OOP Exercise Book

TRAINING MATERIALS - EXERCISE

Beginner

Hello World!

1. Output “Hello World!” to the console.
2. Store “Hello World!” in a variable, then output it to the console.
3. Create a method that accepts a **String** as a parameter, and then outputs it to the console.
4. Modify your method to return a **String** once called, which you then use to output to the screen.

Maths

1. Create a method that accepts two integers as an input, then returns the integer that is the sum of these two parameters.
2. Create the following additional 3 methods that each take 2 parameters:
   a. **Multiplication** – which takes 2 numbers and returns the product.
   b. **Subtraction** – which takes 2 numbers, then returns the result of subtraction.
   c. **Division** – which takes 2 numbers, and returns the result of division.
3. Your Division method may have returned the wrong result; this was because we were using **ints** rather than **doubles**. Modify the Division method so that it takes **double** parameters and return a **Double** if not already.
4. With the Division method, the sum should only execute if the first parameter is smaller than the second, otherwise it prints out a message saying that the division cannot be performed.

Results

A person takes 3 exams in college, Physics, Chemistry and Biology, each exam can have a maximum of 150 marks. When all the exam marks have been added together, we can find the overall percentage that the person has got by multiplying their score by 100 and then dividing by 450.

1. Create the **results** class, this class has 5 variables, Physics, Chemistry and Biology, total and percentage. This class should also have 2 methods:
   a. **Method 1** - displays the results that the person got for each exam and then the total mark. Try to make the output neat and bespoke for each exam.
   b. **Method 2** - which finds and displays the percentage that the person received for the exams overall.
2. Expand the Results project, there is now a pass rate of 60% for the overall result, if the person receives under 60% they get a fail message.
3. Expand the previous example so that even if the person gets higher than 60% overall, if they fail 1 or more of the exams (pass grade of 60% for each exam) they still fail overall.
4. Expand the above so that the message that is displayed varies depending on the number of subjects that they have failed.

Conditionals

1. Create a method which accepts 3 parameters, 2 integers and a Boolean,

a. If the Boolean is true, the method will return a sum of the two numbers, and if it is false it will return the multiplication of the two numbers.

```
Input (1,2,true) -> 3
Input (3,3,false) -> 9
Input (1,1,true) -> 2
```

2. Recreate the flowchart in appendix A as a project. Ensure that your logic and outputs match that of the flowchart.
3. Modify your method from conditionals 1 to have another if statement that checks if one of the both the numbers are even. If this is true, return 0.

BlackJack

Given 2 Integer values greater than 0, return whichever value is closest to 21 without going over 21. If they both go over 21 then return 0;

```
play (10,21) -> 21
play (20,18) -> 20
play (1,22) -> 1
play (22,23) -> 0
```

Unique Sum

1. Given 3 integer values, return their sum. If one value is the same as another value, they do not count towards the sum. In other words, only return the sum of unique numbers given.

```
Input (1,2,3) -> 6
Input (3,3,3) -> 0
Input (1,1,2) -> 2
```

Taxes

1. Create the tax class, it contains 2 methods.
    a. Method 1 - which takes a salary and works out the percentage by which the salary will be taxed.
    b. Method 2 - which works out the exact amount that the user will be taxed using a similar process to that in the first method. This amount should be returned and output to the console.
    These 2 methods should be able to work independently so it is ok if some of the code is repeated.
2. The Salaries are taxed as below:
    a. 0 – 14999: 0% Tax
    b. 15000 – 19999: 10% Tax
    c. 20000-29999: 15% Tax
    d. 30000-44999: 20% Tax
    e. 45000+: 25%: Tax
3. Finally combine the 2 methods so that the second method utilizes the first method to return the correct result.

Numbers

1. Create a method that takes a number 10-99, and adds the two digits together

   E.G: 74 = 7 + 4 = 11

2. Create a second method that when given the number 1 – 99 returns a String representation of this number.

   For example: 1 = one, 11 = eleven, 21 = twenty-one

3. As above but 1-999.

4. As above, but 1-9999.

Iteration

1. Refer to Appendix B, recreate this flowchart as a project.

2. Refer to Appendix C, recreate this flowchart as a project.

3. Create a method that can print out the numbers 1-10 10 times each.

4. If you have used a while loop at any point in this exercise, replace them with a for loop– *remember that you should use a for loop when you know when the iteration should end.*

5. Create a method to print the numbers 1 to 10 as many times as the value of that number. For example, you will print '1' one time, and '10' ten times.

Numbers-Revisited

1. Use a for loop and print from 1 to 100 in words.

Coins

1. Given a cost and amount, work out the change given in specific coinage.

String Manipulation

The cost is £4.58 and the customer pays with a £20 note, therefore they would receive
- 1 £10 note
- 1 £5 note
- 2 20p's
- 1 2p
This information should be printed out in a form similar to that above.

1. For this task, you are to implement 4 methods that manipulate string objects. for all parts of this section you are only allowed to use the length, substring and the equals method. Avoid using arrays or methods you have yet to be taught as this defeats the purpose of the exercise.

    a. Method 1: When given a String, count and return how many words there are in that String.
    b. Method 2: When given a String, print out this String in a vertical fashion, each word on a different line.
    c. Method 3: As above, but reverse the order.
    d. A find method –to which we will send 2 strings, first is message and the second is the String you want to find within that message. If the second word appears in the message, then a Boolean value should be returned reflecting this.

Arrays

1. Create an array that will hold 10 integer values, populate the array with values, then call and output each element.
2. Create a for loop that populates an integer array with values, outputting them at each iteration. Then create another loop that iterates through the array, changing the values at each point to equal itself times 10, outputting them at each iteration.

```
1,1,1,1,1,1,1,1,1,1
2,2,2,2,2,2,2,2,2,2
…
10,10,10,10,10,10,10,10,10,10
```

Results-Revisited

1. Expand the results class so that the variables are now private and can only be accessed through public methods.
2. Modify these methods so that they do not allow the variables to be set to values that are less than 0 or greater than 150.
3. Now we have an issue in that the method that calculates the result will still run even if invalid data is entered, for example if the user tries to set the Physics mark to be 200 this is not allowed. But the defaults score of 0 can still be used in the method to calculate the results. Modify the class to stop this from happening.

Binary Conversion

1. Create a method that can take a Binary number (for example “101”, or “10101”) as a String and can print out or return the relevant decimal number.
2. Create a method that can take a decimal number (for example 14, 76) and print out or return the relevant Binary number, in the above example this would be (1110, 1001100).

OOP Principles

Inheritance

1. Create a class called Animal, Animal should have at least 2 methods of your choosing and 3 variables.
2. Now create 3 more classes that inherit from Animal, each of these classes should have their own methods and variables as well as the methods that they inherit from Animal.
    - Ensure that your inheritance hierarchy makes sense.

Singleton

1. Experiment and work with the Singleton design pattern.
2. Implement the design pattern.
3. Only create 7 objects of this one class.

Division with Exceptions

1. Create a program that can take 2 ints as input from the user and can produce the division of these numbers. You can use the Maths Project created earlier. This method will need to catch 2 specific exceptions as well as an overall third exception.

2. Handle each of these exceptions within the method and handle them in different ways.

3. Recreate the Division method to throw your own bespoke exception. This exception will be thrown if the user tries to divide a number (A) by a larger number (B).
    a. Create a separate class that will be your Exception, this class will extend the Exception class.
    b. Create and implement the Division method so that it takes account of your new exception.
    c. Handle this exception in varying ways to show the flexibility of exception handling.

Person

1. Create a Person class that models the following:
    a. Name
    b. Age
    c. Job Title
2. Create a method to return all three of these in a formatted string.
   HINT: Override the ToString() method.
3. Create some example objects with this class.
4. Create a List Implementation and store those object inside.
5. Use a stream to output all of your people to the console.
6. Create a method that can search for a specific Person by their name.

Garage

1. Using Vehicle as a base class, create three derived classes (car, motorcycle etc.), each derived class should have its own attributes in addition to the normal Vehicle attributes.
2. Using a List implementation, store all your vehicles in a Garage class.

3. Create a method in Garage that iterates through each Vehicle, calculating a bill for each type of Vehicle in a different way, depending on the type of vehicle it is. You can decide how this bill is calculated based on any attributes you see fit. (You do not get extra marks for making the calculation overly complex)
4. Garage should have methods that add Vehicle, remove Vehicle(s) (By ID, By Vehicle Type) fix Vehicle (Calculate bill) and empty the garage.

Paint Wizard

1. Create a paint requirement wizard that will calculate which of the following three products, would be the cheapest to buy, for the room you are painting.
    a. Work out which one wastes the least.
    b. Work out which is the best choice for any room (Cheapest).

```
1) CheapoMax (21Litre) £19.99
     This tin of paint will cover 10m² per Litre
2) ThatOneWithTheDog (40 Litre) £34.38
     This tin of paint will cover 12m² per Litre
2) AverageJoes (16 Litre) £17.99
     This tin of paint will cover 11m² per Litre
3) DuluxourousPaints (10 Litre) £25
     This tin of paint will cover 22m² per Litre
```

File I/O (Person Extended)

1. Using your Person class from earlier, create a list and populate it with 5 of these objects (Make up the values etc.).
2. Create a loop to iterate through the list, writing each object to one file (Think about how you format this)
3. Separately, create another list and populate it with the data in the file you just wrote too. (You're going to have to parse it back in the format you wrote it in).

Library/TDD

Create a library system with functionality to manage items and people within the library.

Rules:

1. Class diagram created and adhered to as much as possible
2. TDD Implemented.
3. At least one Abstract Class must be implemented.
4. At least one Interface Class must be implemented.
5. Each item should have at least 1 additional attributes unique to itself.
6. Method Overloading implemented.
7. Correct usage of access modifiers
8. Naming conventions adhered too
9. Commenting only where necessary.
10. At least 3 of the following items:
    a. Books
    b. Maps
    c. Government documents
    d. Media resources (Camera etc.)
    e. Newspapers

    f. Journals
    g. Magazines
    h. Dissertations
    i. Theses

11. **ALL** of the following:
    a. Check out item
    b. Check in item
    c. Add item
    d. Remove item
    e. Update item
    f. Register person
    g. Delete person
    h. Update person
12. Implement an ID system in your library project, utilizing a static integer variable.

Stretch Goal

Create a method to write the current library contents (Any items currently in the library) to a file, and another method to load them from a file into the library.

Anagrams

1. Create a method that reads words from a file, one per line, and stores them in an Array.
2. Create a method that takes a String as a parameter, and returns an alphabetically sorted version of the string back. (bac->abc).
3. Using these methods, perform the following;
    a. Find the word that has the most anagrams located in the list
    b. If two words have the same amount of anagrams, output the word that is longer.
    c. If two words have the same length and the same amount of anagrams, output both.

Prime Numbers

1. Create an algorithm that determines how many prime numbers are between 1 and 3 million.
    a. Extension: Have it finish running in under 2 minutes.
2. Create an algorithm that determines how many prime numbers are between 1 and 2 billion.
    a. Extension: Have it finish running in under 3 minutes.

Strings

1. Given two strings, write a program that efficiently finds the longest common subsequence.
2. Given two strings, write a program that outputs the shortest sequence of character insertions and deletions that turn one string into the other.

Maths-Revisited

1. Integrate a GUI into the Math project, it should incorporate all the features that you have been able to build into Math so far.
2. The GUI should have 5 buttons, +,-,/,* and =.
3. It should take 2 numbers, typed in by the user, and then return the answer of the calculation which is performed upon them.

4. The answer should be returned in a third text field.

Calculator Project

1. We need to create a GUI that represents a calculator, it should have buttons for the number 0 – 9, as well as buttons representing the 4 binary operators and =.
2. Should also have a clear button.
3. There should only be 1 text field in the calculator, this field should behave like a calculators screen, in that it displays output of the sums but cannot be used to input data into the calculator, this is done by pressing the buttons.
4. Some tips for the calculator project:

- This project should make use of the panel layout.
- When the actionEvent method is called it does not wait for another button to be pressed, it can only cope with one button at a time!
- Get the minimum requirements completed before you consider harder requirements.

5. This program can be easily extended and improved in lots of ways. And whether or not it is complete is up to debate.

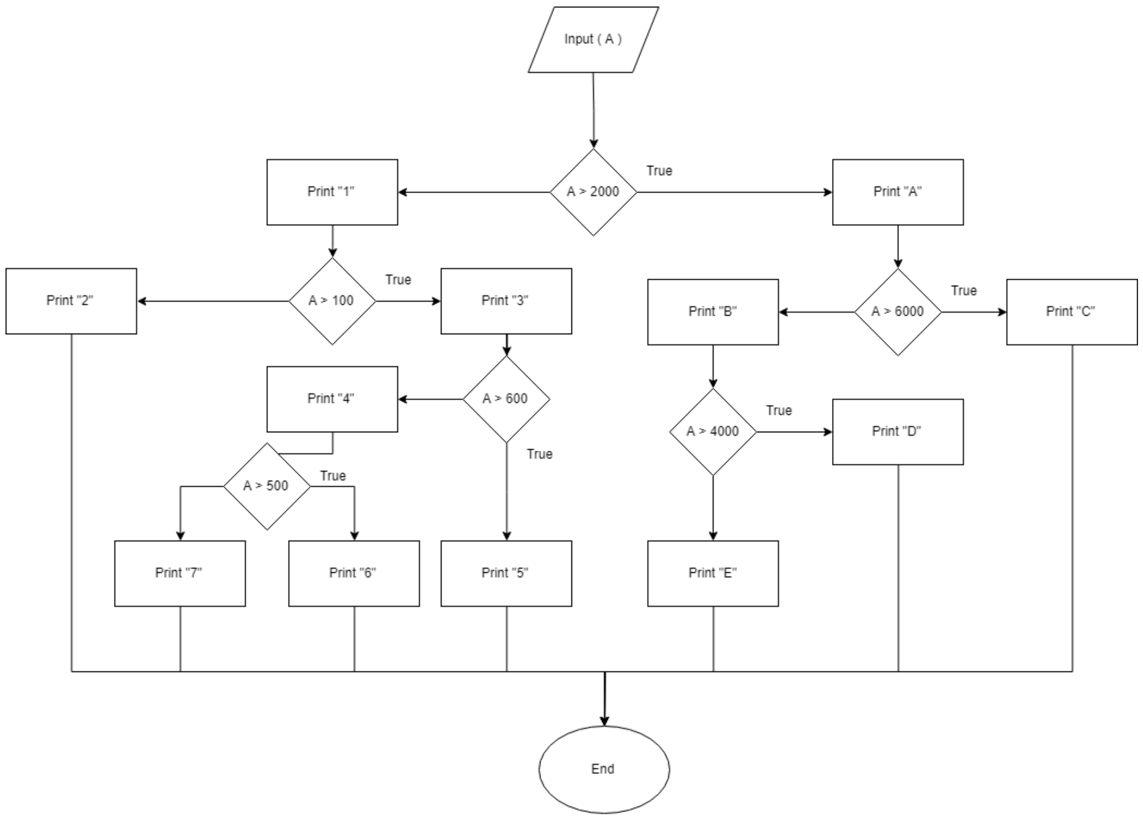
Maths Project - Take 2

1. This time the calculator should utilise radio buttons to complete the calculations.
2. Further to this the radio buttons should mean that only one can be selected at any one time. Based on which radio button is selected the results button at the bottom of the screen will calculate the result using the 2 numbers provided.

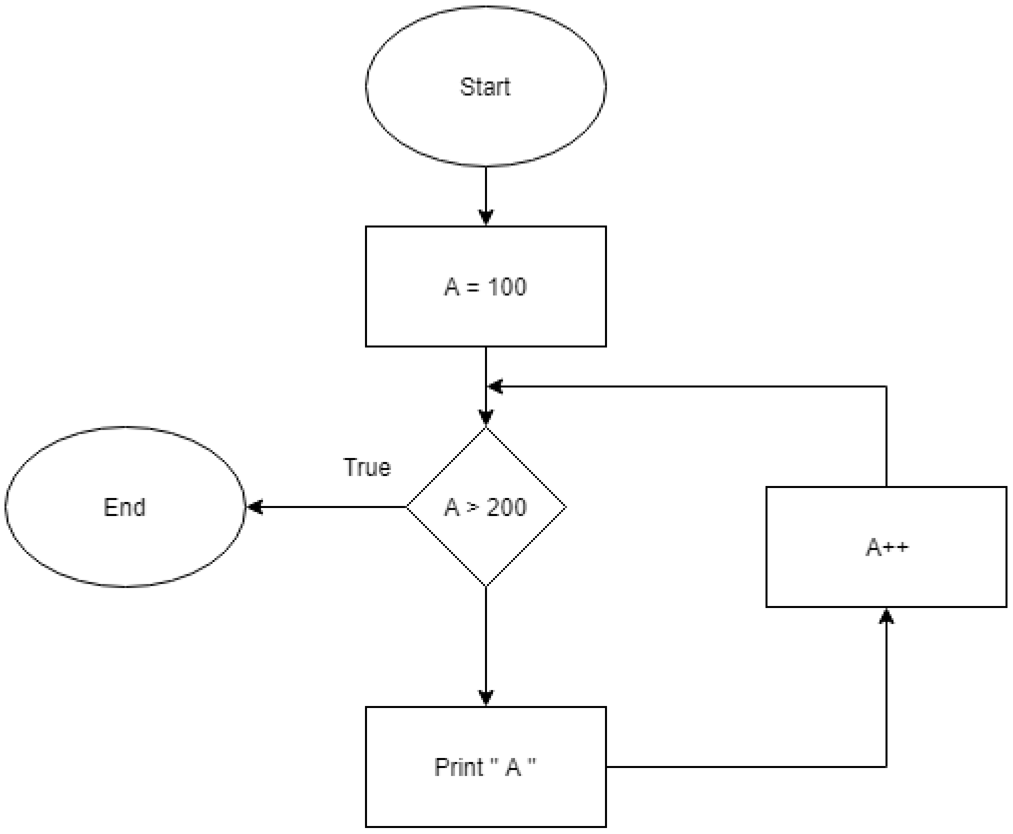
Calculator Project - Part 2

1. We would like a text field added at the bottom of the screen that displays help text when the mouse hovers over different buttons, for example when the mouse hovers over the '+' button it should provide different text compared to the '-' button.
2. Try to spread the buttons out a bit so that there is space between them.
3. Even though we want the user to enter numbers into this calculator using the buttons there is still the possibility that a user could type directly into the text field. We want to make it so that the user cannot use the text field in this way.
4. Ensure that the user can type numbers into the text field if they so desire, but no letters or other characters.

Appendix A


Input (A)
A > 2000
True
Print "1"
Print "A"
A > 100
True
Print "2"
Print "3"
A > 6000
True
Print "B"
Print "C"
A > 600
Print "4"
True
A > 4000
True
Print "D"
A > 500
True
Print "7"
Print "6"
Print "5"
Print "E"
End

Appendix B


Start
A = 100
A > 200
True
End
Print " A "
A++

Appendix C