Exercise - Write a Proxy that implements schedule-based access (25 minutes)

1. Create an interface with a few methods on it (the methods can be anything you want).
2. Write a class that implements your interface (the method implementations can be anything you want).
3. Write a proxy class that can be used to control on what days of the week and at what times of day methods can be called on instances of the class created in step 2. Your proxy class should have the following properties:
    a. AllowedDays: A list of days of the week on which method calls are allowed (e.g., M, T, W, Th, F)
    b. AllowedTimeRange: A time range during which methods calls are allowed (e.g., 8 am - 5 pm)
    c. When a method is called on the proxy, it should check the current day and time. If the method call is allowed, it should be passed to the "real object' and the result returned to the caller. If the method call is not allowed, the proxy should throw an exception at the caller.
4. Write a small program to test your proxy and demonstrate that it works.
5. Zip up and submit your code on Canvas.

Exercise - Write a Proxy that implements lazy loading (25 minutes)

1. Create an interface named Array2D that represents a two-dimensional array of integers and contains the following methods:
    a. A method that sets the value of an array element: Set(row, col, value)
    b. A method that returns the value of an array element: Get(row, col)
2. Create a class that implements the Array2D interface and that also has the following methods:
    a. A constructor that lets the caller specify the dimensions of the array.
    b. A method that saves the object's state to a file: Save(fileName)
    c. A method that loads the object's state from a file: Load(fileName)
3. Write a program that creates a 2D-array object and saves it to a file. This file will be used in the next step.
4. Write a proxy class that implements lazy loading for your 2-D array class.
    a. What is lazy loading? Suppose you have an object stored in a file or database. If the object is large, it will be time-consuming to load, and you might not want to incur the expense of loading the object into RAM until the program actually uses the object. This way, if the program never uses the object, you will not waste time loading it.
    b. An elegant way to implement this idea is with the Proxy pattern.
        i. Create a proxy that implements the same interface as the large object
        ii. The proxy class should also have a constructor that accepts the name of the file that stores the object
        iii. When the program begins, create a proxy object that represents the large object. Initially, the proxy will contain a null pointer to the "real object", because it hasn't been loaded yet.

    iv. The first time a method is called on the proxy, it should load the large object from file or database into RAM, and store a pointer to the loaded object.
    v. After loading the object, all method calls on the proxy should be delegated to the “real object” that was loaded.
5. Write a small program to test your proxy and demonstrate that it works.
6. Zip up and submit your code on Canvas.