Stack
(Short Version)

Dr. Matthias Hölzl

May 4, 2022

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 task after having finished the current one.
- (Also note that the stack implemented in this workshop has several pretty bad design flaws and should not be used for anything but demonstrating TDD.)

Stack

- Open the `Stack` project in 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) // Add element to the top of the stack`
  - `int pop() // Remove the topmost element of the stack and return it`
  - `bool is_empty() // Check whether the stack is empty`
- If the stack is empty, `pop()` should throw an exception of type `std::out_of_range`

Hints:

- Use a member variable `std::vector<int> elements` to store the elements of the vector
- You can use the following functions of `std::vector<int>`:
  - `elements.push_back(element)` to add `element` to the end of `elements`
  - `elements.back()` to get the last element in the vector of elements
  - `elements.pop_back()` to remove the last element in the vector of elements
  - `elements.empty()` to check whether the vector of elements is empty

Extension (1)

- Extend the `Stack` data type with a method

  `std::size_t size()`

  that returns the number of elements on the stack
- Can you simplify your tests using the `Stack::size()` member function? If your tests use implementation details, can you remedy this using `Stack::size()`? If so, refactor your tests.

Hint:

- You can use `elements.size()` to get the number of elements in `elements`

Extension (2)

- 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`.
  An example implementation is available at `https://tinyurl.com/2kcn7v8f`.
  How many of your test stop working if you replace your implementation with the one from the link?
  Test that `push()` throws an exception of type `std::out_of_range` if the stack is full.

If you have time left and nothing better to do...

Bonus Content

Extension (3)

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

  `std::size_t count(int element)`

  that counts the occurrences of *element* on the stack

Extension (4)

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

  `int pop_default(int default_value)`

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

Extension (5)

- Extend the `Stack` data type with member functions

  ```
  void set_default(int default_value)
  void clear_default()
  ```

  After `set_default(default_value)` has been called, `pop()` should act like `pop_default(default_value)`. When `clear_default()` is called, `pop()` should revert to its original behavior, i.e., throw an exception when the stack is empty.