الجامعة السعودية الإلكترونية
SAUDI ELECTRONIC UNIVERSITY
2011-1432
كلية الحوسبة والمعلوماتية |
College of Computing and Informatics

Second Semester – 2021/2022

Course Code	DS650
Course Name	Predictive Analytics
Assignment type	Critical Thinking
Module	03
Total Points	105 Points

Student ID	G200007615
Student Name	Abdulaziz Alqumayzi
CRN	21601

Solutions:

Critical Thinking Assignment 1

Design and Create a Python Class Library

Introduction

In this activity, we will design and build a Python class library, which will have client, child, and parent class files. We'll walk through the procedures of adding three methods to the child and parent classes, as well as demonstrating the OOP characteristics inheritance, encapsulation, and polymorphism in the client class.

Design and create a Python class library including the client, child, and parent class files

Three python files have been created, Parent, Child and Client. Parent file has three classes, Stadium, car and aircraft. Child file has sanSiro class. Lastly Client class has objects to test inheritance, encapsulation, and polymorphism features.

Add three methods to the child and parent classes

In Parent file, three methods have been created stadiumType, pitchSize and pitchType as we can see in the following code.

```
class Stadium:
    def __init__(self):
        # Encapsulation restricts __sportType variable to prevent data from
direct modification
        # by using underscore as prefix __sportType
        self.__sportType= 'Foorball'

    # creating instance method
    def stadiumType(self):
        print("Sport is: {}".format(self.__sportType))

    # creating class method
    @classmethod
    def pitchSize(cls):
        print("105x68",cls)

    # creating static method
    @staticmethod
    def pitchType():
```

```
        print('Grass')

# creating two classes for polymorphism example
class car:
    def fly(self):
        print('Cars can not fly')

class aircraft:
    def fly(self):
        print('Aircrafts can fly')
```

stadiumType method prints the variable __sportType, which is restricted form modification because the two underscore before sportType. Second method is pitchSize that is a class method and prints size of the pitch. Final method is pitch type which is mostly grass for any football pitch. Also, two classes have been created for the polymorphism example, car and aircraft classes. all classes have the same method “fly” which is print if the car and aircraft can fly or not. In the Child file, Parent file has been imported and one class (sanSiro) has been created as we can see in the following code.

```
from Parent import Stadium

# by adding Stadium class in parentheses of class sanSiro we are applying
inheritance
class sanSiro(Stadium):
    def __init__(self):
        print("San Siro Stadium")

    # creating instance method
    def team(self):
        print("AC Milan and Inter Milan")

    # creating class method
    @classmethod
    def capacity(cls):
        print(80018,cls)

    # creating static method
    @staticmethod
    def opened():
        print('September 19, 1926')
```

San Siro is an Italian football stadium, and the class took the name of that stadium. It has three methods, team, capacity and opened. sanSiro class inherits Stadium class methods from Parent file because we add Stadium class in parentheses of sanSiro class.

Write the code in the client to show the following OOP features: Inheritance, Encapsulation, and Polymorphism

Client file has objects to test inheritance and encapsulation features and a class canFly to test polymorphism feature as shown in the following code.

```
from Child import sanSiro
from Parent import Stadium,car,aircraft

# Inheritance example
obj1 = sanSiro()
# accessing pitchType method that does not exist in sanSiro class
obj1.pitchType()

# Encapsulation example
obj2 = Stadium()
# putting Basketball value in __sportType
obj2.__sportType = "Basketball"
obj2.stadiumType()

# Polymorphism example
def canFly(test):
    test.fly()

carTest = car()
aircraftTest = aircraft()

# testing polymorphism
canFly(carTest)
canFly(aircraftTest)
```

Let's start by testing inheritance from Child file for Parent file. Object “obj1” has been created and pitchType method tested to check if inheritance has been applied. The result in figure 1 below:

Figure 1

Inheritance output

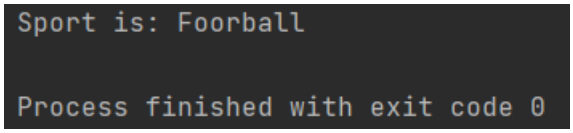
```
San Siro Stadium
Grass

Process finished with exit code 0
```

We can see Grass printed which means Stadium class has been inherited successfully. Note that each test will apply separately as we saw from the result above. Next, we test encapsulation

by changing __sportType variable to Basketball instead of Football. The result in the following figure 2:

Figure 2

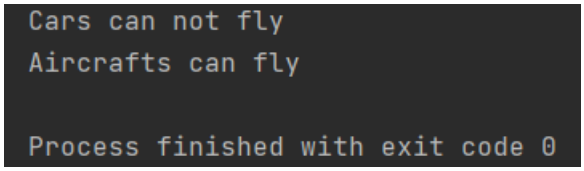
Encapsulation output



As we know it shouldn't be changed because it is restricted variable. And the result above is confirm the restriction on the variable. Lastly is the polymorphism test. The result of the test is shown below (Figure 3):

Figure 3

Polymorphism output



We can see from the result above it runs successfully.

References

Kalb, I. (2022). *Object-oriented python: Master oop by building games and guis*. No Starch Press.

Lott, S. F. (2014). *Mastering object-oriented python*. Packt Publishing.