EECS 293
Software Craftsmanship
2016 Fall Semester

# Programming Assignment 11

Due at the beginning of your discussion session on

November 14-18, 2016

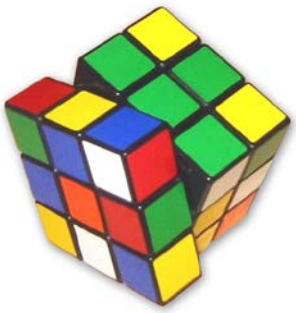
## Reading

- Section 6.2 (skip "Present a consistent level of abstraction in the class interface" and "Provide services in pairs with their opposites") in Code Complete
- Section 6.4 in Code Complete
- Items 13, 14, 15 in Effective Java
- Section 19.6 in Code Complete

## Programming

In this assignment, you will formulate the object-oriented architecture of a Rubik's cube simulator. As you have learned in Section 5.1, you fully expect the design process to be sloppy, non-deterministic, based on heuristics, and iterative (and possibly very frustrating). However, you also shoot for a product that is tidy, clearly addresses the relevant trade-offs, priorities, and restrictions: in other words, it will be a beautiful architecture.

For those of you unfamiliar with the puzzle, a Rubik's Cube® comes in the form of a cube where each face is divided into three rows and three columns (nine "squares"). Any of the six faces of the cube may be rotated either clockwise or counterclockwise, which also rotates the three nearest squares on each adjoining face onto a new face, respectively. When solved (or taken from the factory packaging), each face of the cube contains squares of only one color. There is no way to change the relationship between the colors of the central squares on each face.



The class `RubiksCube` should support methods to:

- Create a new cube in a default or in a given configuration
- Get the current cube configuration
- Display a textual or graphic representation of the cube
- Rotate one face of the cube

Additional support classes and methods may be needed to complete the project. No method is needed to solve the cube, since separate algorithms are known for this problem.

## Design

Create a design document that describes your architectural decisions. Your document can optionally contain sketches and diagrams of your architecture. Your objective is that your design document should be used as a blueprint for the implementation. You can define classes or interfaces as you see fit. Commonalities among classes or interfaces should be expressed through inheritance or containment. For each class or interface, you should describe at least the abstraction it captures, its position in the inheritance hierarchy, the signature of its constructors and public methods, its private data structures (if any), and the pseudo-code of any complicated method. You can outline your architecture for error handling and your approach to testing.

You are not required to submit an implementation, which is the main topic of the next assignments.

## Discussion Guidelines

The discussion will focus on class design.

## Submission

Submit design documents that describes your architectural decisions. No implementation is required: you will implement your design in the next programming assignments. This assignment can be submitted on Blackboard.