Programming
Academic Year: 2021/22

Practical Case

Phase 3: Storage and Equipment Rental

Content

1 Introduction

This document contains the statement of the Practical Case: Phase 3. This must be understood as a continuation of the statement for Phase 2, meaning that all information from the statement from Phase 2 is valid for this phase. Each group can modify the initial code included with this statement in order to solve all the tasks included in Section 2 of this document.

1.1 Evaluation

Phase 3 scores for **20% of the total grade** of this course.
The teacher will evaluate the deliverable and publish the grades at the end of the semester. The students can request a revision of the grade after its publication.

1.2 Delivery

The deliverable must be uploaded to Aula Global before the end of **December 15th**.

The deliverable consists of **a single compressed file** (.zip or .rar) with the following documents:

- **Report (4%)**: a report of the practical case in **PDF format**, including the sections described below. **Important:** it is not necessary to include source code in the report, although students can include some code lines in order to explain a fragment of the project itself.
  - Cover: phase of the practical case, students' name, group and course number.
  - Introduction: brief description of the work, adding any specific comments about the practical case.
  - Implementation: for each task to implement, include validation tests to verify that the program always executes correctly (including exceptional cases).
  - Conclusions: about the completeness of the solution provided, as well as comments about the practical case. Students may include personal comments about the current phase, difficulties and problems encountered during the implementation.
- **Source code (16%)**: one or more .py files with the source code that implements the practical case. The code must include program execution examples to confirm the validation tests, including exceptional cases.
  - Note: it is recommended to store all .py files in the same project folder in order to facilitate the delivery of the practical case. That project must be compressed into a .zip or .rar file and uploaded to Aula Global.

1.3 Changes with respect to Phase 2

Unlike Phase 2, in this phase it is mandatory to use data structures such as lists, sets, dictionaries when needed. It is not allowed to use external libraries that have not been teached during theory or practice classes.

2 Phase 3 Statement

Phase 3 adds functionality to the source code delivered on Phase 2, specifically the use of data structures to store information throughout the program.

2.1 Tasks to implement

The current document shows the tasks to be implemented on Phase 3 of the project, built on top of the previous deliverable. With this statement, an initial code is included with some base functionalities and functions; groups are allowed to extend/modify that code in order to implement the tasks.

The tasks to implement are the following:

1. Make the code 100% robust (capturing possible errors using exceptions)
2. Store all data in lowercase in order to avoid mistakes, except when asking for the DNI, which must be in uppercase.
3. Store all data from different lifting equipment and clients sorted as described below.
4. Allow the user to enter data for new equipment rental (option 3) and store that information using dictionaries and lists.
5. Display the information of all lifting equipment, clients and rentals as described in the following sections.

All of these tasks can be grouped into the following 4 main tasks:

2.1.1 TASK 1: Robust code (exception handling)

Update the code to avoid unexpected errors from happening during the program execution, capturing and handling all possible exceptions, specially when users enter any kind of data.
For instance, what would happen if the user enters a DNI like this: “1234a567L”?
Figure 1 shows different examples of incorrect DNI and how the errors can be handled:

```
Enter the DNI of the new client using the format 'NNNNNNNNL':
2312312iauhsds

ERROR! The first 8 characters must be numbers! Please, enter a correct
DNI.

Enter the DNI of the new client using the format 'NNNNNNNNL':
cas321fcas

ERROR! The first 8 characters must be numbers! Please, enter a correct
DNI.

Enter the DNI of the new client using the format 'NNNNNNNNL':
12345678931

ERROR! The entered DNI is not valid.

Enter the DNI of the new client using the format 'NNNNNNNNL':
987654321A
```

Figure 1. Error handling for DNI value

2.1.2 TASK 2: Storing data in lowercase

Storing all strings in uppercase or lowercase can facilitate the comparison of values in the source code and can help to avoid errors from users when entering the data.
For instance, when entering the brand of the equipment, previous phases detected an error when the data entered was not exactly the same as expected, and probably the only "error" was the use of uppercase or lowercase.
For this task, is it asked to always store all strings in lowercase, allowing the user to enter data in uppercase or lowercase.
Figure 2 shows how the user can enter the brand name POLFINGER including uppercase and lowercase letters without raising an error, storing the information in all lowercase.

```
Please, enter the brand of the lifting equipment (POLFINGER, KONEG OR
SMALZ): POLFING
Error! brand not allowed. Please, enter a valid Brand.

Please, enter the brand of the lifting equipment (POLFINGER, KONEG OR
SMALZ): POLfing
Error! brand not allowed. Please, enter a valid Brand.

Please, enter the brand of the lifting equipment (POLFINGER, KONEG OR
SMALZ): PolFinGER
Brand saved: polfinger
```

Figure 2. Example of data storage in lowercase

There's only one exception: the letter for the DNI information must be stored in uppercase following the usual format of that document number. If the user enters a lowercase letter for the DNI, it must be fixed and stored in uppercase, as shown in Figure 3:

```
Enter the DNI of the new client using the format 'NNNNNNNNL':
31734403s
DNI stored: 31734403S
```

Figure 3. Example of data storage of the DNI data

2.1.3 TASK 3: Data storage of Lifting Equipments and Clients

Implement the functionality to store multiple lifting equipment and clients, without any limitation. It must be allowed to make modifications to equipment and clients at any particular time.
The **order** of the data stored is important:

1. Lifting equipment: they must be sorted in the same order as entered by the user. Every time that the user chooses to enter a new lifting equipment, it must be shown a list of all existing equipment, as shown in Figure 4.

```
Please, enter an option (1-4): 1
## LIFTING EQUIPMENT ##
ID   BRAND     MODEL   TYPE          PRICE    PRICE+VAT   STOCK
==   =====     =====   ====          =====    =========   =====
1    polfinger m12     crane         1300.0    1508.0      10
2    koneg     m1234   lifter        1500.0    1740.0       5
```

Figure 4. Display lifting equipment

2. Clients: they must be sorted at all times based on their DNI (from lowest to highest). It is important to note that it is **not** allowed to use the method sort() or any other external sorting method, since the idea is for each group to come up with a sorting algorithm of their own.
   Every time the user enters the data of a new client, the system will check that there cannot be 2 clients with the same DNI and must ask the user for the information, as shown in Figure 5. Finally, it must display the client list in order as shown in Figure 6.
   Take into account that a DNI with the number 123456789 is higher than 8765432.

```
NEW CLIENT

Enter the first name of the new client: Pepe

Enter the last name of the new client: Fernandez

Enter the DNI of the new client using the format 'NNNNNNNNL':
6273514H
6273514H is not a valid DNI

Enter the DNI of the new client using the format
'NNNNNNNNL':123456789V
ERROR! That DNI already exists for another client. Enter another
DNI.

Enter the DNI of the new client using the format 'NNNNNNNNL':
```

Figure 5. Incorrect or duplicated DNI

```
## CLIENTS ##
DNI         NAME      LAST NAME    PHONE    ADDRESS
===         =====     ==========   ======   ========
12345678Z   john      smith        91245645   main st. 1
23456715D   mary      white        98712684   park ave. 5
34152459E   peter     parker       94125426   spidey st. 6


************ MAIN MENU ************
   3. New Lifting Equipment
   4. New client
   5. New Equipment Rental
   6. Exit

Please, enter an option (1-4):
```

Figure 6. List all clients

2.1.4 TASK 4: New Equipment Rental

Implement the functionality to allow the user to enter the data of new equipment rental. For this, the following information is requested:

1. DNI of the client that is renting the equipment: it is very important that the program verifies that the client already exists in the system, i.e. it was entered previously. If the entered DNI does not exist, the program must ask the user to enter a new DNI.
2. ID of the lifting equipment: similarly, it is mandatory to verify that the equipment exists based on its ID before it is rented. At this point, the system will not verify that this particular equipment has sufficient stock to be rented.
3. Start and end date for the rental: it is not necessary to verify that the dates are correct, but they must follow the format DD/MM/YYYY at all times, asking repeatedly for the date in case the user enters an incorrect date format.
4. Price without VAT: checking that this data is a positive floating point number.

The information of the rentals is vital for the company, therefore it is important to store it adequately in the system. It is requested to store it using a **dictionary** where the ID of the equipment is the **key**. As **value** of the dictionary, it will include a list with all the rental data for that particular equipment (DNI, start date, end date and price).

In order to facilitate the tasks to the user when including a new rental, a list of all clients' DNI and a list of all equipment IDs must be displayed, one for each line. It is recommended to implement a function that returns the list of DNIs and another function that returns the list of equipment IDs of the lifting equipment available. Figure 7 shows an execution example of this.

```
************ MAIN MENU ************
   1. New Lifting Equipment
   2. New client
   3. New Equipment Rental
   4. Exit

Please, enter an option (1-4): 3
NEW EQUIPMENT RENTAL
EQUIPMENT: [1, 2, 3]
CLIENTS: ['1231321A', '3213213212F']
```

Figure 7. Displaying all equipment and clients

Finally, to facilitate entering an adequate price for the rental, it must be displayed on screen the lowest and highest price of previous rentals made for that particular equipment, displaying a message "there haven't been previous rentals for this equipment" in case there is none. This information must be displayed right before entering the price for the rental equipment. In this case, it is recommended to implement a function that returns this information.

2.2 Important: Exclusions

Under any circumstance it is allowed to:

- use global variables or constants.
- use external libraries or objects.
- use of method sort() at any point