Exercises

October 24rd 2017

This exercise sheet can be downloaded as PDF

This will be the first exercise lecture, where Christian is available to help all of you. Make use of this.

Exercise 1 - Exception handling to Stack

Earlier we implemented our own version of a stack. We chose to forego implementing any kind of exception handling. Today the task is to identify critical points of failure and introduce correct exception handling.

It is possible to use your own implementation or use my StackWithArray.cs and AbstractStack.cs

More specifically, introduce one or more stack-related exception classes. This will involve creating your own exception and making your stack throw these exceptions. Make sure to specialize the appropriate pre-existing exception class!

```
throw new MyException("Description of problem");
```

```
class MyException : ApplicationException{
  public MyException(String problem) : base(problem){
  }
}
```

Implement exception handling on:

- Push
  - Should throw an exception on a full stack
- Pop / Top
  - Should throw an exception on an empty stack
- ToggleTop
  - Should throw an exceotion if the stack is empty or only contains one element

Finally, implement a client program using the Stack and handle the exeptions that are thrown. It is possible to use below client:

```
using System;

class SimpleClient {

  public static void Main(){
```

```
    Stack s = new StackWithArray(10);

    Console.WriteLine("Empty: {0}", s.Empty);
    Console.WriteLine("Full: {0}", s.Full);

    s.Push(5); s.Push(6); s.Push(7);
    s.Push(15); s.Push(16); s.Push(17);
    s.Push(18); s.Push(19); s.Push(20);
    s.Push(21);

    Console.WriteLine("{0}", s.Size);
    Console.WriteLine("{0}", s.Top);
    Console.WriteLine("{0}", s);

    s.Pop();s.Pop();s.Pop();s.Pop();s.Pop();s.Pop();s.Pop();

    Console.WriteLine("Empty: {0}", s.Empty);
    Console.WriteLine("Full: {0}", s.Full);
    Console.WriteLine("Top: {0}", s.Top);

    Console.WriteLine("{0}", s);
    s.ToggleTop();
    Console.WriteLine("{0}", s);

  }

}
```

Exercise 2 - More exceptions in our Stack

In continuation of the previous exercise, we now wish to introduce the following somewhat ***unconventional*** handling of stack exceptions:

- If you push an element on a full stack throw half of the elements away, and carry out the pushing.
- If you pop/top an empty stack, push three dummy elements on the stack, and do the pop/top operation after this.

With these ideas, most stack programs will be able to terminate normally (run to the end).

Introduce this new behavior in another specialization of the stack class, which specializes Push, Pop, and Top. The specialized stack operations should handle relevant stack-related exceptions, and delegate the real work to its superclass. Thus, in the specialized stack class, each stack operation, such as Push, you should embed base.push(el) in a try-catch control structure, which repairs the stack - as suggested above - in the catch clause. See below:

```
public class UnconventionalStack : StackWithArray {
    ...
    public override void Push(Object el){
```

```
        try{
            base.Push(el);
        }
        catch (StackOverflowException){
            ...
        }
    }
    ...
}
```

Exercise 3 - Using streams

The purpose of this exercise is to train the use of the Read method in class Stream, and subclasses of class Stream.

Write a variant of the file copy program, see below. Your program should copy the entire file into a byte array. Instead of the method ReadByte you should use the Read method, which reads a number of bytes into a byte array. (Please take a careful look at the documentation of Read in class FileStream before you proceed). After this, write out the byte array to standard output such that you can assure yourself that the file is read correctly.

Are you able to read the entire file with a single call to Read? Or do you prefer to read chunks of a certain (maximum) size?

```
using System;
using System.IO;

public class CopyApp {
    public static void Main(string[] args) {
        FileCopy("path/to/file.txt", "path/to/destination/file.txt");
    }

    public static void FileCopy(string fromFile, string toFile){
        try{
            using(FileStream fromStream = new FileStream(fromFile,
FileMode.Open))
            {
                using(FileStream toStream = new FileStream(toFile,
FileMode.Create))
                {
                    int c;
                    do
                    {
                        c = fromStream.ReadByte();
                        if(c != -1) toStream.WriteByte((byte)c);
                    } while (c != -1);
                }
            }
        }
```

```
        catch(FileNotFoundException e){
            Console.WriteLine("File {0} not found: ", e.FileName);
            throw;
        }
        catch(Exception){
            Console.WriteLine("Other file copy exception");
            throw;
        }
    }
}
```