sOFTWARE 2 PRACTICAL

## Exceptions & Interfaces

Week 6 – Practical 6

### Problem:

Today’s problem is to refactor the code implemented in practical 5 for a counter tally. **Make sure you read the entre exercise before starting your implementation**. First, we would like third party users to use our code, therefore if you have not done so in the previous practical, you should move your code into a package named **tools**.

#### Step 1:

We need to think first of the design of the class before starting any implementation. What does a tally counter do?

1. We should be able to increment the counter by one (using one button)
2. We should be able to reset the counter to zero (using a second button)
3. Let assume that our counter has three digits (i.e. can count from 0 to 999). Once we have reached 999, the next click to increment the counter should throw a checked exception of type **InvalidOperationException** that you must implement yourself.
4. We should be able to read the value of the counter.

As stated in the previous practical, points 1-4 represent the behaviour of our type. To improve the design of our project, we may also want to create several types of tally counters, all with the same behaviours. Create an interface **ITallyCounter** that encapsulate all the behaviour from points 1-4.

* Create a new class **BasicTallyCounter** which implement **ITallyCounter**.
* The client wants to be able to create tally counter object with 3 or more digits. For example, a counter using 5 digits can go up to 99999. No counter should have less than 3 digits. In addition, the client would like to have counter with 3 digits as a default. Implement the necessary constructor(s).
* Implement the method toString(). The String returned should be of the form 001, 011 or 999. If you cannot find a way to do that (even after making a search on the internet), just return a String of the form 1, 11 or 999.

#### Step 2

A year later we want to update our package to provide more functionality. We must ensure backward compatibility, that is all programs that used our original package should still work after updating to our new version. We must not modify the interface **ITallyCounter** and the class **BasicTallyCounter**.

In the new package we should be able to create advanced counter where we can decrement the counter by one. Note that the counter cannot become negative, so if the counter is already at 0, decrementing the counter should throw a **InvalidOperationException**.

* Implement an interface **IAdvancedCounter** to cover all the functionality of an advanced tally counter. Can we reuse the interface from step 1?
* Implement a class **BetterTallyCounter** that implements the interface **IAdvancedCounter**.

#### Step 3: self-reflection

Let’s assume I want to store **BetterTallyCounter** and **BasicTallyCounter** objects into a single collection, for example an ArrayList. In addition, I would like to compute the sum of all the tallies within that collection. Is your design able to do so?

Discuss your design with your peers, the GTAs and myself.