Câu 1:

Moi quan he giua hai class la Association.

Chon D

Câu 2:

```
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}')

manul = Manufacturer(identity=100, location="Vietnam")
manul.describe()
```

Khi chay chuong trinh ket qua la:

Identity: 100 - Location: Vietnam. Chon A

s

Câu 3:

```
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()
```

Khi chay chuong trinh ket qua la:

```
#Name: touchpad - Price: 3.3
#Identity: 1111 - Location: Vietnam
```

Hai class moi quan he Composition. Chon B

Câu 4:

```
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()
```

Khi chay chuong trinh tren ket qua la:

#Name: touchpad - Price: 3.3

#Identity: 1111 - Location: Vietnam

Hai class moi quan he Agreeation. Chon A

Câu 5:

```
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):
        super().__init__(name, yob)
        self._grade = grade

    def describe(self):
        print(f'Name: {self._name}, Year of Birth: {self._yob}, Grade: {self._grade}')


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

Kết quả là: Name: studentZ2023, Year of Birth: 2011, Grade: 6

Chọn A

Câu 6:

```
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):
        super().__init__(name, yob)
        self._subject = subject

    def describe(self):
        print(f'Name: {self._name}, Year of Birth: {self._yob}, Subject: {self._subject}')


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

Kết quả là: Name: teacherZ2023, Year of Birth: 1991, Subject: History

Chọn A

Câu 7:

```
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):
        super().__init__(name, yob)
        self._specialist = specialist

    def describe(self):
        print(f'Name: {self._name}, Year of Birth: {self._yob}, Specialist: {self._specialist}')


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

Chọn A

Câu 8:

```
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):
        super().__init__(name, yob)
        self._grade = grade

    def describe(self):
        print(f'Name: {self._name}, Year of Birth: {self._yob}, Grade: {self._grade}')

class Teacher(Person):
    def __init__(self, name: str, yob: int, subject: str):
        super().__init__(name, yob)
        self._subject = subject
    def describe(self):
        print(f'Name: {self._name}, Year of Birth: {self._yob}, Subject: {self._subject}')

class Doctor(Person):
    def __init__(self, name: str, yob: int, specialist: str):
        super().__init__(name, yob)
        self._specialist = specialist

    def describe(self):
        print(f'Name: {self._name}, Year of Birth: {self._yob}, Specialist: {self._specialist}')

class Ward:
    def __init__(self, name: str):
        self._name = name
        self._people = []  # Danh sách chứa mọi người trong Ward

    def addPerson(self, person: Person):
        self._people.append(person)  # Thêm người vào danh sách

    def describe(self):
        print(f'Ward Name: {self._name}')
        for person in self._people:
            person.describe()  # Gọi phương thức describe của từng người

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()
```

Kết quả là:

```
Ward Name: Ward11
Name: studentK-111, Year of Birth: 2012, Grade: 5
Name: teacherK-222, Year of Birth: 1966, Subject: Math
Name: teacherK-444, Year of Birth: 1945, Subject: History
Name: doctorK-333, Year of Birth: 1965, Specialist: Endocrinologists
Name: doctorK-555, Year of Birth: 1975, Specialist: Cardiologists
```

Chọn A

Câu 9:

Chọn A

```
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())
```

Câu 10:

Chọn A

```
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())
```

Câu 11:

```
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):
        if not self.isFull():
            self.__stack.append(value)
        else:
            print("Stack is full!")
    def top(self):
        if not self.isEmpty():
            return self.__stack[-1]
        else:
            print("Stack is empty!")

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

Kết quả chương trình là: 2. Chọn B

Câu 12: Dùng đoạn code của bài 11: Kết quả chương trình là 2. Chọn B

Câu 13:

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

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

queue1 = MyQueue(capacity=5)
print(queue1.isEmpty())
```

Kết quả là True. Chọn A

Câu 14:

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

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

queue1 = MyQueue(capacity=5)
print(queue1.isFull())
```

Kết quả là False. Chọn B

Câu 15:

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

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

    def enqueue(self, value):
        if not self.isFull():
            self.__queue.append(value)
        else:
            print("Hàng đợi đã đầy!")

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

Kết quả là: False. Chọn B

Câu 16:

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

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

    def enqueue(self, value):
        if len(self.__queue) < self.__capacity:
            self.__queue.append(value)
        else:
            print("Queue is full!")

    def front(self):
        if not self.isEmpty():
            return self.__queue[0]
        else:
            print("Is Empty!")

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

Kết quả là: 1. Chọn A