설계패턴 2주차 과제

컴퓨터공학과 202001818 송강규

Lab 1)

class Button:

    def click(self):

        pass

class DarkButton(Button):

    def click(self):

        print("dark click")

class LightButton(Button):

    def click(self):

        print("light click")

class RedButton(Button):

    def click(self):

        print("red click")

class BlueButton(Button):

    def click(self):

        print("blue click")

class ScrollBar:

    def scroll(self):

        pass

class DarkScrollBar(ScrollBar):

    def scroll(self):

        pass

class LightScrollBar(ScrollBar):

    def scroll(self):

        print("light ScrollBar")

class RedScrollBar(ScrollBar):

    def scroll(self):

        print("red ScrollBar")

class BlueScrollBar(ScrollBar):

    def scroll(self):

        print("blue ScrollBar")

class CheckBox:

    def check(self):

        pass

class DarkCheckBox(CheckBox):

    def check(self):

        print("dark check")

class LightCheckBox(CheckBox):

    def check(self):

        print("light check")

class RedCheckBox(CheckBox):

    def check(self):

        print("red check")

class BlueCheckBox(CheckBox):

    def check(self):

        print("blue check")

class Slider:

    def slide(self):

        pass

class DarkSlider(Slider):

    def slide(self):

        print("dark slide")

class LightSlider(Slider):

    def slide(self):

        print("light slide")

class RedSlider(Slider):

    def slide(self):

        print("red slide")

class BlueSlider(Slider):

    def slide(self):

        print("blue slide")

class TextBox:

    def text(self):

        pass

class DarkTextBox(TextBox):

    def text(self):

        print("dark text")

class LightTextBox(TextBox):

    def text(self):

        print("light text")

class RedTextBox(TextBox):

    def text(self):

        print("red text")

class BlueTextBox(TextBox):

    def text(self):

        print("blue text")

class UIFactory:

    def getButton(self):

        pass

    def getScroll(self):

        pass

    def getCheck(self):

        pass

    def getSlider(self):

        pass

    def getTextBox(self):

        pass

class DarkFactory(UIFactory):

    def getButton(self):

        return DarkButton()

    def getScroll(self):

        return DarkScrollBar()

    def getCheck(self):

        return DarkCheckBox()

    def getSlider(self):

        return DarkSlider()

    def getTextBox(self):

        return DarkTextBox()

class LightFactory(UIFactory):

    def getButton(self):

        return LightButton()

    def getScroll(self):

        return LightScrollBar()

    def getCheck(self):

        return LightCheckBox()

    def getSlider(self):

        return LightSlider()

    def getTextBox(self):

        return LightTextBox()

class RedFactory(UIFactory):

    def getButton(self):

        return RedButton()

    def getScroll(self):

        return RedScrollBar()

    def getCheck(self):

        return RedCheckBox()

    def getSlider(self):

        return RedSlider()

    def getTextBox(self):

        return RedTextBox()

class BlueFactory(UIFactory):

    def getButton(self):

        return BlueButton()

    def getScroll(self):

        return BlueScrollBar()

    def getCheck(self):

        return BlueCheckBox()

    def getSlider(self):

        return BlueSlider()

    def getTextBox(self):

        return BlueTextBox()

Lab2)

1. 주어진 코드에서 새로운 도형을 추가하려면 Drawing 클래스를 변경해야 한다는 점에서 Open-Closed 원칙을 위반한다.

2)

class Shape:

    def draw(self):

        pass

class Factory:

    def create\_shape(self) -> Shape:

        pass

class Circle(Shape):

    def draw(self):

        print("Drawing a circle")

class Square(Shape):

    def draw(self):

        print("Drawing a square")

class Rectangle(Shape):

    def draw(self):

        print("Drawing a rectangle")

class CircleFactory(Factory):

    def create\_shape(self) -> Shape:

        return Circle()

class SquareFactory(Factory):

    def create\_shape(self) -> Shape:

        return Square()

class RectangleFactory(Factory):

    def create\_shape(self) -> Shape:

        return Rectangle()

class Drawing:

    def draw\_shape(self, shape):

        shape.draw()

3)

class CircleFactory(Factory):

    def \_\_init\_\_(self):

        self.count = 0

    def create\_shape(self) -> Shape:

        self.count += 1

        return Circle()

class SquareFactory(Factory):

    def create\_shape(self) -> Shape:

        return Square()

    def rotate(self):

        print("Square Rotated")

4)

class Triangle(Shape):

    def draw(self):

        print("Drawing a triangle")

class TriangleFactory(Factory):

    def create\_shape(self):

        return Triangle()

5)

제공된 코드에 비해서 수정한 코드가 좀 더 길더라도 클래스 별 기능이 명확하고 코드를 다른 사람이 봐도 이해하기 쉬워 보인다. 또한 기존 코드가 위반했던 Open-Closed 원칙에 대해서도 새로 작성한 코드는 도형 추가 및 기능 추가에 대해서 자유롭다 라는 점도 기존 코드에 비해서 더 나은 코드라고 생각한다.