Jacob Alspaw
EECS 293 - Hamm
Assignment 11
14 November 2016

Rubix Cube Design Document

Input

Input will be read from stdin and will expect either no input for a randomly generated cube or the below format for a predetermined cube.

Color Color Color
Color Color Color
Color Color Color

Color Color Color
Color Color Color
Color Color Color

Color Color Color
Color Color Color
Color Color Color

Color Color Color
Color Color Color
Color Color Color

Color Color Color
Color Color Color
Color Color Color

Color Color Color
Color Color Color
Color Color Color

Defense Design

This program will have a strong barricade that is a class validating the input. The validating class will throw exceptions according to whether or not the data is presented correctly. Inside the barricade, all parameters will be checked for non-null pointers and assert checked for expected input. Preconditions, post conditions, and outputs will have checks throughout to make sure data isn't being mishandled and error does not propagate. Particular validation checks will be discussed in the InputValidator. All other classes assume valid and correctly formatted data.

Testing Design

The testing classes will have complete branch coverage while also covering cases of structured basis, boundaries, compound boundaries, bad data, good data, and stress tests.

Hierarchy / Abstraction Design

Color

An enum class that will represent the colors on the rubix cube. The color enum class will contain the following enums: BLUE, GREEN, RED, YELLOW, WHITE, and ORANGE.

Direction

An enum class that will represent the directions to rotate the rubix cube. The direction enum class will contain the following enums: COUNTER_CLOCKWISE, CLOCKWISE

CubeFace

A private static nested class that will represent a face of the rubix cube. Its fields will include a 3x3 2D array of Colors. Each CubeFace is distinguished by the middle tile.

Constructor Signature

The constructor for the CubeFace will expect an array of all nine colors to be included. The order from which they will be just transferred from 1D array to 2D array is top to bottom, left to right.

Methods

- getter methods for fields
- equals method will override Object's equals method to equate CubeFaces
- toString method that will print out the 2D-array in a grid format that is identical to a single face in the input

RubixCube Class

A class that will represent the rubix cube. Its fields will be an array of six CubeFaces and the current CubeFace that the user is viewing.

Constructor Signature

The constructor for the RubixCube will take a two dimensional array of 6 strings that have been validated for formatting. The constructor will make six CubeFaces from these strings. The CubeFaces array field will be initialized to these new CubeFaces. If the incoming array is empty, then the constructor will initialize the CubeFaces array field to a random permutation.

Methods

- getter methods for fields
- randomCube method will generate a random permutation of the rubix cube
- toString method will print out the current cube configuration in the input format

- rotate method will rotate a given CubeFace in a certain direction and adjust every other CubeFace according to the direction

InputValidator

A class that will serve to interpret the input. The class will ensure that the input follows a correct format such that only allowed colors will be used, six sides have been given, there are exactly nine tiles per color, the middle tile is different for each face, etc.

Constructor Signature

None, the class will contain all static methods.

Methods

- cubeFaces has no input parameters. The method will read the input, validate the input, then interpret the input into a two dimensional array in a format that can be understood by the RubixCube class. Returns this two dimensional array
- validate will take the user input as an array of strings. The method will use as many private helper methods as necessary to validate against individual sources of error. Will pass on an error thrown in helper methods
- additional helper methods that will validate formatting and data quality. The methods will throw an error with a flag that correlates to the bad data