Stack

Dr. Matthias Hölzl

June 7, 2019

Goals

- Create a simple `Stack` data structure using TDD
- Proceed incrementally, baby steps
- This exercise is meant to demonstrate
  - how to develop features `incrementally` using TDD
  - how to write good tests that help you to grow your design
- Therefore: **Please do not look ahead in this document!**
  Only move to the next page after having finished the current exercise.

Stack

- Clone the repository at `https://github.com/hoelzl/StackCpp.git`
- Open the contained project in CLion (or your IDE/editor of choice)
- Ensure that you can build the target `StackTest` and that the test fails when you execute it

- Solve simply!

Stack

- Implement a `Stack` data type for integers with the following signature:
- `void Push(int element)`
- `int Pop()`
- `bool IsEmpty()`
- If the stack is empty, `Pop()` should throw an exception of type `std::out_of_range`

Extension (1)

- Extend the `Stack` data type with a method

  `int Size()`

  that returns the number of elements on the stack

Extension (2)

- Extend the `Stack` data type with a member function

  `int Count(int element)`

  that counts the occurrences of *element* on the stack

Extension (3)

- Extend the `Stack` data type with a member function

  `int PopDefault(int default)`

  that acts like `pop()` when the stack is not empty and returns `default` when the stack is empty

Extension (4)

- Extend the `Stack` data type with member functions

  ```
  void SetDefault(int default)
  void ClearDefault()
  ```

  After `SetDefault(default)` has been called, `Pop()` should act like `PopDefault(default)`. When `ClearDefault()` is called, `Pop()` should revert to its original behavior, i.e., throw an exception when the stack is empty

Extension (5)

- Our `Stack` class will now be used on an embedded system where dynamic memory allocation at runtime is not allowed. Therefore we have to replace its implementation with one that does not use `std::vector`.
  Modify your implementation so that it uses a `std::array<int, 16>` to store its elements. Make `Push()` throw an exception of type `std::out_of_range`