ĐỀ KIỂM TRA

Thời gian: 120 phút

Câu 1: Một Ward gồm có name (string) và danh sách của mọi người trong Ward. Một người person có thể là student, doctor, hoặc teacher. Một student gồm có name, yob (int) (năm sinh), và grade (string). Một teacher gồm có name, yob, và subject (string). Một doctor gồm có name, yob, và specialist (string). Lưu ý cần sử dụng a list để chứa danh sách của mọi người trong Ward.

(a) Thực hiện các class student, doctor, và teacher theo mô tả trên. Thực hiện describe() method để print ra tất cả thông tin của các objects.

(b) Viết addPerson(person) method trong Ward class để add thêm một người mới với nghề nghiệp bất kỳ (student, teacher, doctor) vào danh sách người của ward. Tạo ra một ward object, và thêm vào 1 student, 2 teacher, và 2 doctor. Thực hiện describe() method để in ra tên ward (name) và toàn bộ thông tin của từng người trong ward.

(c) Viết countDoctor() method để đếm số lượng doctor trong ward.

(d) Viết sortAge() method để sort mọi người trong ward theo tuổi của họ với thứ tự tăng dần. (hint: Có thể sử dụng sort của list hoặc viết thêm function đều được)

(e) Viết aveTeacherYearOfBirth() method để tính trung bình năm sinh của các teachers trong ward.

Câu 2: Giả sử công ty có 2 loại nhân viên: Nhân viên văn phòng và Nhân viên sản xuất. Viết chương trình quản lý và tính lương cho từng nhân viên của công ty. Mỗi nhân viên cần quản lý các thông tin sau: Họ tên, ngày sinh, địa chỉ. Trong đó **địa chỉ** là 1 đối tượng, cần lưu trữ thông tin cụ thể: *số nhà, tên đường, tên quận, thành phố.*

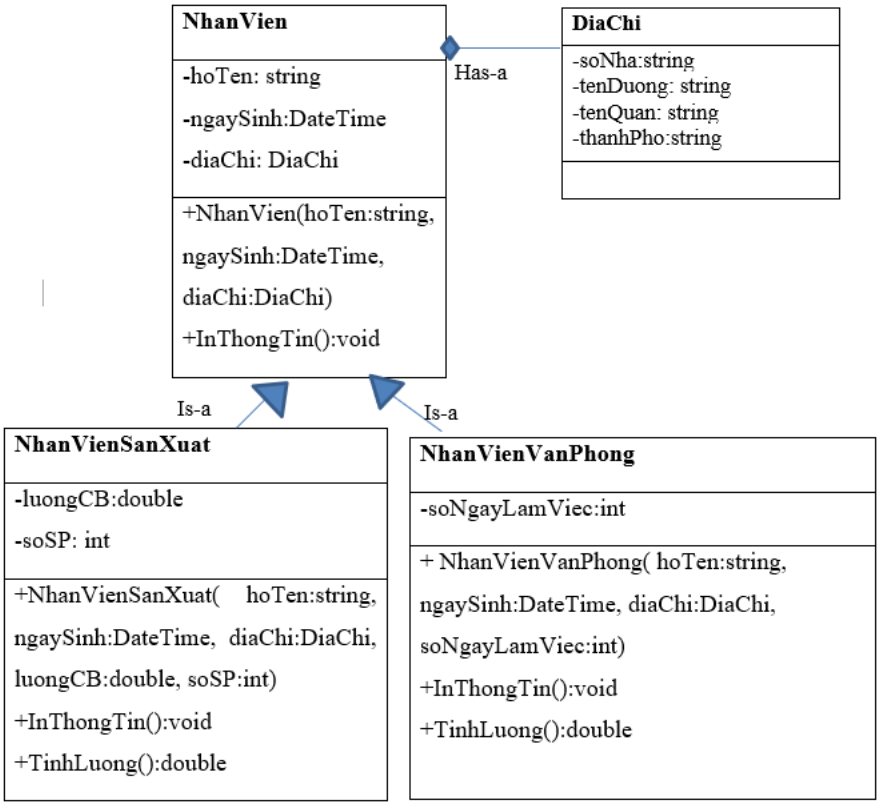
Công ty cần tính lương cho nhân viên như sau:

- Đối với nhân viên sản xuất: Lương = lương căn bản + số sản phẩm * 5000

  Đối với nhân viên văn phòng: Lương = số ngày làm việc * 100000

Tại chương trình chính hãy tạo ra 1 danh sách gồm các Nhân viên sản xuất và 1 danh sách gồm các nhân viên văn phòng. Gọi các chức năng in thông tin nhân viên, xem lương của từng nhân viên.

Gợi ý sơ đồ lớp:



Câu 3: Trong một ứng dụng có quản lý bất động sản của công ty ABC có 4 loại đối tượng là: đất trồng, nhà ở, biệt thự và khách sạn. Biết rằng tất cả đối tượng này đều có các thông tin sau: mã số, chiều dài, chiều rộng và phương thức khởi tạo, in thông tin, tính giá trị. Biết rằng giá trị được tính như sau:

- Đất trồng: giá trị = diện tích * 30,000,000;
- Nhà ở có thêm thông tin về số lầu: giá trị = diện tích * 60,000,000 + số lầu * 100,000,000.
- Khách sạn có thêm thông tin về số sao: giá trị = diện tích * 70,000,000 + số sao * 50,000,000
- Biệt thự: giá trị = diện tích * 100,000,000

Trong 4 loại bất động sản trên thì có 2 loại là biệt thự và khách sạn phải đóng phí kinh doanh. Phí kinh doanh được tính như sau:

- Biệt thự: chiều rộng * 5000
- Khách sạn: chiều rộng * 10000

Yêu cầu:

1. Thiết kế các lớp BatDongSan (lớp cha) và các lớp con: DatTrong, NhaO, KhachSan, BietThu với các thuộc tính và chức năng như mô tả. Thiết kế giao diện PhiKinhDoanh chứa một chức năng tính phí kinh doanh. Sau đó 2 lớp BietThu và KhachSan sẽ triển khai giao diện PhiKinhDoanh.

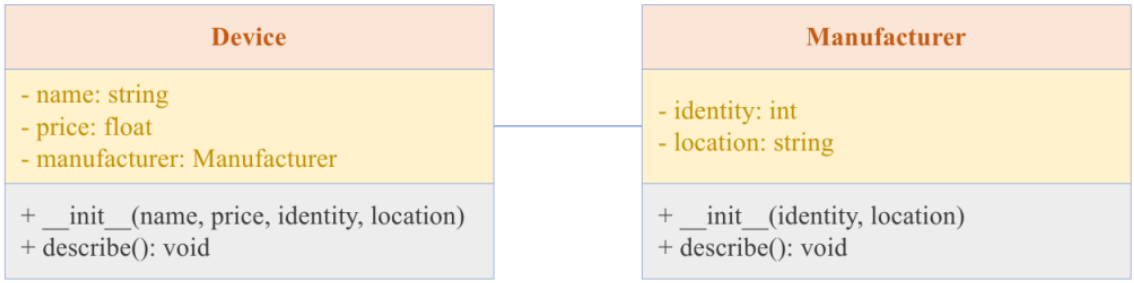
2.Tại chương trình chính xây dựng các chức năng sau:

a) Nhập 1 danh sách bất động sản
b) Cho biết số lượng bất động sản theo từng loại
c) In ra danh sách các bất động sản có diện tích trên 1000
d) Tính tổng phí dinh doanh thu được

II. Câu hỏi trắc nghiệm

- Đọc tự luận trước để nắm được idea tổng quát (sẽ không yêu cầu nhưng khuyến khích các bạn tự làm tự luận) và các bài này sẽ được giải trong buổi TA.
- Các bạn phải làm phần trắc nghiệm
  - Các câu hỏi có ký hiệu **(LT)**: là các câu hỏi lý thuyết và/hoặc đã có sẵn trong file hint nên các bạn chỉ cần hiểu và chạy lại để chọn được đáp án đúng
  - Các câu hỏi có ký hiệu (Code): là câu hỏi yêu cầu các bạn phải trực tiếp code vào phần bị khuyết để có thể chọn được đáp án đúng
  - **Lưu ý:** Đối với dạng câu hỏi (Code) trong file hint luôn có 1 test casse bắt đầu với từ khóa assert nếu các bạn chạy không báo lỗi có nghĩa các bạn đã vượt qua được test case này và chạy lệnh tiếp theo để trả lời câu hỏi trắc nghiệm
  - **Lưu ý:** Đọc kỹ các code gợi ý và code ví dụ mẫu ở tự luận có thể sẽ có ích cho các bạn khi làm trắc nghiệm

Câu hỏi 1 **(LT)** : Theo thiết kế này thì mối quan hệ giữa 2 class này được gọi là gì?



Hình 4: Class Diagram

a) Aggregation
b) Composition
c) Inheritance
d) Tất cả đều sai

Câu hỏi 2 **(LT)** : Đầu ra của chương trình sau đây là gì?

```python
class Manufacturer:
    def __init__(self, identity:int, location: str ):
        self.__identity = identity
        self.__location = location

    def describe(self):
        print(f"Identity: {self.__identity} - Location: {self.__location}")

manu1 = Manufacturer(identity=100, location='Vietnam')
manu1.describe()
```

a) Identity: 100 - Location: Vietnam
b) Identity: Vietnam - Location: 100
c) None
d) Raise an error

Câu hỏi 3 (LT) : Đoạn code sau đây thể hiện mối quan hệ gì?

```
class Manufacturer:
    def __init__(self, identity:int, location: str ):
        self.__identity = identity
        self.__location = location

    def describe(self):
        print(f"Identity: {self.__identity} - Location: {self.__location}")


class Device:
    def __init__(self, name:str, price:float, identity:int, location:str ):
        self.__name = name
        self.__price = price
        self.__manufacturer = Manufacturer(identity, location)

    def describe(self):
        print(f"Name: {self.__name} - Price: {self.__price}")
        self.__manufacturer.describe()
device1 = Device(name="touchpad", price=3.3, identity=1111, location="Vietnam")
device1.describe()
```

a) Aggregation
b) Composition
c) Inheritance
d) Tất cả đều sai

Câu hỏi 4 (LT) : Đoạn code sau đây thể hiện mối quan hệ gì?

```
class Manufacturer:
    def __init__(self, identity:int, location: str ):
        self.__identity = identity
        self.__location = location

    def describe(self):
        print(f"Identity: {self.__identity} - Location: {self.__location}")


class Device:
    def __init__(self, name:str, price:float, manu:Manufacturer):
        self.__name = name
        self.__price = price
        self.__manufacturer = manu

    def describe(self):
        print(f"Name: {self.__name} - Price: {self.__price}")
        self.__manufacturer.describe()

manu1 = Manufacturer(identity=1111, location='Vietnam')
device1 = Device(name="touchpad", price=3.3, manu=manu1)
device1.describe()
```

a) Aggregation
b) Composition
c) Inheritance
d) Tất cả đều sai

Câu hỏi 5 (Code) : Một người (person) có thể là student, doctor, hoặc teacher. Một student gồm có name (string), yob (int) (năm sinh), và grade (string). Các bạn thực hiện viết class Student theo mô tả trên (Các bạn sẽ viết thêm describe() method để print ra tất cả thông tin của object) và kết quả đầu ra là gì? Chọn đáp án đúng nhất bên dưới.

```
from abc import ABC, abstractmethod

class Person(ABC):
    def __init__(self, name:str, yob:int):
        self._name = name
        self._yob = yob

    def getYoB(self):
        return self._yob

    @abstractmethod
    def describe(self):
        pass


class Student(Person):
    def __init__(self, name:str, yob:int, grade:str):
        #########YOUR CODE HERE#########

        ################################
    def describe(self):
        #########YOUR CODE HERE#########

        ################################

student1 = Student(name="studentZ2023", yob=2011, grade="6")
student1.describe()
```

a) Student - Name: studentZ2023 - YoB: 2011 - Grade: 6
b) Student - Name: studentZ2023 - YoB: 6 - Grade: 2011
c) Student - Name: 6 - YoB: studentZ2023 - Grade: 2011
d) Tất cả đều sai

Câu hỏi 6 (Code) : Một người (person) có thể là student, doctor, hoặc teacher. Một teacher gồm có name (string), yob (int), và subject (string). Các bạn thực hiện viết class Teacher theo mô tả trên (Các bạn sẽ viết thêm describe() method để print ra tất cả thông tin của object) và kết quả đầu ra là gì? Chọn đáp án đúng nhất bên dưới.

```
from abc import ABC, abstractmethod

class Person(ABC):
    def __init__(self, name:str, yob:int):
        self._name = name
        self._yob = yob

    def getYoB(self):
        return self._yob

    @abstractmethod
```

```
    def describe(self):
        pass


class Teacher(Person):
    def __init__(self, name:str, yob:int, subject:str):
        #########YOUR CODE HERE#########

        ################################

    def describe(self):
        #########YOUR CODE HERE#########

        ################################

teacher1 = Teacher(name="teacherZ2023", yob=1991, subject="History")
teacher1.describe()
```

a) Teacher - Name: teacherZ2023 - YoB: 1991 - Subject: History
b) Teacher - Name: 1991 - YoB: teacherZ2023 - Subject: History
c) Teacher - Name: History - YoB: teacherZ2023 - Subject: 1991
d) Tất cả đều sai

Câu hỏi 7 (Code) : Một người (person) có thể là student, doctor, hoặc teacher. Một doctor gồm có name (string), yob (string), và specialist (string). Các bạn thực hiện viết class Teacher theo mô tả trên (Các bạn sẽ viết thêm describe() method để print ra tất cả thông tin của object) và kết quả đầu ra là gì? Chọn đáp án đúng nhất bên dưới.

```
from abc import ABC, abstractmethod

class Person(ABC):
    def __init__(self, name:str, yob:int):
        self._name = name
        self._yob = yob

    def getYoB(self):
        return self._yob

    @abstractmethod
    def describe(self):
        pass


class Doctor(Person):
    def __init__(self, name:str, yob:int, specialist:str):
        #########YOUR CODE HERE#########

        ################################

    def describe(self):
        #########YOUR CODE HERE#########

        ################################

doctor1 = Doctor(name="doctorZ2023", yob=1981, specialist="Endocrinologists")
doctor1.describe()
```

a) Doctor - Name: doctorZ2023 - YoB: 1981 - Specialist: Endocrinologists
b) Doctor - Name: 1981 - YoB: doctorZ2023 - Specialist: Endocrinologists

c) Doctor - Name: Endocrinologists - YoB: 1981 - Specialist: doctorZ2023
d) Tất cả đều sai

Câu hỏi 8 (Code) : Một Ward gồm có name (string) và danh sách của mọi người trong Ward. Một người person có thể là student, doctor, hoặc teacher và cần sử dụng một list để chứa danh sách của mọi người trong Ward. Viết addPerson(person) method trong Ward class để add thêm một người mới với nghề nghiệp bất kỳ (student, teacher, doctor) vào danh sách người của ward. Tạo ra một ward object, và thêm vào 1 student, 2 teacher, và 2 doctor (sử dụng code ở câu 5,6,7 để tạo person mong muốn). Thực hiện describe() method để in ra tên ward (name) và toàn bộ thông tin của từng người trong ward. Chọn đáp án đúng nhất bên dưới.

```
class Ward:
    def __init__(self, name:str):
        #########YOUR CODE HERE#########

        ################################

    def addPerson(self, person:Person):
        #########YOUR CODE HERE#########

        ################################

    def describe(self):
        #########YOUR CODE HERE#########

        ################################

student1 = Student(name="studentK-111", yob=2012, grade="5")
teacher1 = Teacher(name="teacherK-222", yob=1966, subject="Math")
doctor1 = Doctor(name="doctorK-333", yob=1965, specialist="Endocrinologists")
teacher2 = Teacher(name="teacherK-444", yob=1945, subject="History")
doctor2 = Doctor(name="doctorK-555", yob=1975, specialist="Cardiologists")
ward1 = Ward(name="Ward11")
ward1.addPerson(student1)
ward1.addPerson(teacher1)
ward1.addPerson(teacher2)
ward1.addPerson(doctor1)
ward1.addPerson(doctor2)
ward1.describe()
```

a) Ward Name: Ward11
Student - Name: studentK-111 - YoB: 2012 - Grade: 5
Teacher - Name: teacherK-222 - YoB: 1966 - Subject: Math
Teacher - Name: teacherK-444 - YoB: 1945 - Subject: History
Doctor - Name: doctorK-333 - YoB: 1965 - Specialist: Endocrinologists
Doctor - Name: doctorK-555 - YoB: 1975 - Specialist: Cardiologists
b) Ward Name: Ward22
Student - Name: studentK-111 - YoB: 2012 - Grade: 5
Teacher - Name: Math - YoB: teacherK-222 - Subject: 1966
Teacher - Name: teacherK-444 - YoB: 1945 - Subject: History
Doctor - Name: 1965 - YoB: doctorK-333 - Specialist: Endocrinologists
Doctor - Name: doctorK-555 - YoB: 1975 - Specialist: Cardiologists
c) Tất cả đều đúng
d) Tất cả đều sai

Câu hỏi 9 (LT) : Thực hiện xây dựng class Stack với các chức năng (method) sau đây: **initialization**

method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa. **.isEmpty()**: kiểm tra stack có đang rỗng. Kết quả đầu ra là gì?

```
class MyStack:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__stack = []

    def isEmpty(self):
        return len(self.__stack) == 0

stack1 = MyStack(capacity=5)
print(stack1.isEmpty())
```

a) True
b) False
c) None
d) Raise an error

Câu hỏi 10 (LT) : Thực hiện xây dựng class Stack với các chức năng (method) sau đây: **initialization** method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa (tại đây stack cũng đã được thêm vào các element). **.isFull()**: kiểm tra stack đã full chưa. Kết quả đầu ra là gì?

```
class MyStack:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__stack = [1,2,3,4,5]

    def isFull(self):
        return len(self.__stack) == self.__capacity

stack1 = MyStack(capacity=5)
print(stack1.isFull())
```

a) True
b) False
c) None
d) Raise an error

Câu hỏi 11 (Code) : Thực hiện xây dựng class Stack với các chức năng (method) sau đây: **initialization** method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa. **.isFull()**: kiểm tra stack đã full chưa. **.push(value)** add thêm value vào trong stack. Kết quả đầu ra là gì?

```
class MyStack:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__stack = []

    def isFull(self):
        return len(self.__stack) == self.__capacity

    def push(self, value):
        #########YOUR CODE HERE#########

        ################################

```

```
stack1 = MyStack(capacity=5)
stack1.push(1)
stack1.push(2)
print(stack1.isFull())
```

a) True
b) False
c) None
d) Raise an error

Câu hỏi 12 (Code) : Thực hiện xây dựng class Stack với các chức năng (method) sau đây: **initialization** method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa. **.isEmpty()**: kiểm tra stack có đang rỗng. **.isFull()**: kiểm tra stack đã full chưa. **.push(value)** add thêm value vào trong stack. **.top()** lấy giá trị top element hiện tại của stack, nhưng không loại bỏ giá trị đó. Kết quả đầu ra là gì?

```
class MyStack:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__stack = []

    def isEmpty(self):
        return len(self.__stack) == 0

    def isFull(self):
        return len(self.__stack) == self.__capacity

    def push(self, value):
        #########YOUR CODE HERE#########

        ################################

    def top(self):
        #########YOUR CODE HERE#########

        ################################

stack1 = MyStack(capacity=5)
stack1.push(1)
stack1.push(2)
print(stack1.top())
```

a) 1
b) 2
c) None
d) Raise an error

Câu hỏi 13 (LT) : Thực hiện xây dựng class Queue với các chức năng (method) sau đây **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa.**.isEmpty()**: kiểm tra queue có đang rỗng. Kết quả đầu ra là gì?

```
class MyQueue:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__queue = []

    def isEmpty(self):
        return len(self.__queue) == 0
```

```
8
9  queue1 = MyQueue(capacity=5)
10 print(queue1.isEmpty())
```

a) True
b) False
c) None
d) Raise an error

Câu hỏi 14 (LT) : Thực hiện xây dựng class Queue với các chức năng (method) sau đây **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa. **.isFull()**: kiểm tra queue đã full chưa. Kết quả đầu ra là gì?

```
1  class MyQueue:
2      def __init__(self, capacity):
3          self.__capacity = capacity
4          self.__queue = []
5
6      def isFull(self):
7          return len(self.__queue) == self.__capacity
8
9
10 queue1 = MyQueue(capacity=5)
11 print(queue1.isFull())
```

a) True
b) False
c) None
d) Raise an error

Câu hỏi 15 (Code) : Thực hiện xây dựng class Queue với các chức năng (method) sau đây **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa. **.isFull()**: kiểm tra queue đã full chưa. **.enqueue(value)** add thêm value vào trong queue. Kết quả đầu ra là gì?

```
1  class MyQueue:
2      def __init__(self, capacity):
3          self.__capacity = capacity
4          self.__queue = []
5
6      def isFull(self):
7          return len(self.__queue) == self.__capacity
8
9      def enqueue(self, value):
10         #########YOUR CODE HERE#########
11
12         ################################
13
14 queue1 = MyQueue(capacity=5)
15 queue1.enqueue(1)
16 queue1.enqueue(2)
17 print(queue1.isFull())
```

a) True
b) False
c) None
d) Raise an error

Câu hỏi 16 (Code) : Thực hiện xây dựng class Queue với các chức năng (method) sau đây **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa. **.isFull()**: kiểm tra queue đã full chưa. **.enqueue(value)** add thêm value vào trong queue. **.front()** lấy giá trị first element hiện tại của queue, nhưng không loại bỏ giá trị đó . Kết quả đầu ra là gì?

```
class MyQueue:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__queue = []

    def isEmpty(self):
        return len(self.__queue) == 0

    def isFull(self):
        return len(self.__queue) == self.__capacity

    def enqueue(self, value):
        #########YOUR CODE HERE#########

        ################################

    def front(self):
        #########YOUR CODE HERE#########

        ################################
queue1 = MyQueue(capacity=5)
queue1.enqueue(1)
queue1.enqueue(2)
print(queue1.front())
```

a) 1
b) 2
c) None
d) Raise an error