Chain of responsibility: Here, each violation is passed to the next object in the chain. This happens until it reaches the Headmaster, which is the last class in the chain of responsibility. This program uses print statements to document the violation being passed up this chain.

Command: Here, there is a list of commands that are executed in order to “protect” an instance of the Wizard class (In this case it is Harry Potter.) These commands are then iterated through and each one is passed into the execute() method, which will perform an action to “protect” Harry based on a set of conditionals located in the execute method. If a command is not able to be executed to “protect” Harry (the preconditions in the execute() method are not satisfied,) the program will throw a UnableToProtectHarryPotterException.

Flyweight: I could not get the flyweight to work, but after looking over the code, it seems as though there are different images provided, and the user can specify a certain picture and border type in the main method and the program will render the chosen image with the specified border. If an image isn’t available, it will throw an ImageNotAvailableException.

Interpreter: I could not get the interpreter to run, but after looking through the code, it seems as though there is a file with “magic phrases” that the classes parse into executable instructions with the help of a perl file that injects code into different java class files of the program to assist in making the parse tree.

Proxy: Here, there is an instance of DarkLord created in main, as well as an instance of DarkWizard. The DarkWizard instance is used to “talk” to the DarkLord instance (in this case it is Voldemort,) instead of the user being able to “talk to” (access) the DarkLord instance directly.

State: Here, there are different “states” of teaching for “defense against the dark arts” teachers. In the main, instances of the different years of teaching can be instantiated, and the state can be switched to any of the newly instantiated teaching years. This allows the user to easily switch between different years of DADA teaching without exposing the structure of the program.