Object-Oriented Analysis and Design

HW2

古景睿40971216H 戴子棋40971229H 許力文40971130H 李承曄41071133H

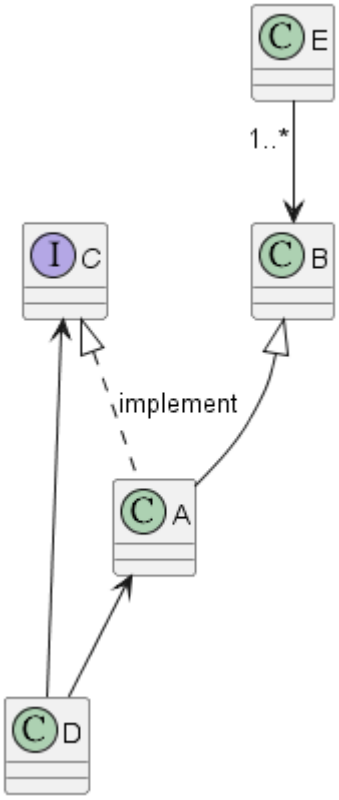
Github link(推薦看 github readme，比 word 清楚)：

https://github.com/wannaflyhigh/Object-Oriented-Programming/tree/b95386252db1f91d3b0aedc78eae2beca27e7efd/hw2

1.

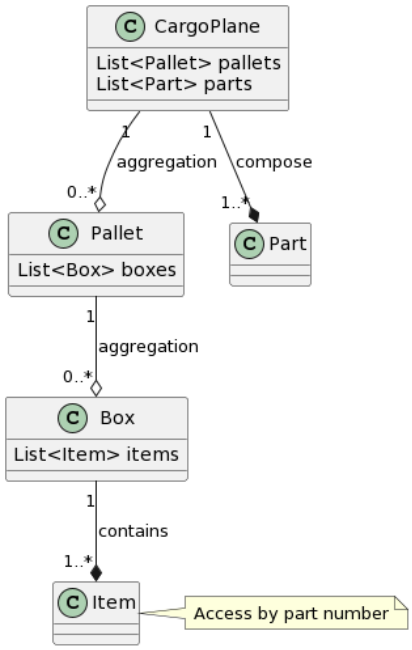
(a)

A is a subclass of B. A implements an interface C which is used by D to access A. B is associated with one or more Es.



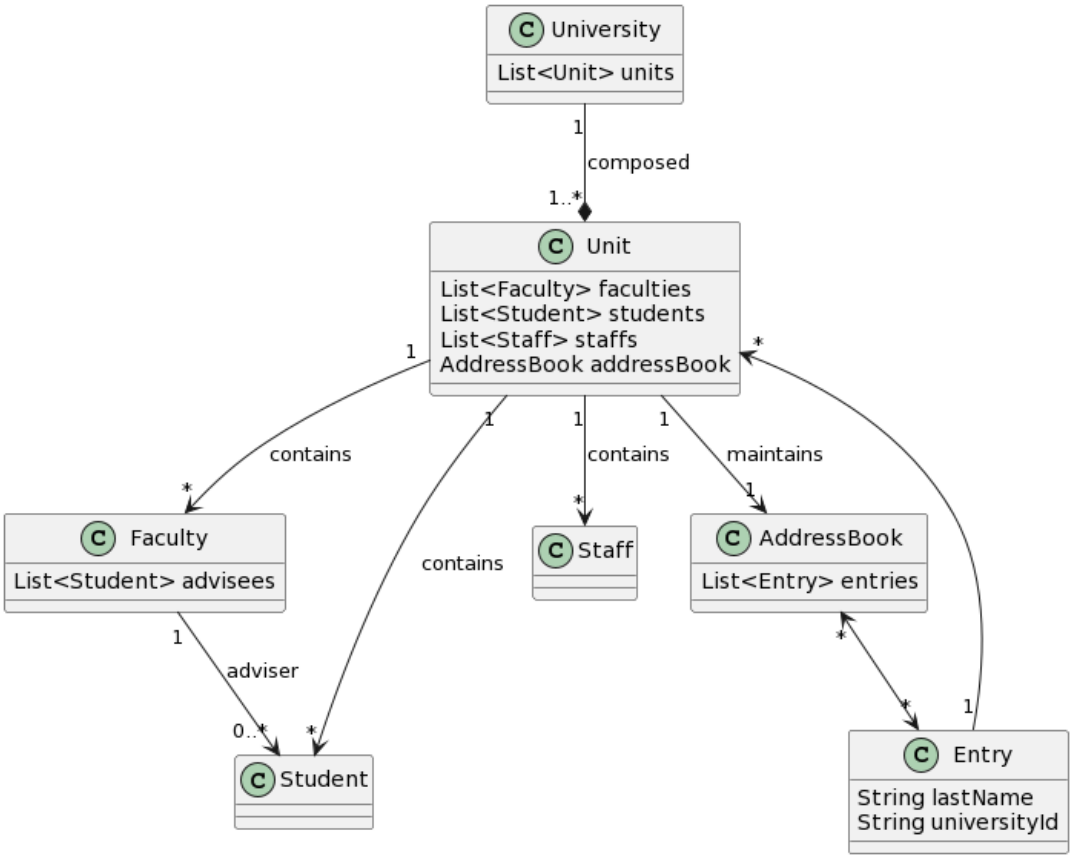
(b)

A CargoPlane aggregates zero or more Pallets. Each Pallet aggregates zero or more Boxes. A CargoPlane is composed of one or more Parts. Each Box contain one or more Items that are accessed by part numbers.



(c)

A University is composed of one or more Units, such as Colleges and Schools. Each Unit contains Faculty, Students, and Staff. A Unit maintains an AddressBook filled with Entries, and one Entry for each type of Person contained in that Unit. An Entry can be located in the AddressBook by supplying their last Name or their UniversityId. Faculty members can be the adviser of zero or more Students.



2.

This would break abstraction and seriously degrade the code maintainability. Once developer wants to use getArea() with all shapes, it can't work correctly with Square, and Shape is no longer obey polymorphism.

3.

(a)

```
class Animal:
    def __init__(self, name):
        self.name = name
    def sleep(self):
        print(f"{self.name} is sleeping.")
    def play(self):
        print(f"{self.name} is playing.")
    def takeForWalk(self):
        print(f"{self.name} is being taken for a walk.")
    def makeNoise():
        print("aaa")
class Pachyderm(Animal):
    def roam():
        print("roam")
class Feline(Animal):
    def roam():
        print("roam")
class Canine(Animal):
    def roam():
        print("roam")

class Rhino(Pachyderm):
    pass
class Hippo(Pachyderm):
    pass
class Elephant(Pachyderm):
    pass

class Cat(Feline):
    pass
class Tiger(Feline):
    pass
class Lion(Feline):
    pass
class Dog(Canine):
    pass
```

```
class Wolf(Canine):
    pass


dog = Dog("brave")
dog.play()
dog.takeForWalk()

cat = Cat("mimi")
cat.play()
cat.takeForWalk()
```

(b)

```
class Animal:
    def __init__(self, name):
        self.name = name
    def sleep(self):
        print(f"{self.name} is sleeping.")

class Pet:
    def __init__(self, name):
        self.name = name
    def play(self):
        print(f"{self.name} is playing.")
    def takeForWalk(self):
        print(f"{self.name} is being taken for a walk.")


class Pachyderm(Animal):
    def roam(self):
        print(f"{self.name} is roaming.")


class Feline(Animal):
    def roam(self):
        print(f"{self.name} is roaming.")
```

```
class Canine(Animal):
    def roam(self):
        print(f"{self.name} is roaming.")


class Hippo(Pachyderm):
    def makeNoise(self):
        print(f"{self.name} is making noise.")


class Rhino(Pachyderm):
    def makeNoise(self):
        print(f"{self.name} is making noise.")


class Elephant(Pachyderm):
    def makeNoise(self):
        print(f"{self.name} is making noise.")


class Tiger(Feline):
    def makeNoise(self):
        print(f"{self.name} is making noise.")


class Cat(Feline, Pet):
    def makeNoise(self):
        print(f"{self.name} is making noise.")


class Lion(Feline):
    def makeNoise(self):
        print(f"{self.name} is making noise.")


class Dog(Canine):
    def makeNoise(self):
        print(f"{self.name} is making noise.")
```

```
class Wolf(Canine):
    def makeNoise(self):
        print(f"{self.name} is making noise.")


hippo = Hippo("popo")
hippo.sleep()
hippo.roam()
hippo.makeNoise()


cat = Cat("mimi")
cat.sleep()
cat.roam()
cat.makeNoise()
cat.play()
cat.takeForWalk()
```