Object-Oriented Design
COMP 2000
Spring 2024

Lab 10.1

Mastermind Part II

Based on the instructions in Part I, you should have a rough idea of how a *Model-View-Controller* approach to this problem would be organized. In the following specifications I will list and describe the minimal classes necessary in each of these parts of the design of the application (you may want to add others, or discover the need for others as you proceed).

Model

The main class in the model will be the `MMGame` class. The `MMGame` class will have a constructor that instantiates an instance of the game. The constructor will be parameterized with at least three things: the number of pegs, the number of colors, and a secret pattern whose length is the number of pegs and which consists of no more than the number of colors distinct values. Apart from knowing these three things, the `MMGame` class also keeps track of how many guesses have been made, what the most recent guess was, and whether the pattern has been discovered.

The `MMGame` class also has certain behaviors. It should allow guesses to be made, unless the most recent guess was a correct solution, in which case it will no longer accept new guesses. It should also be able to provide feedback on the most recent guess: how many were the correct color in the correct location (aka complete), and how many were the correct color in the wrong location (aka partial). Finally, the `MMGame` class is able to report its current state (with the exception of the secret pattern), one attribute at a time. In other words, almost all of the state variables should have query methods (getters).

View

The user interface of the Mastermind application will consist of a window in which are found a collection of widgets that represent the visual elements of the game. This window should be represented by a subclass of `JFrame` called `Mastermind`. You may use any Swing components, but I suggest keeping it simple to start and gradually adding more features as you feel comfortable and have time. In the first version of the GUI it would be a good idea to fix the number of pegs and number of colors as constants (but do so in a way that will make it easy to change this later).

Minimally, your `Mastermind` class should have the following:

1. **The user should have the ability to select a guess by choosing the pattern of colors.**
2. **There should be a button to submit the guess.**
3. **There should be a visual collection of all the guesses so far and their feedback.**
4. **There should be some visual representation of how many guesses the user has remaining.**
5. **When the pattern is found or the number of guesses runs out, there should be some mechanism that communicated this to the user.**

It would be entirely appropriate to make parts of the GUI into classes that are inherited from Swing components in order to represent different aspects of the game.

There are many other options that you could add to the view. The ability to set the number of colors or pegs; the ability to reset the game, or to play again; allowing blank to be a color; allowing duplicate colors in a pattern; etc. Be creative.

Control

Each button should have a listener object that carries out the appropriate tasks. [It is acceptable practice to have the main application class's constructor be part of the view model and contain inner classes that represent the control. This is an implementation detail, however, and should not impact your UML diagrams.]

To Do

Now that we have a basic analysis, it is time to do the detailed design. In other words we need to think about the methods—including signatures and specification (preconditions/postconditions), the instance variables (at least the ones relating to the state required to support external behavior), and the interfaces (public components) of each class. We also need to think about how the classes fit together statically (class diagrams).

Complete class diagrams for the Mastermind application. In addition, prepare a written document that details the specification of each user-defined class (at least MMGame and Mastermind, but also any other classes that are substantially new – simple subclasses of Swing components don't count). By specification I mean the public interface of each class along with its contract.

We will spend one week on Part II. The more you have complete by the end of the first lab period the better off you will be for wrapping it up during the following lab. Part III will be available in one week.

Your diagrams should be created them digitally. You can use Microsoft Word, or yuml (https://yuml.me), or SmartDraw (https://www.smartdraw.com/, 7-day free trial), or any other online tool that you find useful. Your specification document should be a Word or PDF document.