Lab 7

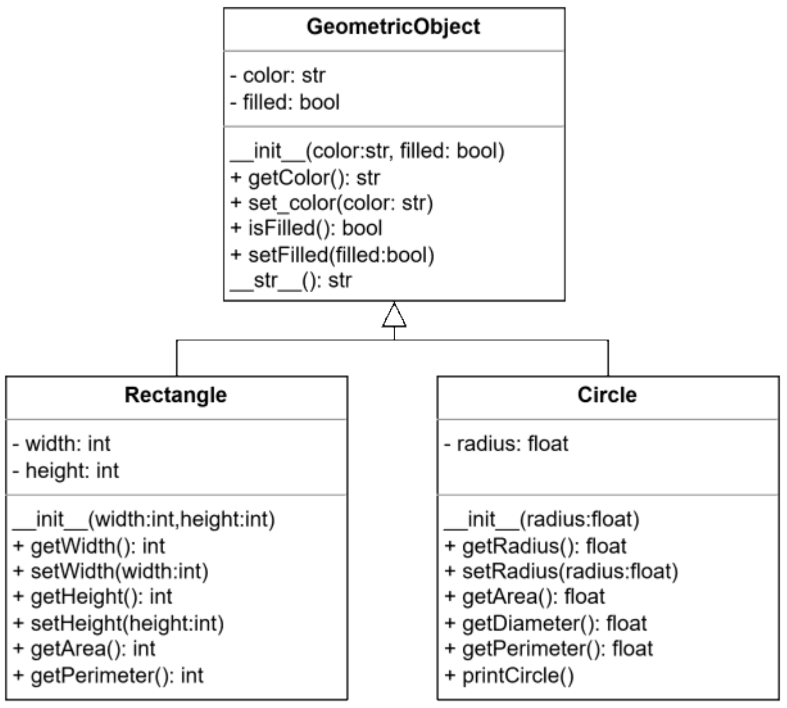
November 3, 2024 12:36 PM

Q1:

Draw the complete class diagram for the program below. Include the complete class diagram for each of the classes as well as correctly showing the relationships.

Consider the following code:

```
import math
                                                str                 bool
class GeometricObject:
    def __init__(self, color = "green", filled = True):
        self.__color = color
        self.__filled = filled
    def getColor(self):
        return self.__color → str
    def setColor(self, color):
        self.__color = color
    def isFilled(self):
        return self.__filled → bool
    def setFilled(self, filled):
        self.__filled = filled
    def __str__(self):
        return "color: " + self.__color + \
            " and filled: " + str(self.__filled) → str

class Circle(GeometricObject):
    def __init__(self, radius):
        super().__init__()
        self.__radius = radius
    def getRadius(self):
        return self.__radius → float
    def setRadius(self, radius):
        self.__radius = radius
    def getArea(self):
        return self.__radius * self.__radius * math.pi → float
    def getDiameter(self):
        return 2 * self.__radius → float
    def getPerimeter(self):
        return 2 * self.__radius * math.pi → float
    def printCircle(self):
        print(self.__str__() + " radius: " + str(self.__radius))

                                           int        int
class Rectangle(GeometricObject):
    def __init__(self, width = 1, height = 1):
        super().__init__()
        self.__width = width
        self.__height = height
    def getWidth(self):
        return self.__width → int
    def setWidth(self, width):
        self.__width = width
    def getHeight(self):
        return self.__height → int
    def setHeight(self, height):
        self.__height = self.__height
    def getArea(self):
        return self.__width * self.__height → int
    def getPerimeter(self):
        return 2 * (self.__width + self.__height) → int
if __name__ == "__main__":
    circle = Circle(1.5)
    print("A circle", circle)
    print("The radius is", circle.getRadius())
    print("The area is", circle.getArea())
    print("The diameter is", circle.getDiameter())
    rectangle = Rectangle(2, 4)
    print("\nA rectangle", rectangle)
    print("The area is", rectangle.getArea())
    print("The perimeter is", rectangle.getPerimeter())
```



super class

sub class

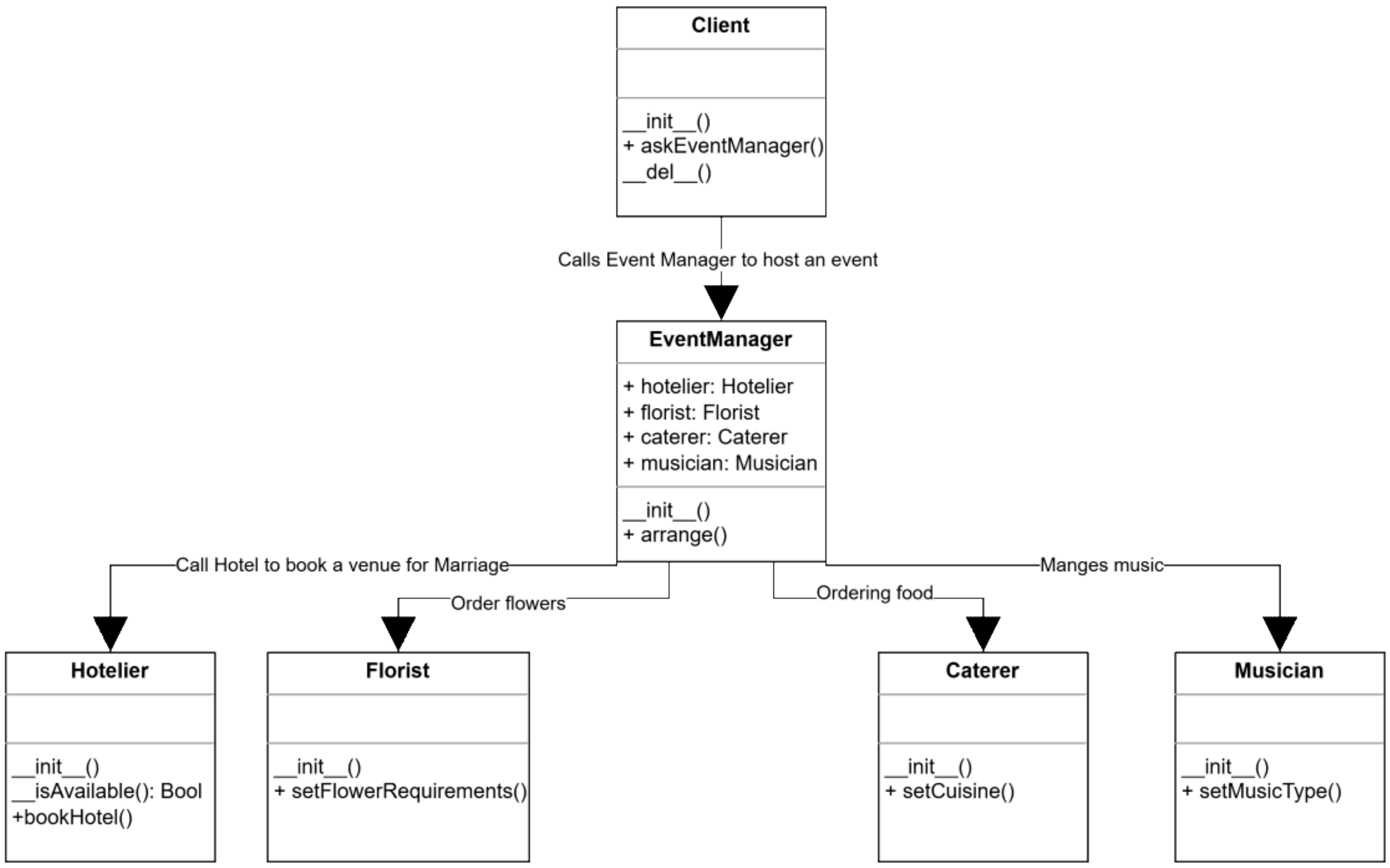
Q2.

Draw the class diagram for the program below. For this question, for each individual class, use the simplified class diagram (that is, a rectangle with the class name in it).

Consider the following code:

```
class EventManager:
    def __init__(self):
        print("Event Manager: Let me talk to the folks\n")

    def arrange(self):
        self.hotelier = Hotelier()
        self.hotelier.bookHotel()

        self.florist = Florist()
        self.florist.setFlowerRequirements()

        self.caterer = Caterer()
        self.caterer.setCuisine()

        self.musician = Musician()
        self.musician.setMusicType()

class Hotelier:
    def __init__(self):
        print("Arranging the Hotel for Marriage? --")
```

```python
class Hotelier:
    def __init__(self):
        print("Arranging the Hotel for Marriage? --")

    def __isAvailable(self):
        print("Is the Hotel free for the event on given day?")
        return True

    def bookHotel(self):
        if self.__isAvailable():
            print("Registered the Booking\n")

class Florist:
    def __init__(self):
        print("Flower Decorations for the Event? --")

    def setFlowerRequirements(self):
        print("Carnations, Roses and Lilies would be used for Decorations\n")

class Caterer:
    def __init__(self):
        print("Food Arrangements for the Event --")

    def setCuisine(self):
        print("Chinese & Continental Cuisine to be served\n")

class Musician:
    def __init__(self):
        print("Musical Arrangements for the Marriage --")

    def setMusicType(self):
        print("Jazz and Classical will be played\n")

class Client:
    def __init__(self):
        print("Client: Whoa! My best friend is getting married!")
    def askEventManager(self):
        print("Client: Let's Contact the Event Manager\n")
        em = EventManager()
        em.arrange()
    def __del__(self):
        print("Client: Thanks to Event Manager, all preparations done! Phew!")

client = Client()
client.askEventManager()
```

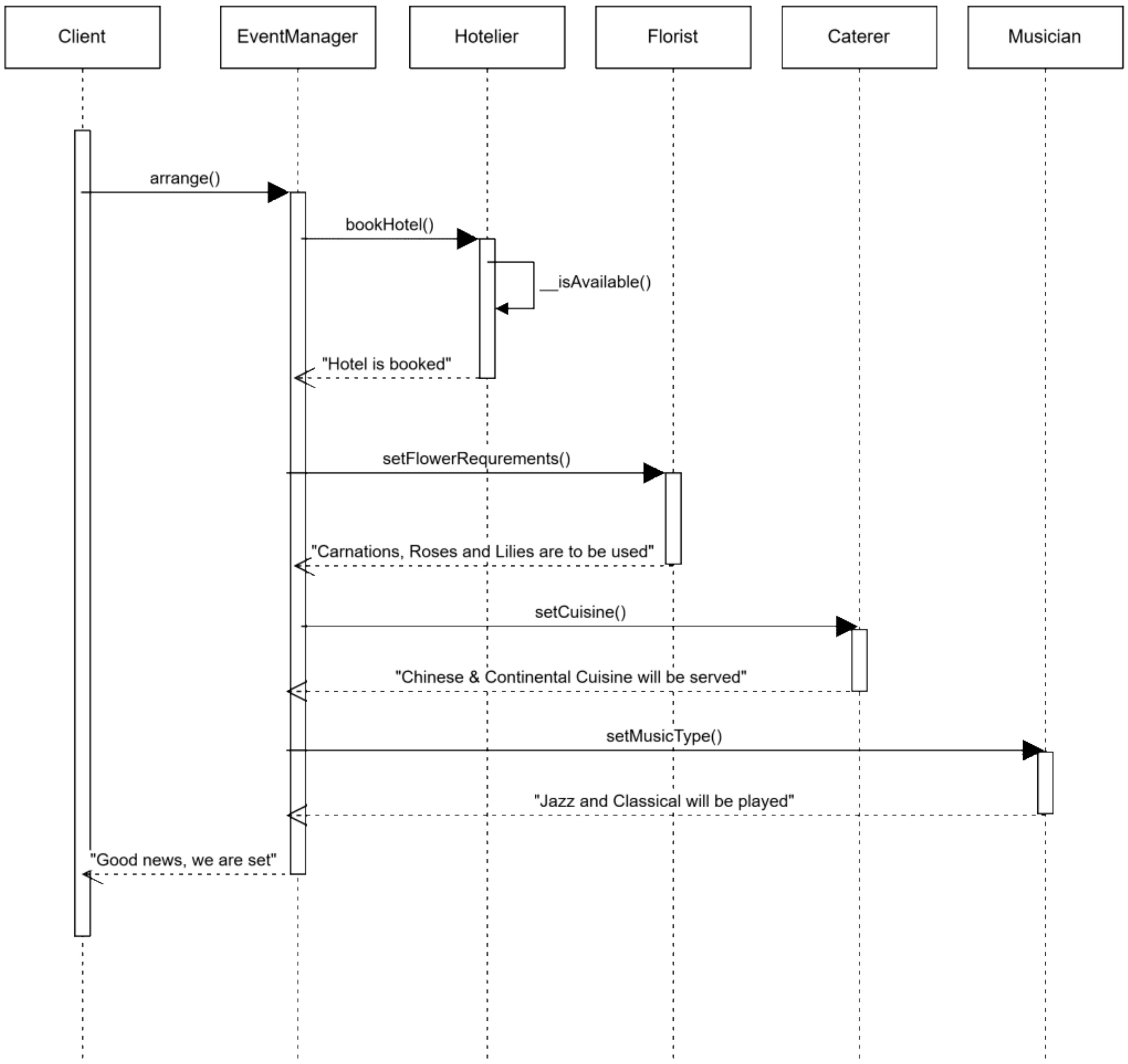


Q3.

Draw the sequence diagram for the program below. The code from the previous question has been slightly modified and some comment statements have been added.

The comment statements indicate the call messages and response messages to be used for the sequence diagram (note that for simplification, only some necessary ones are selected). For example, the **bookHotel** method call on the **hotelier** object is marked as a call message (so the message **bookHotel** from **eventManager** object to **hotelier** object), and the relevant print message **Hotel is booked** done by that method is marked as the response message to the caller (so the response message **Hotel is booked** from the **hotelier** object to **eventManager** object).

```
class EventManager:
    def __init__(self):
        print("Event Manager: Let me talk to the folks\n")

    def arrange(self):
        self.hotelier = Hotelier()
        self.hotelier.bookHotel()  #a call message (2)

        self.florist = Florist()
        self.florist.setFlowerRequirements()  #a call message (3)

        self.caterer = Caterer()
        self.caterer.setCuisine()  #a call message (4)

        self.musician = Musician()
        self.musician.setMusicType()  #a call message (5)
        print("Good news, we are set\n") #a response message (5)
class Hotelier:
    def __init__(self):
        print("Arranging the Hotel for Marriage? --")

    def __isAvailable(self):
        print("Is the Hotel free for the event on given day?")
        return True

    def bookHotel(self):
        if self.__isAvailable():
            print("Hotel is booked\n")  #a response message   ①

class Florist:
    def __init__(self):
        print("Flower Decorations for the Event? --")

    def setFlowerRequirements(self):
        print("Carnations, Roses and Lilies are to be used\n") #a response message   ②

class Caterer:
    def __init__(self):
        print("Food Arrangements for the Event --")

    def setCuisine(self):
        print("Chinese & Continental Cuisine will be served\n") #a response message   ③

class Musician:
    def __init__(self):
        print("Musical Arrangements for the Marriage --")

    def setMusicType(self):
        print("Jazz and Classical will be played\n")  #a response message   ④

class Client:
    def __init__(self):
        print("Client: Whoa! My best friend is getting married!")
    def askEventManager(self):
        print("Client: Let's Contact the Event Manager\n")
        eventManager = EventManager()
        eventManager.arrange()    #a call message   ①
    def __del__(self):
        print("Client: Thanks to Event Manager, all preparations done! Phew!")
client = Client()
client.askEventManager()
```


Client
EventManager
Hotelier
Florist
Caterer
Musician
arrange()
bookHotel()
__isAvailable()
"Hotel is booked"
setFlowerRequrements()
"Carnations, Roses and Lilies are to be used"
setCuisine()
"Chinese & Continental Cuisine will be served"
setMusicType()
"Jazz and Classical will be played"
"Good news, we are set"