>>>>>>>> No. of Calories Accumulated: 840.0<<<<<<<<
>>>>>>>>>> No. of Calories Remaining: 1160.0<<<<<<<<
Make your selection: 7
Low Fat Burger: (190.0)

>>>>>>>>>> No. of Calories Accumulated: 1030.0<<<<<<<<
>>>>>>>>>> No. of Calories Remaining: 970.0<<<<<<<<
Make your selection: 6
Low Fat Chips: (180.0)

>>>>>>>>>> No. of Calories Accumulated: 1210.0<<<<<<<<
>>>>>>>>>> No. of Calories Remaining: 790.0<<<<<<<<
Make your selection: 5
Salad Bowl: (200.0)

>>>>>>>>>> No. of Calories Accumulated: 1410.0<<<<<<<<
>>>>>>>>>> No. of Calories Remaining: 590.0<<<<<<<<
Make your selection: 4
Low Fat Ham and Cheese Sandwich: (300.0)

>>>>>>>>>> No. of Calories Accumulated: 1710.0<<<<<<<<
>>>>>>>>>> No. of Calories Remaining: 290.0<<<<<<<<
Make your selection: 3
Low Fat Steak and Kidney Pie: (450.0)
You do not have sufficient calories remaining for that option.
Make your selection: 0
Goodbye
```



20160119_Meal



20160119_MealDetails



```
Meal Selection
Date: Tue Jan 19 21:39:33 GMT 2016

Low Fat Strawberries and Cream: (290.0)

Low Fat Chocolate Cake Slice: (310.0)

Low Fat Pizza: (240.0)

Low Fat Burger: (190.0)

Low Fat Chips: (180.0)

Salad Bowl: (200.0)

Low Fat Ham and Cheese Sandwich: (300.0)

Calorie Count: 1710.0
****************Have a good day****************
```