Design Patterns:

Homework Question: **60** pts

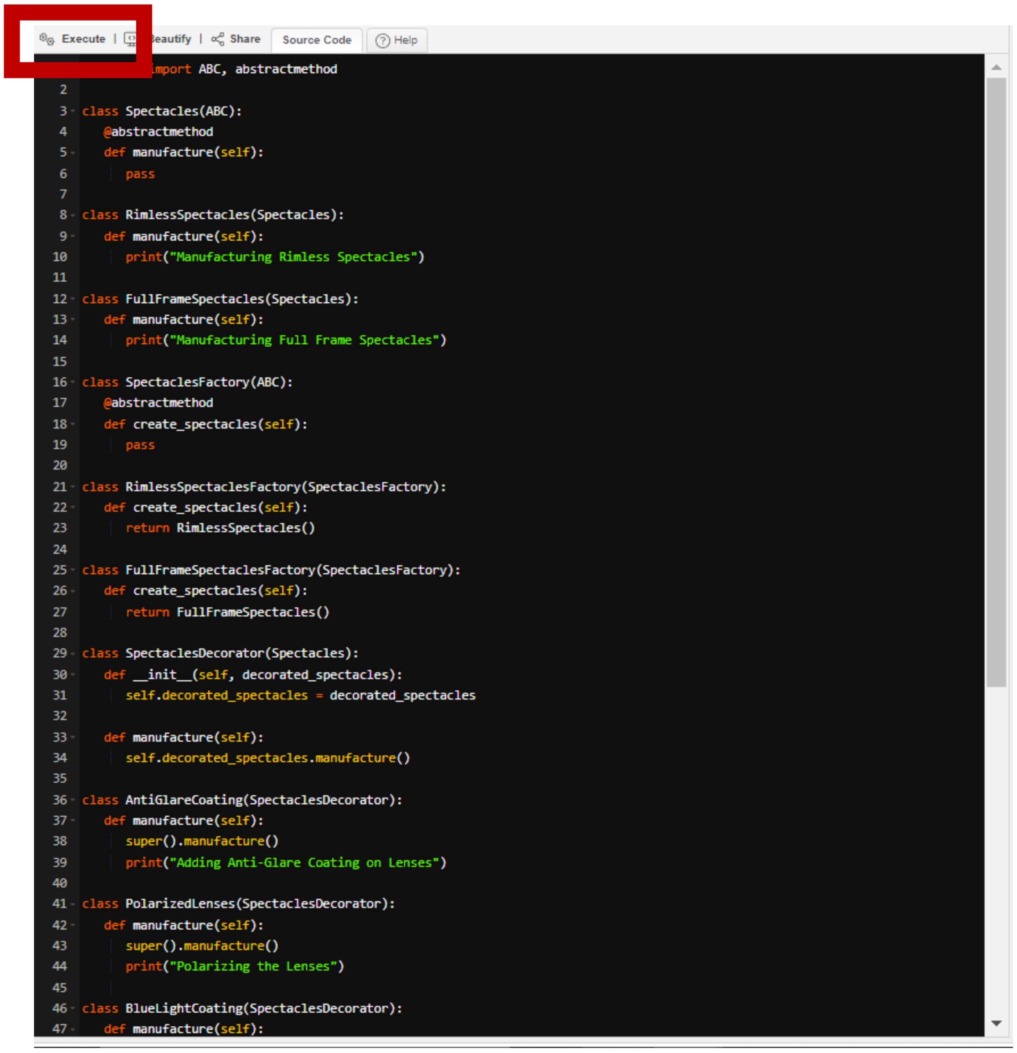
1. write code in an **Object-Oriented** programing language, e.g., Java, Python, or C++, for the following:
   a. choose one of design patterns from **Creational** catalog and implement the pattern.
   b. choose one of design patterns from **Structural** catalog and implement the pattern.
   c. implement a function that will use both 2 design patterns from in 1.a) and 1.b).
   d. implement a main/test function to run/test the code from 1.c).
2. in your instructions, define input argument(s) and the expected result for the function from 1.c) and 1.d).
3. include the following 4 items in your HW submission:
   a. your source codes (2 design patterns from 1.a), 1.b); the function from 1.c), and the testing function from 1.d) in txt format. please indicate the names of design patterns in your source code.
   b. a screenshot of code got compiled, executed, and generated the expected result.
   c. instructions showing how your code can be compiled/executed/tested by grader.

Please note that any **built-in deign patterns**, such as singleton (object) from Spring (boot) framework, **CANNOT** be used in your answer. You need to write your own code to implement each design pattern.

Please DO NOT hard code any input values, output values in your code.

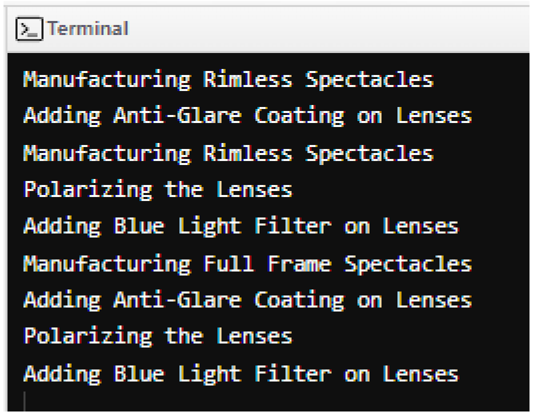
Steps for execution

1. Copy the code from ***"Advanced Software Paradigms_CSCI_6221_10_Sagar_Sheth_HW9.txt".***
2. Open the below link in the web browser.
   - https://www.tutorialspoint.com/online_python_compiler.php
3. Paste the Code in and Click on the Execute Button.



```
mport ABC, abstractmethod

class Spectacles(ABC):
    @abstractmethod
    def manufacture(self):
        pass

class RimlessSpectacles(Spectacles):
    def manufacture(self):
        print("Manufacturing Rimless Spectacles")

class FullFrameSpectacles(Spectacles):
    def manufacture(self):
        print("Manufacturing Full Frame Spectacles")

class SpectaclesFactory(ABC):
    @abstractmethod
    def create_spectacles(self):
        pass

class RimlessSpectaclesFactory(SpectaclesFactory):
    def create_spectacles(self):
        return RimlessSpectacles()

class FullFrameSpectaclesFactory(SpectaclesFactory):
    def create_spectacles(self):
        return FullFrameSpectacles()

class SpectaclesDecorator(Spectacles):
    def __init__(self, decorated_spectacles):
        self.decorated_spectacles = decorated_spectacles

    def manufacture(self):
        self.decorated_spectacles.manufacture()

class AntiGlareCoating(SpectaclesDecorator):
    def manufacture(self):
        super().manufacture()
        print("Adding Anti-Glare Coating on Lenses")

class PolarizedLenses(SpectaclesDecorator):
    def manufacture(self):
        super().manufacture()
        print("Polarizing the Lenses")

class BlueLightCoating(SpectaclesDecorator):
    def manufacture(self):
```

4. The below output should be displayed on the terminal.



```
Manufacturing Rimless Spectacles
Adding Anti-Glare Coating on Lenses
Manufacturing Rimless Spectacles
Polarizing the Lenses
Adding Blue Light Filter on Lenses
Manufacturing Full Frame Spectacles
Adding Anti-Glare Coating on Lenses
Polarizing the Lenses
Adding Blue Light Filter on Lenses
```