

설계패턴 실습 레포트

**과목명 설계패턴**

**담당교수 전병환 교수님**

**제출일 2023. 4. 18.**

**전공 컴퓨터.전자시스템 공학부**

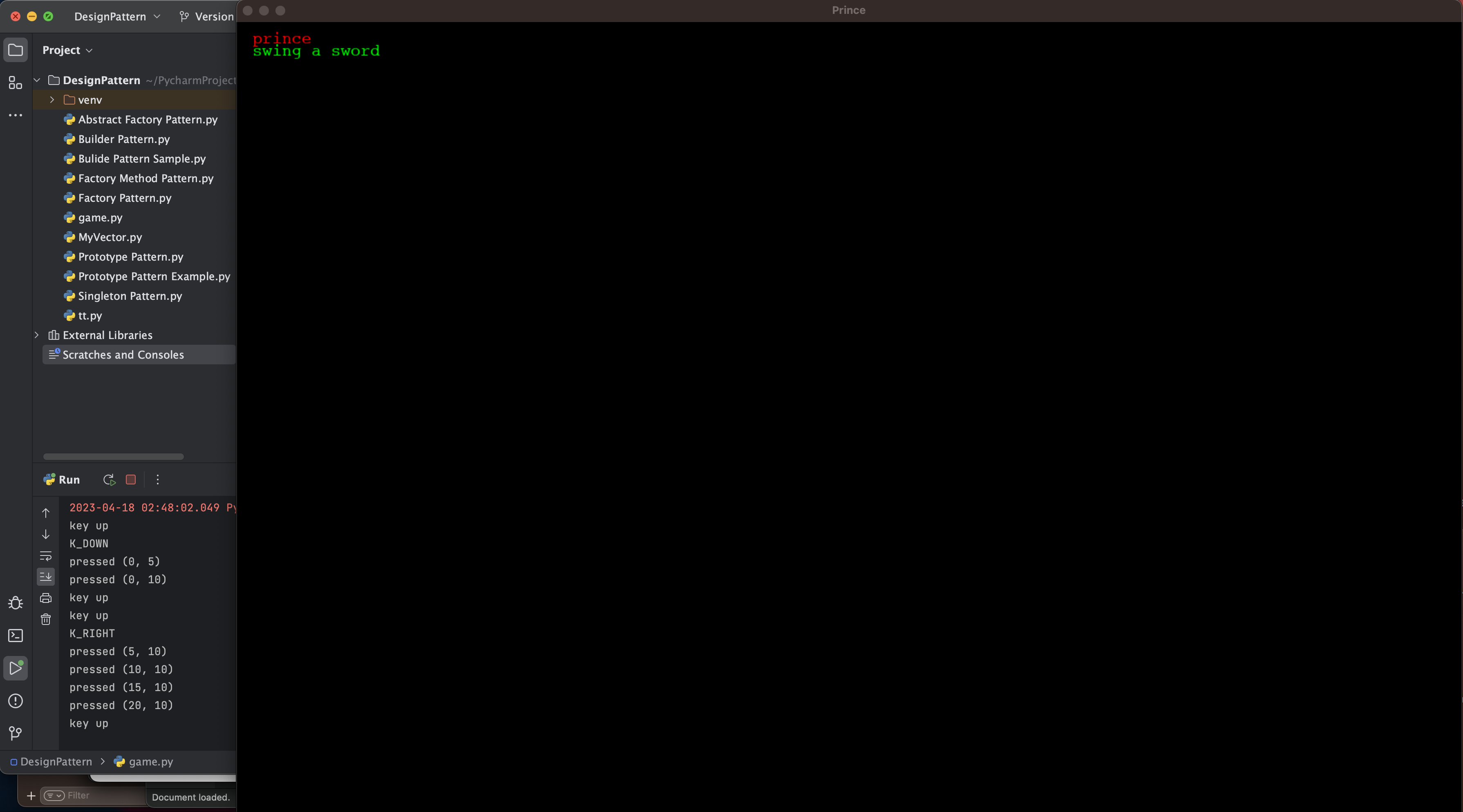
**학번 201703091**

**이름 전기범**

실습 1)

제공된 game.py 코드를 실행 및 분석하고 주석을 달고, 실행해보세요.

실행 결과)



소스코드)

import pygame  
import MyVector as mv #vector 클래스  
  
rgb = {  
 'BLACK':(0, 0, 0),  
 'WHITE':(255, 255, 255),  
 'BLUE':(0, 0, 255),  
 'GREEN':(0, 255, 0),  
 'RED':(255, 0, 0)  
} # 색상의 정의가 있는 딕셔너리 타임  
  
# Implementor  
class Actor:  
  
 def \_\_init\_\_(self, x, y):  
 self.pos = mv.MyVector(x, y)  
 self.name = ""  
 self.skill = ""  
  
 def setPos(self, x, y):  
 self.pos.x = x  
 self.pos.y = y  
  
 def move(self, delta):  
 self.pos = self.pos + delta  
   
 def setName(self, name):  
 self.name = name  
   
 def setSkill(self, skill):  
 pass  
  
# Concrete Implementor 1  
class Hero(Actor):  
  
 def setSkill(self, skill):  
 self.skill = skill  
  
# Concrete Implementor 2  
class Enermy(Actor):  
  
 def setSkill(self, skill):  
 self.skill = skill  
  
  
# Abstraction  
class GameFramework:  
  
 def \_\_init\_\_(self):  
 self.pygame = pygame  
 self.screen = 0  
  
 self.nY = 0 # 스크린의 크기를 담당   
 self.nX = 0  
  
 self.hero = 0 # 기능을 실제로 수행하는 위임자가 존재한다.  
  
 print("init")  
  
 def setDisplay(self, nX, nY):   
 self.nY = nY  
 self.nX = nX  
 self.screen = self.pygame.display.set\_mode([self.nX, self.nY]) # 스크린 설정  
 self.pygame.display.set\_caption("Prince") # 게임창의 이름  
  
  
 def setHero(self, hero:Actor):  
 self.hero = hero  
  
 def ready(self):  
 self.pygame.init() #pygame 초기화  
  
 def drawPolygon(self, color, points, thickness):  
 self.pygame.draw.polygon(self.screen, color, points, thickness)  
  
 def drawEdges(self):  
 p1 = mv.MyVector(0, 0)  
 p2 = mv.MyVector(0, 10)  
 p3 = mv.MyVector(10, 0)  
   
 self.drawPolygon(rgb["WHITE"], [p1.vec(), p2.vec(), p3.vec()], 1)  
  
 def printText(self, msg, color, pos):  
 font= self.pygame.font.SysFont("consolas",20)  
 textSurface = font.render(msg,True, color, None) #self.pygame.Color(color)  
 textRect = textSurface.get\_rect()  
 textRect.topleft= pos  
 self.screen.blit(textSurface, textRect)  
  
 #게임 실행  
 def launch(self):  
 pass  
  
  
# Refind Abstraction 1  
class WhiteGame(GameFramework):  
  
 def launch(self):  
 print("launch")  
 clock = self.pygame.time.Clock()  
   
 delta = mv.MyVector(0, 0)  
  
 keyFlag = None  
  
 done = False  
 while not done:   
 clock.tick(60) #set on 30 frames per second  
  
 for event in self.pygame.event.get():  
 if event.type == self.pygame.QUIT: #alt + f4  
 print("종료")  
 done = True  
  
 elif event.type == self.pygame.KEYDOWN: # 키를 눌렀을때  
 print("key down")  
 if event.key == self.pygame.K\_LEFT: # 어떤키가 눌렸는가?  
 print("K\_LEFT")  
 delta.x = -5  
 elif event.key == self.pygame.K\_RIGHT: # 방향키 상하좌우 입력에 대해 출력  
 print("K\_RIGHT")  
 delta.x = 5  
 elif event.key == self.pygame.K\_DOWN:  
 print("K\_DOWN")  
 delta.y = 5  
 elif event.key == self.pygame.K\_UP:  
 print("K\_UP")  
 delta.y = -5  
  
 keyFlag = True  
  
 elif event.type == self.pygame.KEYUP: # 키를 놓았을때 key up 출력  
 delta.setPos(0, 0)  
 print("key up")  
 keyFlag = False  
  
  
 if keyFlag == True:  
  
 self.hero.move(delta) #주인공의 위치가 업데이트가 됨  
  
 print("pressed", self.hero.pos.getState()) #in console  
 self.screen.fill(rgb["WHITE"]) #특성을 살린 부분  
 self.printText(self.hero.name, rgb["RED"], self.hero.pos.vec())   
 self.printText(self.hero.skill, rgb["GREEN"], (self.hero.pos + mv.MyVector(0, 15)).vec())  
  
 self.pygame.display.flip()  
  
 self.pygame.quit()  
  
  
# # Refind Abstraction 2  
class BlackGame(GameFramework):  
  
 def launch(self):  
 print("launch")  
 clock = self.pygame.time.Clock()  
   
 delta = mv.MyVector(0, 0)  
  
 keyFlag = None  
  
 done = False  
 while not done:   
 clock.tick(30) #set on 30 frames per second  
  
 for event in self.pygame.event.get():  
 if event.type == self.pygame.QUIT:  
 done = True  
  
 elif event.type == self.pygame.KEYDOWN:  
 print("key up")  
 if event.key == self.pygame.K\_LEFT:  
 print("K\_LEFT")  
 delta.x = -5  
 elif event.key == self.pygame.K\_RIGHT:  
 print("K\_RIGHT")  
 delta.x = 5  
 elif event.key == self.pygame.K\_DOWN:  
 print("K\_DOWN")  
 delta.y = 5  
 elif event.key == self.pygame.K\_UP:  
 print("K\_UP")  
 delta.y = -5  
  
 keyFlag = True  
  
 elif event.type == self.pygame.KEYUP:  
 delta.setPos(0, 0)  
 print("key up")  
 keyFlag = False  
  
  
 if keyFlag == True:  
  
 self.hero.move(delta)  
  
 print("pressed", self.hero.pos.getState()) #in console  
 self.screen.fill(rgb["BLACK"]) #특성화된 부분  
 self.printText(self.hero.name, rgb["RED"], self.hero.pos.vec())   
 self.printText(self.hero.skill, rgb["GREEN"], (self.hero.pos + mv.MyVector(0, 15)).vec())  
  
 self.pygame.display.flip()  
  
 self.pygame.quit()  
  
  
  
game = BlackGame() # 게임을 검은 배경의 블랙 게임으로 설정  
game.ready()  
game.setDisplay(1500, 1000) # 게임창의 사이즈 설정 예제는 1500\*1000  
game.drawEdges() # MyVector에 설정된 값으로 도형 그리기  
  
hero = Hero(0, 0) # 히어로 세팅  
hero.setName("prince")  
hero.setSkill("swing a sword")  
  
monster = Enermy(50, 50) # 몬스터 세팅  
monster.setName("weak moster")  
monster.setSkill("hit the body")  
  
game.setHero(hero) # 히어로 생성  
game.setHero(monster) # 몬스터 생성  
  
game.launch() # 게임 실행

실습 2)

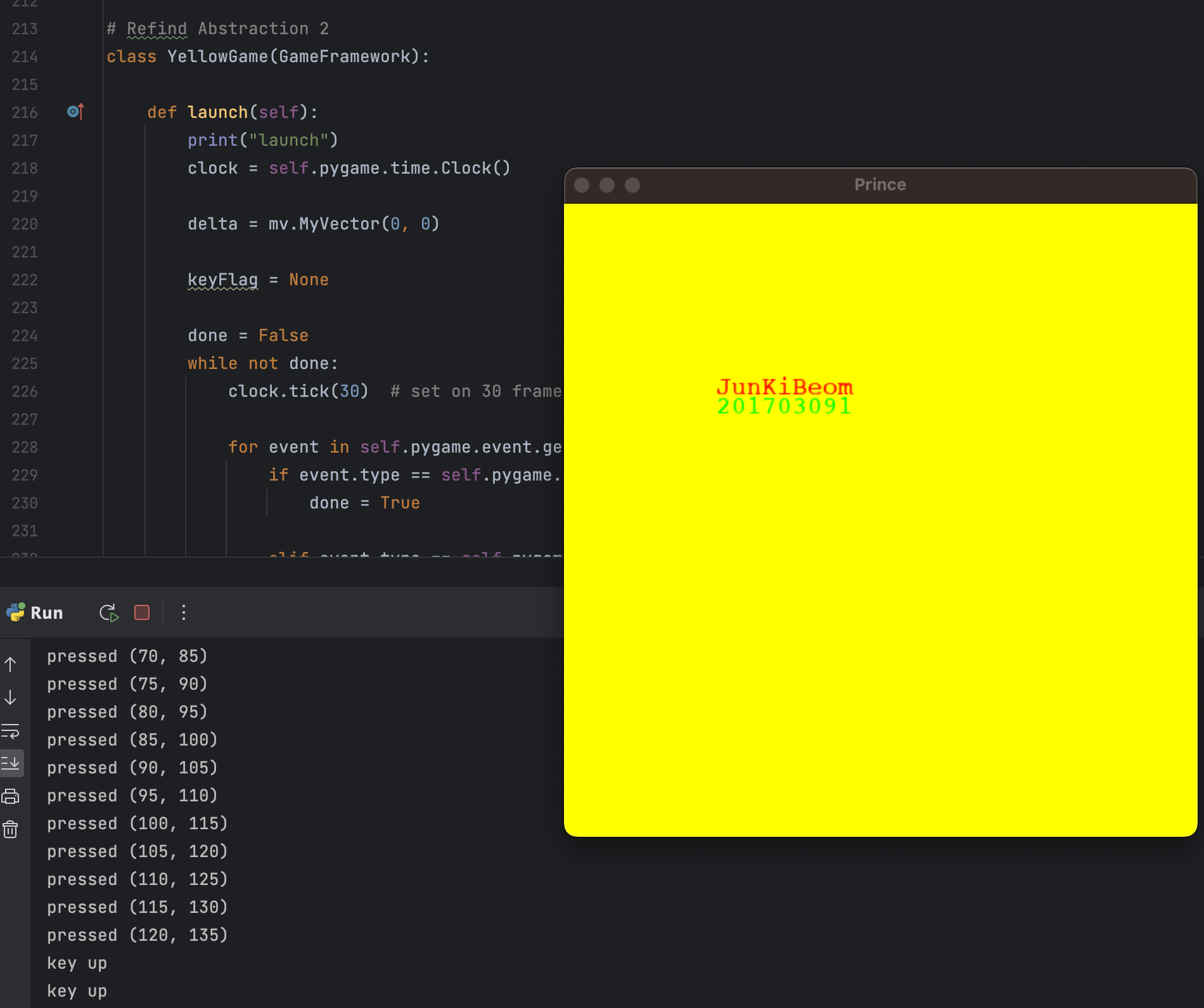
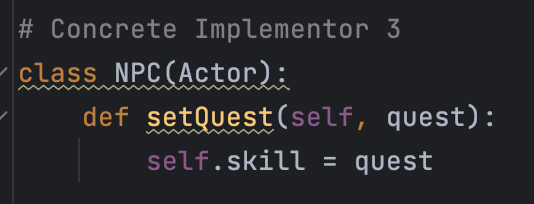
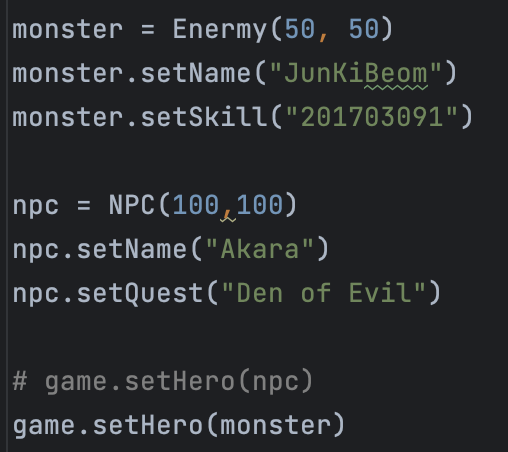
Bridge Pattern 강의에서 제공된 pygame라이브러리를 활용한 소스코드를 기반으로 아래의 기능을 추가하시오.

 - NPC (non player character) 기능을하는 Concrete Implementor를 추가해보세요. NPC는 skill이 없는 대신 quest를 가지고 있을 수 있습니다. Quest는 문자열로 표현되며 setQuest 함수 NPC가 제공하는 quest를 정의해줄 수 있습니다.

 - Yellow 버전의 게임을 위한 Refined Abstraction class를 추가하시오.

 - Hero 또는 Enermy의 name에 본인의 이름을 설정해주고, skill 속성에 학번을 설정해보세요.

실행 결과)



해결방안)

NPC (non player character) 기능을하는 Concrete Implementor를 추가하기 위해 Hero와 Enermy 하단에 Actor를 상속받는 class를 작성하였습니다.

NPC는 setQuest함수를 가지고 있고 skill대신 quset를 가지고 있을 수 있도록 하였습니다.

Yellow 버전의 게임을 위한 Refined Abstraction class를 추가하기위해서 rgb딕셔너리에 'YELLOW':(255,255,0) 값을 추가해줬습니다. 이후 기존의 BlackGame 클래스를 복사하여 클래스명을 YellowGame으로 변경, self.screen.fill()에 rgb["YELLOW"]를 넣어 앞에서 추가한 딕셔너리의 value값이 넘어가도록 하였습니다.

해당 작업들을 모두 마친 후 YellowGame으로 설정, Enermy에 제 이름과 학번을 넣어 출력을 진행했습니다.

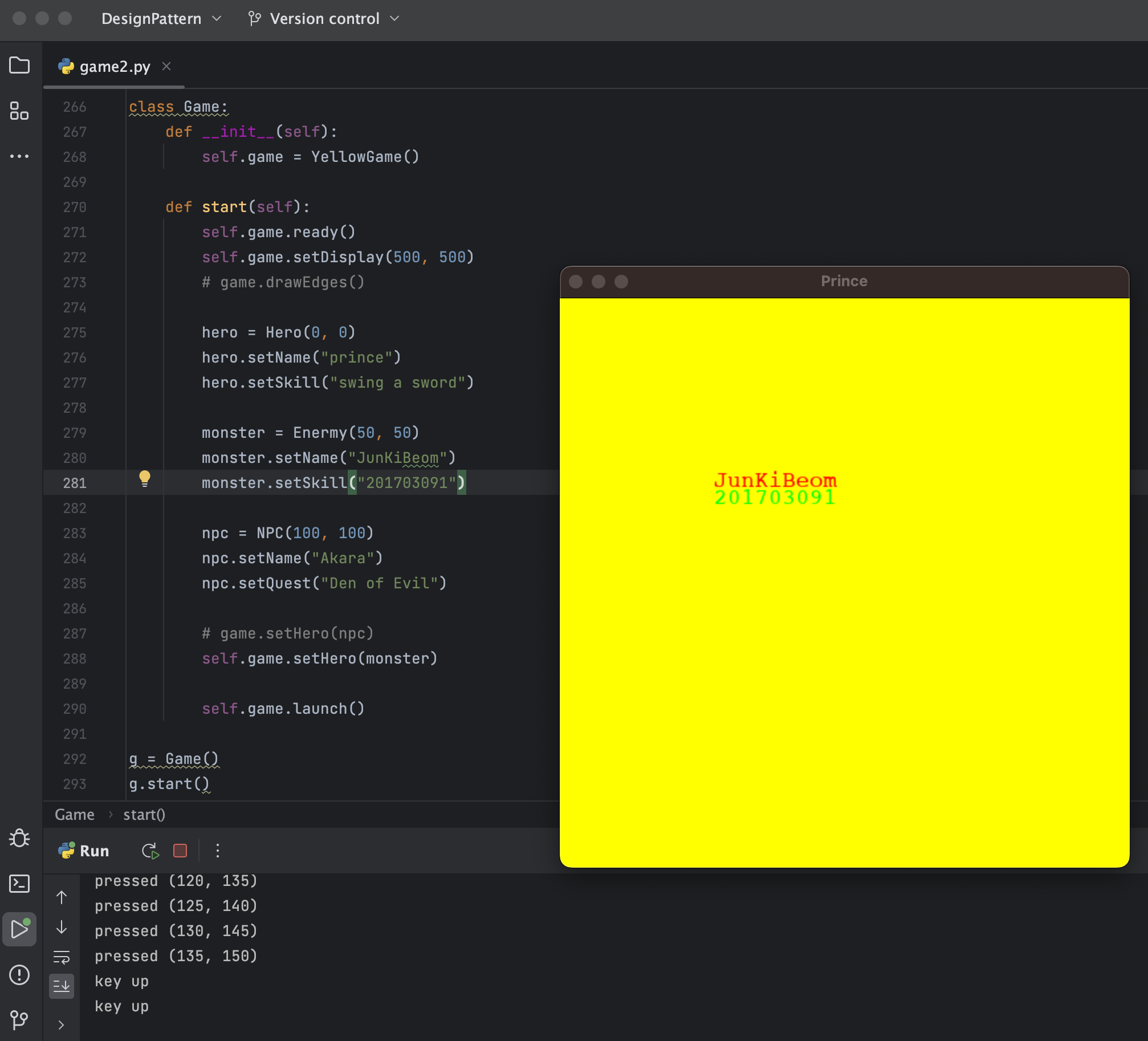
소스코드)

import pygame  
import MyVector as mv #vector 클래스  
  
rgb = {  
 'BLACK':(0, 0, 0),  
 'WHITE':(255, 255, 255),  
 'BLUE':(0, 0, 255),  
 'GREEN':(0, 255, 0),  
 'RED':(255, 0, 0),  
 'YELLOW':(255,255,0) # YELLOW 게임을 위해 추가  
} # 딕셔너리 타임  
  
# Implementor  
class Actor:  
  
 def \_\_init\_\_(self, x, y):  
 self.pos = mv.MyVector(x, y)  
 self.name = ""  
 self.skill = ""  
  
 def setPos(self, x, y):  
 self.pos.x = x  
 self.pos.y = y  
  
 def move(self, delta):  
 self.pos = self.pos + delta  
   
 def setName(self, name):  
 self.name = name  
   
 def setSkill(self, skill):  
 pass  
  
# Concrete Implementor 1  
class Hero(Actor):  
  
 def setSkill(self, skill):  
 self.skill = skill  
  
# Concrete Implementor 2  
class Enermy(Actor):  
  
 def setSkill(self, skill):  
 self.skill = skill  
  
# Concrete Implementor 3  
class NPC(Actor):  
 def setQuest(self, quest):  
 self.skill = quest  
  
# Abstraction  
class GameFramework:  
  
 def \_\_init\_\_(self):  
 self.pygame = pygame  
 self.screen = 0  
  
 self.nY = 0 # 스크린의 크기를 담당   
 self.nX = 0  
  
 self.hero = 0 #기능을 실제로 수행하는 위임자가 존재한다.  
  
 print("init")  
  
 def setDisplay(self, nX, nY):   
 self.nY = nY  
 self.nX = nX  
 self.screen = self.pygame.display.set\_mode([self.nX, self.nY])  
 self.pygame.display.set\_caption("Prince") #게임창의 이름  
  
  
 def setHero(self, hero:Actor):  
 self.hero = hero  
  
 def ready(self):  
 self.pygame.init() #pygame 초기화  
  
 def drawPolygon(self, color, points, thickness):  
 self.pygame.draw.polygon(self.screen, color, points, thickness)  
  
 def drawEdges(self):  
 p1 = mv.MyVector(0, 0)  
 p2 = mv.MyVector(0, 10)  
 p3 = mv.MyVector(10, 0)  
   
 self.drawPolygon(rgb["WHITE"], [p1.vec(), p2.vec(), p3.vec()], 1)  
  
 def printText(self, msg, color, pos):  
 font= self.pygame.font.SysFont("consolas",20)  
 textSurface = font.render(msg,True, color, None) #self.pygame.Color(color)  
 textRect = textSurface.get\_rect()  
 textRect.topleft= pos  
 self.screen.blit(textSurface, textRect)  
  
 #게임 실행  
 def launch(self):  
 pass  
  
  
# Refind Abstraction 1  
class WhiteGame(GameFramework):  
  
 def launch(self):  
 print("launch")  
 clock = self.pygame.time.Clock()  
   
 delta = mv.MyVector(0, 0)  
  
 keyFlag = None  
  
 done = False  
 while not done:   
 clock.tick(60) #set on 30 frames per second  
  
 for event in self.pygame.event.get():  
 if event.type == self.pygame.QUIT: #alt + f4  
 print("종료")  
 done = True  
  
 elif event.type == self.pygame.KEYDOWN: #키를 눌렀을때  
 print("key down")  
 if event.key == self.pygame.K\_LEFT: #어떤키가 눌렸는가?  
 print("K\_LEFT")  
 delta.x = -5  
 elif event.key == self.pygame.K\_RIGHT:  
 print("K\_RIGHT")  
 delta.x = 5  
 elif event.key == self.pygame.K\_DOWN:  
 print("K\_DOWN")  
 delta.y = 5  
 elif event.key == self.pygame.K\_UP:  
 print("K\_UP")  
 delta.y = -5  
  
 keyFlag = True  
  
 elif event.type == self.pygame.KEYUP:  
 delta.setPos(0, 0)  
 print("key up")  
 keyFlag = False  
  
  
 if keyFlag == True:  
  
 self.hero.move(delta) #주인공의 위치가 업데이트가 됨  
  
 print("pressed", self.hero.pos.getState()) #in console  
 self.screen.fill(rgb["WHITE"]) #특성을 살린 부분  
 self.printText(self.hero.name, rgb["RED"], self.hero.pos.vec())   
 self.printText(self.hero.skill, rgb["GREEN"], (self.hero.pos + mv.MyVector(0, 15)).vec())  
  
 self.pygame.display.flip()  
  
 self.pygame.quit()  
  
  
# Refind Abstraction 2  
class BlackGame(GameFramework):  
  
 def launch(self):  
 print("launch")  
 clock = self.pygame.time.Clock()  
  
 delta = mv.MyVector(0, 0)  
  
 keyFlag = None  
  
 done = False  
 while not done:  
 clock.tick(30) #set on 30 frames per second  
  
 for event in self.pygame.event.get():  
 if event.type == self.pygame.QUIT:  
 done = True  
  
 elif event.type == self.pygame.KEYDOWN:  
 print("key up")  
 if event.key == self.pygame.K\_LEFT:  
 print("K\_LEFT")  
 delta.x = -5  
 elif event.key == self.pygame.K\_RIGHT:  
 print("K\_RIGHT")  
 delta.x = 5  
 elif event.key == self.pygame.K\_DOWN:  
 print("K\_DOWN")  
 delta.y = 5  
 elif event.key == self.pygame.K\_UP:  
 print("K\_UP")  
 delta.y = -5  
  
 keyFlag = True  
  
 elif event.type == self.pygame.KEYUP:  
 delta.setPos(0, 0)  
 print("key up")  
 keyFlag = False  
  
  
 if keyFlag == True:  
  
 self.hero.move(delta)  
  
 print("pressed", self.hero.pos.getState()) #in console  
 self.screen.fill(rgb["BLACK"]) #특성화된 부분  
 self.printText(self.hero.name, rgb["RED"], self.hero.pos.vec())  
 self.printText(self.hero.skill, rgb["GREEN"], (self.hero.pos + mv.MyVector(0, 15)).vec())  
  
 self.pygame.display.flip()  
  
 self.pygame.quit()  
  
  
# Refind Abstraction 2  
class YellowGame(GameFramework):  
  
 def launch(self):  
 print("launch")  
 clock = self.pygame.time.Clock()  
  
 delta = mv.MyVector(0, 0)  
  
 keyFlag = None  
  
 done = False  
 while not done:  
 clock.tick(30) # set on 30 frames per second  
  
 for event in self.pygame.event.get():  
 if event.type == self.pygame.QUIT:  
 done = True  
  
 elif event.type == self.pygame.KEYDOWN:  
 print("key up")  
 if event.key == self.pygame.K\_LEFT:  
 print("K\_LEFT")  
 delta.x = -5  
 elif event.key == self.pygame.K\_RIGHT:  
 print("K\_RIGHT")  
 delta.x = 5  
 elif event.key == self.pygame.K\_DOWN:  
 print("K\_DOWN")  
 delta.y = 5  
 elif event.key == self.pygame.K\_UP:  
 print("K\_UP")  
 delta.y = -5  
  
 keyFlag = True  
  
 elif event.type == self.pygame.KEYUP:  
 delta.setPos(0, 0)  
 print("key up")  
 keyFlag = False  
  
 if keyFlag == True:  
 self.hero.move(delta)  
  
 print("pressed", self.hero.pos.getState()) # in console  
 self.screen.fill(rgb["YELLOW"]) # 특성화된 부분  
 self.printText(self.hero.name, rgb["RED"], self.hero.pos.vec())  
 self.printText(self.hero.skill, rgb["GREEN"], (self.hero.pos + mv.MyVector(0, 15)).vec())  
  
 self.pygame.display.flip()  
  
 self.pygame.quit()  
  
# game = BlackGame()  
game = YellowGame()  
game.ready()  
game.setDisplay(500, 500)  
# game.drawEdges()  
  
hero = Hero(0, 0)  
hero.setName("prince")  
hero.setSkill("swing a sword")  
  
monster = Enermy(50, 50)  
monster.setName("JunKiBeom")  
monster.setSkill("201703091")  
  
npc = NPC(100,100)  
npc.setName("Akara")  
npc.setQuest("Den of Evil")  
  
# game.setHero(npc)  
game.setHero(monster)  
  
game.launch()

실습 3)

게임의 초기화부터 실행까지 한번에 실행해주는 Façade Pattern을 추가하시오. (클래스명 등 이름은 자유롭게 설정가능)

실행 결과)



해결방안)

Façade Pattern을 적용하기위한 class Game을 작성 후, self.game = YellowGame을 Has-A관계로 가지고 있게 만들었습니다.

이후 start메서드를 작성하여 기존 main에 있던 코드들을 모두 이전하여, 사용자가 별도의 코드를 작성 할 필요 없이 클래스의 인스턴스를 생성하고, 해당 인스턴스의 메서드인 start를 호출하여 프로그램이 실행될 수 있도록 하였습니다.

소스코드)

import pygame  
import MyVector as mv #vector 클래스  
  
rgb = {  
 'BLACK':(0, 0, 0),  
 'WHITE':(255, 255, 255),  
 'BLUE':(0, 0, 255),  
 'GREEN':(0, 255, 0),  
 'RED':(255, 0, 0),  
 'YELLOW':(255,255,0) # YELLOW 게임을 위해 추가  
} # 딕셔너리 타임  
  
# Implementor  
class Actor:  
  
 def \_\_init\_\_(self, x, y):  
 self.pos = mv.MyVector(x, y)  
 self.name = ""  
 self.skill = ""  
  
 def setPos(self, x, y):  
 self.pos.x = x  
 self.pos.y = y  
  
 def move(self, delta):  
 self.pos = self.pos + delta  
   
 def setName(self, name):  
 self.name = name  
   
 def setSkill(self, skill):  
 pass  
  
# Concrete Implementor 1  
class Hero(Actor):  
  
 def setSkill(self, skill):  
 self.skill = skill  
  
# Concrete Implementor 2  
class Enermy(Actor):  
  
 def setSkill(self, skill):  
 self.skill = skill  
  
# Concrete Implementor 3  
class NPC(Actor):  
 def setQuest(self, quest):  
 self.skill = quest  
  
# Abstraction  
class GameFramework:  
  
 def \_\_init\_\_(self):  
 self.pygame = pygame  
 self.screen = 0  
  
 self.nY = 0 # 스크린의 크기를 담당   
 self.nX = 0  
  
 self.hero = 0 #기능을 실제로 수행하는 위임자가 존재한다.  
  
 print("init")  
  
 def setDisplay(self, nX, nY):   
 self.nY = nY  
 self.nX = nX  
 self.screen = self.pygame.display.set\_mode([self.nX, self.nY])  
 self.pygame.display.set\_caption("Prince") #게임창의 이름  
  
  
 def setHero(self, hero:Actor):  
 self.hero = hero  
  
 def ready(self):  
 self.pygame.init() #pygame 초기화  
  
 def drawPolygon(self, color, points, thickness):  
 self.pygame.draw.polygon(self.screen, color, points, thickness)  
  
 def drawEdges(self):  
 p1 = mv.MyVector(0, 0)  
 p2 = mv.MyVector(0, 10)  
 p3 = mv.MyVector(10, 0)  
   
 self.drawPolygon(rgb["WHITE"], [p1.vec(), p2.vec(), p3.vec()], 1)  
  
 def printText(self, msg, color, pos):  
 font= self.pygame.font.SysFont("consolas",20)  
 textSurface = font.render(msg,True, color, None) #self.pygame.Color(color)  
 textRect = textSurface.get\_rect()  
 textRect.topleft= pos  
 self.screen.blit(textSurface, textRect)  
  
 #게임 실행  
 def launch(self):  
 pass  
  
  
# Refind Abstraction 1  
class WhiteGame(GameFramework):  
  
 def launch(self):  
 print("launch")  
 clock = self.pygame.time.Clock()  
   
 delta = mv.MyVector(0, 0)  
  
 keyFlag = None  
  
 done = False  
 while not done:   
 clock.tick(60) #set on 30 frames per second  
  
 for event in self.pygame.event.get():  
 if event.type == self.pygame.QUIT: #alt + f4  
 print("종료")  
 done = True  
  
 elif event.type == self.pygame.KEYDOWN: #키를 눌렀을때  
 print("key down")  
 if event.key == self.pygame.K\_LEFT: #어떤키가 눌렸는가?  
 print("K\_LEFT")  
 delta.x = -5  
 elif event.key == self.pygame.K\_RIGHT:  
 print("K\_RIGHT")  
 delta.x = 5  
 elif event.key == self.pygame.K\_DOWN:  
 print("K\_DOWN")  
 delta.y = 5  
 elif event.key == self.pygame.K\_UP:  
 print("K\_UP")  
 delta.y = -5  
  
 keyFlag = True  
  
 elif event.type == self.pygame.KEYUP:  
 delta.setPos(0, 0)  
 print("key up")  
 keyFlag = False  
  
  
 if keyFlag == True:  
  
 self.hero.move(delta) #주인공의 위치가 업데이트가 됨  
  
 print("pressed", self.hero.pos.getState()) #in console  
 self.screen.fill(rgb["WHITE"]) #특성을 살린 부분  
 self.printText(self.hero.name, rgb["RED"], self.hero.pos.vec())   
 self.printText(self.hero.skill, rgb["GREEN"], (self.hero.pos + mv.MyVector(0, 15)).vec())  
  
 self.pygame.display.flip()  
  
 self.pygame.quit()  
  
  
# Refind Abstraction 2  
class BlackGame(GameFramework):  
  
 def launch(self):  
 print("launch")  
 clock = self.pygame.time.Clock()  
  
 delta = mv.MyVector(0, 0)  
  
 keyFlag = None  
  
 done = False  
 while not done:  
 clock.tick(30) #set on 30 frames per second  
  
 for event in self.pygame.event.get():  
 if event.type == self.pygame.QUIT:  
 done = True  
  
 elif event.type == self.pygame.KEYDOWN:  
 print("key up")  
 if event.key == self.pygame.K\_LEFT:  
 print("K\_LEFT")  
 delta.x = -5  
 elif event.key == self.pygame.K\_RIGHT:  
 print("K\_RIGHT")  
 delta.x = 5  
 elif event.key == self.pygame.K\_DOWN:  
 print("K\_DOWN")  
 delta.y = 5  
 elif event.key == self.pygame.K\_UP:  
 print("K\_UP")  
 delta.y = -5  
  
 keyFlag = True  
  
 elif event.type == self.pygame.KEYUP:  
 delta.setPos(0, 0)  
 print("key up")  
 keyFlag = False  
  
  
 if keyFlag == True:  
  
 self.hero.move(delta)  
  
 print("pressed", self.hero.pos.getState()) #in console  
 self.screen.fill(rgb["BLACK"]) #특성화된 부분  
 self.printText(self.hero.name, rgb["RED"], self.hero.pos.vec())  
 self.printText(self.hero.skill, rgb["GREEN"], (self.hero.pos + mv.MyVector(0, 15)).vec())  
  
 self.pygame.display.flip()  
  
 self.pygame.quit()  
  
  
# Refind Abstraction 2  
class YellowGame(GameFramework):  
  
 def launch(self):  
 print("launch")  
 clock = self.pygame.time.Clock()  
  
 delta = mv.MyVector(0, 0)  
  
 keyFlag = None  
  
 done = False  
 while not done:  
 clock.tick(30) # set on 30 frames per second  
  
 for event in self.pygame.event.get():  
 if event.type == self.pygame.QUIT:  
 done = True  
  
 elif event.type == self.pygame.KEYDOWN:  
 print("key up")  
 if event.key == self.pygame.K\_LEFT:  
 print("K\_LEFT")  
 delta.x = -5  
 elif event.key == self.pygame.K\_RIGHT:  
 print("K\_RIGHT")  
 delta.x = 5  
 elif event.key == self.pygame.K\_DOWN:  
 print("K\_DOWN")  
 delta.y = 5  
 elif event.key == self.pygame.K\_UP:  
 print("K\_UP")  
 delta.y = -5  
  
 keyFlag = True  
  
 elif event.type == self.pygame.KEYUP:  
 delta.setPos(0, 0)  
 print("key up")  
 keyFlag = False  
  
 if keyFlag == True:  
 self.hero.move(delta)  
  
 print("pressed", self.hero.pos.getState()) # in console  
 self.screen.fill(rgb["YELLOW"]) # 특성화된 부분  
 self.printText(self.hero.name, rgb["RED"], self.hero.pos.vec())  
 self.printText(self.hero.skill, rgb["GREEN"], (self.hero.pos + mv.MyVector(0, 15)).vec())  
  
 self.pygame.display.flip()  
  
 self.pygame.quit()  
  
class Game:  
 def \_\_init\_\_(self):  
 self.game = YellowGame()  
  
 def start(self):  
 self.game.ready()  
 self.game.setDisplay(500, 500)  
 # game.drawEdges()  
  
 hero = Hero(0, 0)  
 hero.setName("prince")  
 hero.setSkill("swing a sword")  
  
 monster = Enermy(50, 50)  
 monster.setName("JunKiBeom")  
 monster.setSkill("201703091")  
  
 npc = NPC(100, 100)  
 npc.setName("Akara")  
 npc.setQuest("Den of Evil")  
  
 # game.setHero(npc)  
 self.game.setHero(monster)  
  
 self.game.launch()  
  
g = Game()  
g.start()

느낀 점)

Bridge Pattern과 Façade Pattern을 pygame을 이용해 실습을 하며 코드를 작성하니 어떤 식으로 코드를 작성해야 하는가, 효율성이 좋은가에 대해서 생각해보게 되었습니다.

Refind Abstraction으로 만드는 색상별로 다른 게임을 만드는 메서드의 코드 전반적인 부분이 중복이 되어, 해당 코드를 하나의 메서드로 통합하고 인자 값으로 입력되는 색상에 따라 게임의 Background를 바꿀 수 있게 변형이 가능한걸 확인하게 되어 해당 코드를 작성하였으나 과제의 취지와는 부합하지 않아 사용을 하지 못했습니다.

그러한 방식을 사용하지 않고 색상마다 새로운 class를 만들어서 진행한 까닭이 궁금해지기도 했고, 제가 생각한 방식을 이용하여 코드를 작성하면 패턴에서 벗어나는 것인지, 아니면 SOLID원칙에서 벗어나는 것인지 궁금증이 생겼으며 해당 부분에 대한 답을 내리기 위해서 더 공부를 할 필요성이 있다고 느꼈습니다.

# Modified Refind Abstraction  
class Game(GameFramework):  
 def \_\_init\_\_(self, color):  
 super().\_\_init\_\_()  
 self.color = color  
  
 def launch(self):  
 print("launch")  
 clock = self.pygame.time.Clock()  
   
 delta = mv.MyVector(0, 0)  
  
 keyFlag = None  
  
 done = False  
 while not done:   
 clock.tick(60) #set on 30 frames per second  
  
 for event in self.pygame.event.get():  
 if event.type == self.pygame.QUIT: #alt + f4  
 print("종료")  
 done = True  
  
 elif event.type == self.pygame.KEYDOWN: #키를 눌렀을때  
 print("key down")  
 if event.key == self.pygame.K\_LEFT: #어떤키가 눌렸는가?  
 print("K\_LEFT")  
 delta.x = -5  
 elif event.key == self.pygame.K\_RIGHT:  
 print("K\_RIGHT")  
 delta.x = 5  
 elif event.key == self.pygame.K\_DOWN:  
 print("K\_DOWN")  
 delta.y = 5  
 elif event.key == self.pygame.K\_UP:  
 print("K\_UP")  
 delta.y = -5  
  
 keyFlag = True  
  
 elif event.type == self.pygame.KEYUP:  
 delta.setPos(0, 0)  
 print("key up")  
 keyFlag = False  
  
  
 if keyFlag == True:  
  
 self.hero.move(delta) #주인공의 위치가 업데이트가 됨  
  
 print("pressed", self.hero.pos.getState()) #in console  
 self.screen.fill(rgb[self.color.upper()]) #특성을 살린 부분  
 # 소문자로 입력이 되는 경우 Dict의 key는 대문자라 모두 문자열이 대문자로 치환하게 upper()를 사용  
 self.printText(self.hero.name, rgb["RED"], self.hero.pos.vec())   
 self.printText(self.hero.skill, rgb["GREEN"], (self.hero.pos + mv.MyVector(0, 15)).vec())  
  
 self.pygame.display.flip()  
  
 self.pygame.quit()