**Name**: Nico Kranni

**Pair:**

**Amount of completed tasks:8**

**Which tasks were left undone or incomplete: 3**

Self-assessment:

This exercise was easy/difficult/ok/etc. for me because…

Doing this exercise, I learned more about loops and how to import classes better!

I am still wondering… about UML

I understood/did not understand that… ; UML is still strange

1. Explain the following terms and what they are used for:

a. Inheritance (in object-oriented programming)

It lets you “inherit” or expand from original class. You can also add more methods or attributes to newer classes without modifying the original.

b. Multiple inheritance

Same as normal inheritance, but there are multiple “parent” classes. It follows depth first and left right order.

c. UML

Unified Modeling Language is modeling language that is used in software engineering. It was designed to standardize visualize design of systems.

It is used to visualize, specify, construct and document software system.

d. UML class diagram

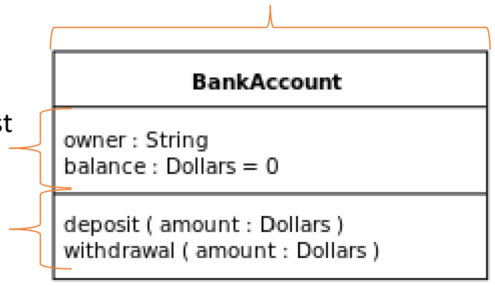
It is a diagram that helps you understand the structure of class. It shows its attributes, operations and relationships.

Name of the class.

Bold, centered and first letter capitalized

Attributes of the class. Left-aligned, first letter lowercase

Operations(methods) of the class. Left-aligned, first letter lowercase



2. True or false?

a. The practice of procedural programming is centered on the creation of objects.

True

b. Object reusability has been a factor in the increased use of object-oriented programming.

True

c. It is a common practice in object-oriented programming to make all of a class's data attributes accessible to statements outside the class.

False

d. Class methods do not have to have a self-parameter.

True

e. Starting an attribute name with two underscores will hide the attribute from code outside the class.

True

f. You cannot directly call the __str__ method.

True

3. Answer the following question: When you model using UML diagrams, why is it important to follow the UML syntax strictly?

## Test report

Write the test report yourself to each coding task (task number, input/action, desired output and then the testing evidence (actual output)). Add rows if necessary. Include answers to theoretical questions and pseudocode to this return document as well in addition to code screen captures. Actual output can be a screen capture of the terminal showing the output.

| **Task** | **Input / action** | **Desired output** | **Actual output (use red color if desired output != actual output)** |
|---|---|---|---|
| **4** | Test if phone works after importing it from another file. | Here is the data that you provided:<br>Manufacturer:  Nokia<br>Model number:  3310<br>Retail price:  150 | Here is the data that you provided:<br>Manufacturer:  Nokia<br>Model number:  3310<br>Retail price:  150 |
| **5** | Make sure there is accessor and mutator methods for all attributes | Here is the data that you provided:<br>Manufacturer:  Nokia<br>Model number:  3310<br>Retail price:  150<br>The ID is:  5 | Here is the data that you provided:<br>Manufacturer:  Nokia<br>Model number:  3310<br>Retail price:  150<br>The ID is:  5 |
| **6** | Test out new ID’s of phones | List of phones with consequence ID’s | List of phones with consequence ID’s |
| **7** | Roll the dice and print phone equal to its ID | Dice chose:<br>Here is the data that you provided:<br>Manufacturer: Nokia<br>Model number: 3310<br>Retail price: 150<br>The ID is: 5 | Dice chose:<br>Here is the data that you provided:<br>Manufacturer: Nokia<br>Model number: 3310<br>Retail price: 150<br>The ID is: 5 |
| **8** | Input:<br>"Volvo","V80","534 miles","5000 €","Blue","600 kg","2x5m" | State of the car:<br>Make: Volvo<br>Model: V80<br>Mileage: 534 miles<br>Price: 5000 €<br>Color: Blue<br>Max load: 600 kg<br>Trunk size: 2x5m | State of the car:<br>Make: Volvo<br>Model: V80<br>Mileage: 534 miles<br>Price: 5000 €<br>Color: Blue<br>Max load: 600 kg<br>Trunk size: 2x5m |
| **9** | Input:<br>"5","Cat","Bläkkis","15m^2","1500g" | Dice chose:<br>State of the Mammal:<br>ID: 3<br>Species: Dog<br>Name: Pata<br>Size: 1m^2<br>Weight: 1000g<br><br>Pata fits into the car! | Dice chose:<br>State of the Mammal:<br>ID: 3<br>Species: Dog<br>Name: Pata<br>Size: 1m^2<br>Weight: 1000g<br><br>Pata fits into the car! |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

## Code

Phone

```python
# Nico Kranni
# Phone.py
# Used to create class CellPhone, and objects

import random
class CellPhone:

    def __init__(self,manufact,model,retail_price):
        self.manufact=manufact
        self.model=model
        self.retail_price=retail_price
        self.ID=random.randint(1,6)

    def __str__(self):
        return "Here is the data that you provided:\nManufacturer: {0}\nModel number: {1}\nRetail price: {2}\nThe ID is: {3}".format(self.manufact, self.model, self.retail_price, self.ID)



    def setID(self):
        self.ID=int(input("Change ID to: "))

    def setmanufact(self):
        self.manufact=str(input("Change manufactor to: "))

    def setmodel(self):
        self.model=str(input("Change model to: "))

    def setprice(self):
        self.retail_price=int(input("Change price to: "))


    def getID(self):
        self.ID=print("The ID is: ", self.ID)

    def getmanufact(self):
        print("Manufactor is: ", self.manufact)

    def getmodel(self):
        print("Model is: ", self.model)

    def getprice(self):
        print("Price is: ", self.retail_price)
```

main

```python
# Nico Kranni
# main.py
# Used to run phone and dice together.


from Phone import CellPhone
from dice import Dice

def main():



    new_phone0=CellPhone("Nokia","3310","150")
    new_phone1=CellPhone("Nokia","3310","150")
    new_phone2=CellPhone("Nokia","3310","150")
    new_phone3=CellPhone("Nokia","3310","150")
    new_phone4=CellPhone("Nokia","3310","150")
    new_phone5=CellPhone("Nokia","3310","150")
    phones=[new_phone0,new_phone1,new_phone2,new_phone3,new_phone4,new_phone5]



    for item in range(0,6):
        phones[item].ID=item

    for phone in phones:
        print(phone)

    new_dice = Dice()
    new_dice.roll_dice(5)



    print("\n\nDice chose: ")
    print(phones[new_dice.get_value()])




main()
```

```python
# Nico Kranni
# car.py
# class Car

class Car:
    def __init__(self, make, model, mileage, price, color, max_load, trunk):
        self._make=make
        self._model=model
        self._mileage=mileage
        self._price=price
        self._color=color
        self._max_load=max_load
        self._trunk=trunk

    def __str__(self):
        return "State of the car:\nMake: {0}\nModel: {1}\nMileage: {2}\nPrice: {3}\nColor: {4}\nMax load: {5}\nTrunk size: {6}".format(self._make,self._model,self._mileage,self._price,self._color,self._max_load,self._trunk)


    def getMake(self):
        print("Make is: ", self._make)

    def getModel(self):
        print("Model is: ", self._model)

    def getMileage(self):
        print("Mileage is: ", self._mileage)

    def getPrice(self):
        print("Price is: ", self._price)

    def getColor(self):
        print("Color is: ", self._color)

    def getMaxLoad(self):
        print("Maximum load is: ", self._max_load)

    def getTrunk(self):
        print("Trunk size is: ", self._trunk)



    def setMake(self):
        self._make=input("Change make: ")

    def setModel(self):
        self._model=input("Change model: ")
    def setMileage(self):
        self._mileage=input("Give new mileage: ")

    def setPrice(self):
        self._price=input("Set new price: ")

    def setColor(self):
        self._color=input("Give new color: ")

    def setMaxLoad(self):
        self._max_load=input("Give max load size: ")

    def setTrunk(self):
        self._trunk=input("Give trunk size: ")
```

```python
class Mammal:
    def __init__(self, ID, species, name, size, weight):
        self._ID=ID
        self._species=species
        self._name=name
        self._size=size
        self._weight=weight


    def __str__(self):
        return "State of the Mammal:\nID: {0}\nSpecies: {1}\nName: {2}\nSize: {3}\nWeight: {4}".format(self._ID,self._species,self._name,self._size,self._weight)


    def getID(self):
        print("ID is: ", self._ID)

    def getSpecies(self):
        print("Species is: ", self._species)

    def getName(self):
        print("Name is: ", self._name)

    def getSize(self):
        print("Size is: ", self._size)

    def getWeight(self):
        print("Weight is: ", self._weight)




    def setID(self):
        self._ID=input("Change ID: ")

    def setSpecies(self):
        self._species=input("Change Species: ")

    def setName(self):
        self._name=input("Give new Name: ")

    def setSize(self):
        self._size=input("Set new Size: ")

    def setWeight(self):
        self._weight=input("Give new Weight: ")
```

```python
kitty=Mammal("1","Cat","Cindy","2x5m","600g")
doggie=Mammal("2","Dog","Leevi","0.5x0.5m","1600g")
fish=Mammal("3","fish","Golden","0.1x0.1m","60g")
doggie2=Mammal("4","Dog","Pata","1x1m","1000g")
kitty2=Mammal("5","Cat","Mesi","2x5m","500g")
kitty3=Mammal("6","Cat","Bläkkis","2x5m","1500g")

print(kitty)
print(doggie)
print(fish)
print(doggie2)
print(kitty2)
print(kitty3)
```

```python
# Nico Kranni
# car_mammal.py
# Used to run multiple classes, car,dice,mammal

from mammal import Mammal
from car import Car
from dice import Dice

def main():

    auto=Car("Volvo","V80","534 miles","5000 €","Blue","600 kg","2x5m")

    kitty=Mammal("0","Cat","Cindy","10m^2","600g")
    doggie=Mammal("1","Dog","Leevi","0.25m^2","1600g")
    fish=Mammal("2","fish","Golden","0.1m^2","60g")
    doggie2=Mammal("3","Dog","Pata","1m^2","1000g")
    kitty2=Mammal("4","Cat","Mesi","10m^2","500g")
    kitty3=Mammal("5","Cat","Bläkkis","15m^2","1500g")
    mammals=[kitty,doggie,fish,doggie2,kitty2,kitty3]




    new_dice = Dice()
    new_dice.roll_dice(5)



    print("\n\nDice chose: ")
    print(mammals[new_dice.get_value()],"\n")
    if mammals[new_dice.get_value()]._size< auto._trunk:
        print (mammals[new_dice.get_value()]._name,"fits into the car!")




main()
```