EXERCISE 3, EXCEPTIONS

Exercise 3, Exceptions

Objective

The primary objectives for this lab are to be able to throw and catch exceptions and have a full understanding of the C# runtime's propagation mechanism for exceptions until a handler is found.

Overview

Firstly, you will handle the exceptions by throwing an exception object in your Account class if balance goes into red.

In a second practical you will enhance the Motorway simulator to throw an exception when the RegistrationFactory runs out of registration plates.

In the third part of this lab you'll handle any exceptions.

Part 1

Using try/throw/catch in the Account class

1. Create a new Console app called lab03.
2. Create a class called Account with the following fields and methods
   a. int id; double balance; string owner
   b. Create a constructor to set these fields
   c. void withdraw(double amount) // reduces the balance by amount
   d. void deposit(double amount) // increases the balance by amount
   e. void close()
      just display a message like "account 123 is closed" where 123 is the account's id.

   f. String GetDetails()to get details of id,balance and owner.
3. Back in the Main() method create an Account instance with a balance of £100.
4. Withdraw £50 and then display the account using it's GetDetails() method
5. Withdraw £60 and display the details of the account.
   Please note, there no withdrawal limit. Such an excellent account!
6. Fix the withdrawal() method so that it throws an `ArgumentException` when the balance is negative.

7. Catch this exception in **Main()** and display a suitable user friendly message.
8. Build and run the program now. Make sure you get the Account class to throw the exception and make sure that it is handled in **Main()**.
9. Add a **finally** clause to the exception handler in Main() to close the account no matter if there is an exception or not.

Part 2

10. Let's get back to the motorway simulator lab which you created in chapter 2.
11. The factory class will at some time run out of registration plates.
    How would you handle this situation?
    How would you inform the Vehicle class that we've ran out of registration plates?

    How would the Vehicle class inform the caller we cannot create a vehicle?
12. Implement the required exception mechanisms the best you can.

Part 3

13. Create a method in the Program class to append a message to a log file.

    Please refer to the fundamentals course (or search the web) if you cannot remember the code by heart

    ```
    private static void log(string msg){
    }
    ```
14. Call the log method from Main()
15. Logging a message should never crash your main application. Please catch any exception thrown by the **log()** method in Main().

** End **