VNIVERSITAT DE VALÈNCIA

STRUCTURES Laboratory

DATA AND ALGORITHMS

Degree in Data Science (1st)

Academic year 2023-24

Practice No. 4: Classes and Sequences
Completion period: Week of**03/15 to 03/19/2024**

Work methodology

Phases	Guy	Dedication
1. Solve the tasks set out in this scriptbefore to start the face-to-face session in the laboratory.	No presential	Max. 4.5 hrs.
2. Solve a problem in the laboratorynew exercise based on the resolution of previous tasks.	In person	Max. 3 hrs.

Introduction

The banking field is a very important field of work for data scientists and it is interesting to start getting familiar with some basic concepts. For example, the concept of a bank account.

The bank accounts of a bank's clients are entities that contain (among others) the identification data of the account holder, an account code (IBAN), an available balance (money) and a set of operations carried out. in that account (deposit or withdrawals).

The code that identifies an account has a complex structure that requires some explanation. The accounts are identified by a code that complies with the provisions of the International Bank Account Code (*International Bank Account Number*), known by the acronym IBAN. This code is an internationally agreed format for identifying cross-border bank accounts with a reduced risk of transcription errors. It was originally adopted by the European Committee for Banking Standards (ECBS), and later as an international standard under ISO 13616.
The IBAN consists of a maximum of 34 alphanumeric characters, depending on the country. In Spain the IBAN has 24 digits and is made up of 2 country digits (ES), 2 control digits and 20 digits of the complete bank account code.

IBAN example: ES9820385778983000760236

		BBAN (Basic Bank Account Code)			
Code of country ISO 3166	2 Digits of IBAN control	Code of the entity	Office of the account	2 Digits of Bank control	Number of account
IS	98	2038	5778	98	3000760236

The objective of the first part of the practice is to build a class that represents the bank accounts of a bank's customers. As always, before carrying out the laboratory session, the tasks indicated in the following exercises must be completed.

Exercises (Phase 1)

Homework 1

A class must be defined *Bank account* that represents this type of concept. A bank account must have as attributes the account identifier (an object of the provided IBAN class), the name of the account holder, and the balance/money available in the account. The values of these attributes will be set in the constructor (passed as arguments). The class will also allow you to perform the following operations with its objects:

- *GetIBAN*: Return the IBAN identifier of the account (object of the IBAN class).
- *GetTitular*: Return the name of the account holder.
- *GetBalance*: Return the balance currently available in the account (in euros).
- *Get into*: Deposit a certain amount of money into the account (in euros). Increase the balance. The argument must be a positive number. Otherwise, the operator must throw an exception.
- *Withdraw*: Withdraw a certain amount of money from the account (in euros). Reduce the balance. The argument must be a positive number. Otherwise, the operator must throw an exception.
- *str*: Which converts the content of the object to string format.

Example of using an account:

```
c = BankAccount("ES9820385778983000760236","P. Pérez",1000)
# Enter €500 into the account, balance = €1,500
c.Enter(500)
# Withdraw €300 from the account, balance = €1,200
c.Withdraw(300)
```

In Aula Virtual you can download the IBAN class (already implemented) that represents objects with the standard format and that includes operations to obtain a structured format of this code (in groups of 4 characters separated by blanks) and to know each of the elements structural code in case it is necessary for some operation.

Task 2

Add to class *Bank account* a mechanism to store the last operations performed by the client. As seen before, operations can involve depositing or withdrawing a certain amount of money into the account. In addition to the amount, each operation has an associated concept, which is nothing more than a text that describes the reason for the operation.

In Aula Virtual you can download the OperacionBancaria class (already implemented) that represents objects of this type.

What to do in class *Bank account* is:

- Add a structure **stack** to save the operations, so that the information of the account operations can be stored/referred from the most recent to the oldest.

- Modify operations *Get into* and *Withdraw* so that each of these functions generates a new *Bank transaction* with the amount entered or withdrawn, as appropriate, and its associated concept. That trade must be stored in the account's trade stack. As the banking operation has a concept associated with it, both *Get into* as *Withdraw* A new argument must be added to them that is the concept that motivates the entry or withdrawal of money. **Important**: the operation *Withdraw* must generate a *Bank transaction* of negative amount.

- Add a new operation to the class, *ShowOperations*, which will display on the screen all the operations stored in the stack (the most recent first).

Task 3

Verify with the provided test program ("test_cuentas.py") the correct functioning of the class you have created. Check that all the results obtained with the program correspond to what was expected.

Phase 2: Final task

Resolution of a new exercise proposed during the face-to-face session in the laboratory.