CSE326: Software Engineering

Assignment 2 - Design Patterns and OOD

Cole Johnson - cole.johnson@student.nmt.edu

April 9, 2025

Problem-Solving

(1.) You are designing a Time and Talent Survey website for a local church. Suppose the administrator would like to be notified whenever a new survey is completed. What design pattern is suggested? Why did you choose this design pattern? (15 points)

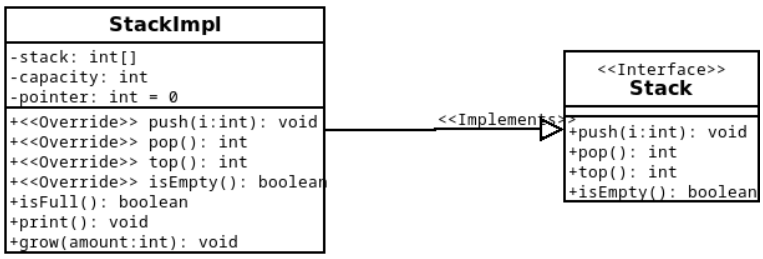
The observer / subject-object pattern would be an ideal design pattern to choose for this problem. I choose this design pattern because it's an effective tool when you have two aspects of a system that are both dependent on the other. Here, the subject could potentially be the survey itself and participants could act as observers.

(2.) You are developing a graphics editor, within which shape can be basic or complex. An example of simple shape is a line object; an example of a complex shape is a rectangle object. The rectangle object consists of four-line objects. Because all shapes have many common operations and can be represented in the hierarchy, you wish to treat all shapes uniformly. What design pattern is suggested? Why did you choose this design pattern? (15 points)

The composite pattern would be an ideal design pattern to choose for this problem. This design pattern is good when you want to represent part-whole hierarchies–components that are composed of simpler items. Line objects could be the leaf or node while larger shapes make up the composite.

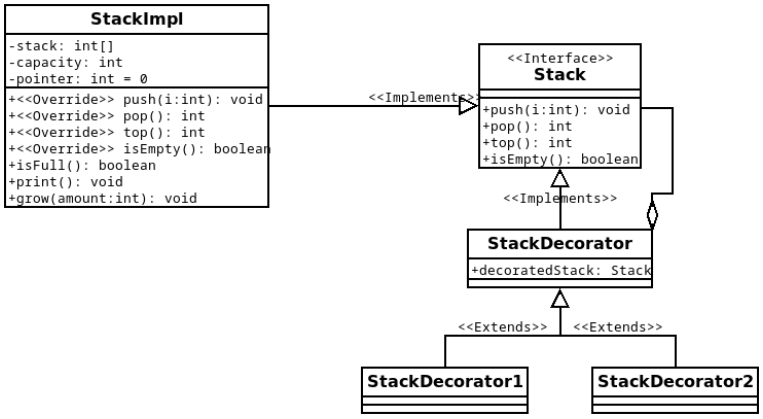
Stack Implementation

The stack implementation just has two parts: an array and a pointer to the next unoccupied address inside the array. To push, just use the pointer to assign a new element and increment the pointer. To pop, get the value that is pointed to, decrement the pointer, and return the value. We need some additional helper functions not described in the interface, like `grow()` and `isFull()` to help us grow the stack when needed.



Decorator Pattern

To make the decorators, I implemented an abstract class called `StackDecorator`. The abstract class just holds a reference to the decorated stack, called `decoratedStack`. Two concrete decorators, `StackDecorator1` and `StackDecorator2`, subclass the abstract decorator. Both the classes will call their specified behavior and then just delegate the task of the task to `decoratedStack`.



Here's the finally tested output using the driver code:

```
8:56:50 PM: Executing ':stack.TestDriver.main()'…

> Task :compileJava
> Task :processResources NO-SOURCE
> Task :classes

> Task :stack.TestDriver.main()
StackDecorator1 Test:
push()
push()
top()
pop()
isEmpty()
StackDecorator2 Test:
TRUE
TRUE
TRUE
FALSE
FALSE
TRUE

BUILD SUCCESSFUL in 351ms
2 actionable tasks: 2 executed
8:56:50 PM: Execution finished ':stack.TestDriver.main()'.
```