Classes and Inheritence

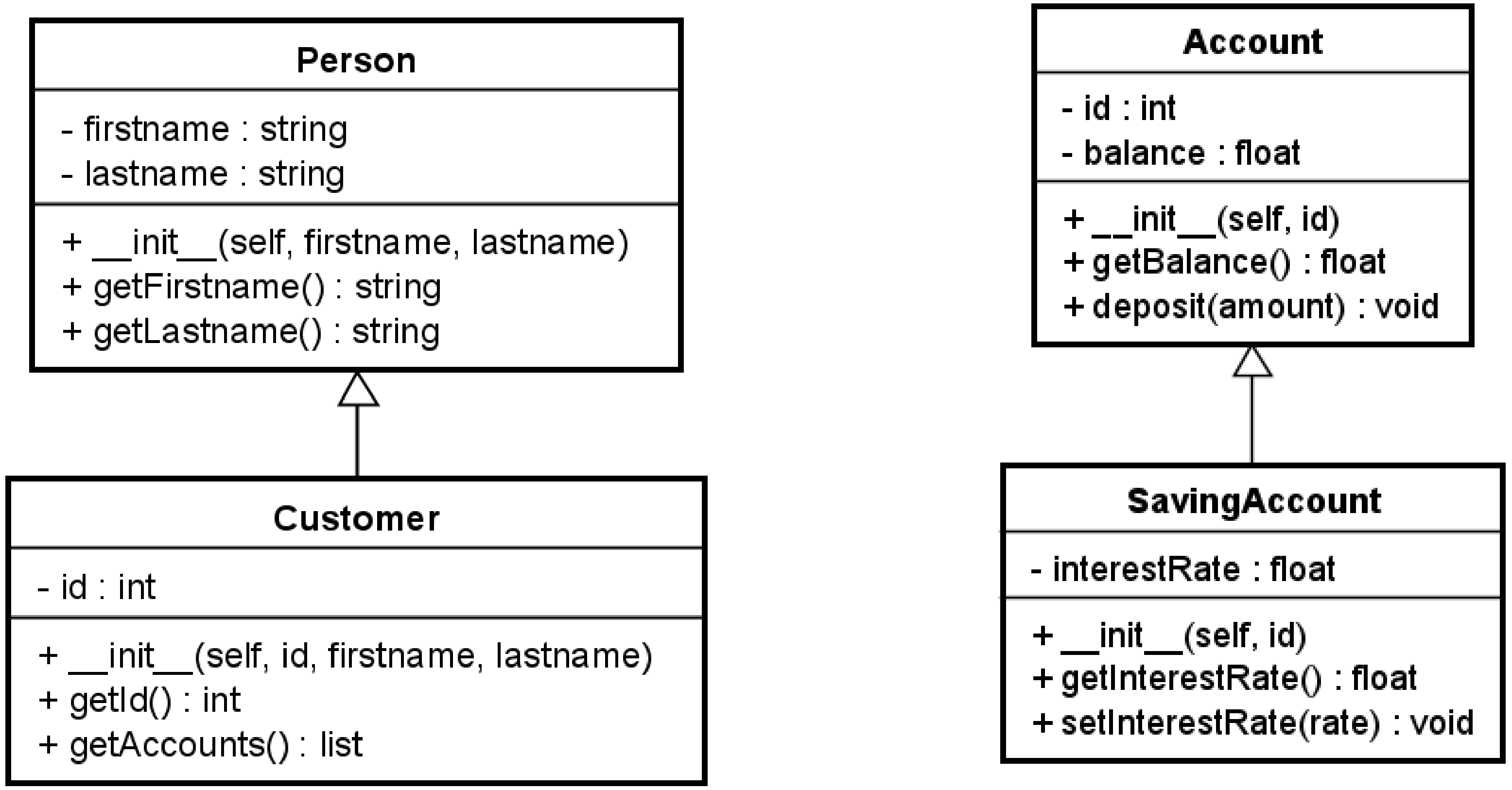
What is inheritance in OOP?

- Inheritance in OOPS
  - …is the process of passing on characteristics from one parent to a child.
  - …enables you to create classes that inherit from another class, and then enhance the list properties and methods of the child classes without affecting the parent class.
- Inheritance allows programs to create more complex structures, which can save time and effort

Berner Fachhochschule

Bank Example

- Customer class inherits from person class
- Saving account class inherits from account class



The Person class

The Person class is the base class. It defines some member variables and methods.

```
class Person:
  def __init__(self, fname, lname):
    self.__firstname = fname
    self.__lastname = lname

  def getFirstname(self):
    return self.__firstname

  def getLastname(self):
    return self.__lastname

  def __str__(self):
    return self.__firstname + " " + self.__lastname
```

The Customer class

The customer class inherits from the person class. Therefore, the member variables and methods are inherited. The constructor of the base has to be called by **super**().__init__(…).

```
# Customer of a class
class Customer(Person):
  def __init__(self, id, lastname, firstname):
    super().__init__(lastname, firstname)
    self.__id = id
    self.__accountList = list()

  def getId(self):
    return self.__id

  def getAccounts(self):
    return self.__accountList
```

The Account class

The account class is the base class of the different kind of bank accounts.

```
class Account():
  def __init__(self, id):
    self.__id = id
    self.__balance = 0

  def getId(self):
    return self.__id

  def getBalance(self):
    return self.__balance

  def deposit(self, amount):
     self.__balance = self.__balance + amount

  def __str__(self):
    return str("Account Nr:" + str(self.__id) +
        ", Balance " + str(self.__balance))
```

Berner Fachhochschule

The SavingAccount class

The SavingAccount class inherits from the Account class.

```
class SavingAccount(Account):
    def __init__(self, id, interestRate):
      super().__init__(id)
      self.__interestRate = interestRate

    def getInterestRate(self):
        return self.__interestRate

    def setInterestRate(self, rate):
        self.__interestRate = rate
```

The Bank class

The Bank class uses the customer and account classes.

```
class Bank():
  def __init__(self):
    self.__customers = dict()
    self.__accounts = dict()

  def addClient(self, client):
      self.__customers[client.getId()] = client

  def addAccount(self, client, account):
      self.__accounts[client.getId()] = account

  def getClient(self, id):
      return self.__customers.get(id)

  def getAccount(self, customer):
      return self.__accounts.get(customer.getId())

  def getAccounts(self):
      return self.__accounts
```

Usage of Bank, Customer and Account classes

```
import Customer as c
import Account as a

#new bank
bank = Bank()

#new customer peter
peter = c.Customer(17, "Peter", "Muster")
bank.addClient(peter)

bank.addAccount(peter,a.SavingAccount(10, 2))
bank.getAccount(peter).deposit(5000)

#print all account data
for key,value in bank.getAccounts().items():
    print(bank.getClient(key), " -> ", value)
```

Berner
Fachhochschule