TD11. Exceptions, files et piles

Bases de la Programmation Impérative – Chapitre 3 : references
version du 24 novembre 2022

Les objectifs de ce TD sont les suivants :

- introduire le concept d'exception présent dans de nombreux langages impératifs ;
- continuer à manipuler des références ;
- maîtriser les piles ;
- maîtriser les files.

Exercice 1 : exceptions

Question 1

Qu'affiche le programme Java ci-dessous sachant que le point d'entrée d'un programme Java est nécessairement la fonction `public static void main(String[] args)` ?

```
1
2       public static int f(int i) {
3           System.out.println("I am f, and I am going to divide " +
4                              String.valueOf(i) +
5                              " by (i - 42) and return the result");
6           int i_minus_42 = i - 42;
7           return i / i_minus_42;
8       }
9
10      public static void g(int i) {
11          System.out.println("I am g, and because I am lazy, I am just going " +
12                             "to callf(i) and print what it returned to me");
13          int res_f = f(i);
14          System.out.println("f(" + i + ") = " + res_f);
15      }
16
17      public static void main(String[] args) {
18          g(84);
19          g(42);
20      }
```

Question 2

Qu'affiche le programme ci-dessous ?

```
1  #!/usr/bin/env python3
2
3  def f(i):
4      print("I am f, and I am going to divide "
5            + str(i)
6            + " by (i - 42) and return the result")
7      i_minus_42 = i - 42
8      if i_minus_42 == 0:
9          raise ArithmeticError("Une division par 0, mais "
10                               "pour qui vous prenez vous ?")
11     return i / i_minus_42
```

```
12
13 def g(i):
14     print("I am g, and because I am lazy, "
15           "I am just going to callf(i) and print what it returned to me")
16     try:
17         res_f = f(i)
18         print("f(" + str(i) + ") = " + str(res_f));
19     except ArithmeticError:
20         print("Oulala, f a fait des bétises")
21
22 def main():
23     g(84);
24     g(42);
25
26 if __name__ == "__main__":
27     main()
```

Exercice 2 : les derniers seront les premiers

Vous êtes contactés par l'association *NiEmacsNiVim* pour réaliser un petit projet logiciel pour eux. Cette association développe un éditeur de texte révolutionnaire dans lequel il **suffit** de taper `contrôle + s` pour sauvegarder un fichier et de taper `contrôle + z` pour annuler la dernière action. Afin justement de pouvoir annuler un grand nombre d'actions, *NiEmacsNiVim* nous demande de leur fournir un module Python permettant de :

- créer un historique d'actions initialement vide ;
- ajouter une action à un historique ;
- récupérer la dernière action ajoutée en levant une exception si il n'y a plus d'action dans l'historique ;
- connaître la taille de l'historique.

Question 1

Ce cahier des charges correspond il à une structure de données (SDD) ou à un type abstrait (TA) ?

Question 2

Avez vous déjà rencontré quelque chose de similaire, et si oui sous quelle dénomination ?

Question 3

Proposez un squelette de votre module pour qu'il puisse être validé par *NiEmacsNiVim* avant que vous ne commenciez le développement. Vous fournirez une nouvelle classe permettant de représenter une pile. Pour rappel, dans le cadre de BPI nous utilisons les classes uniquement comme un agrégat de données. Les opérations que vous fournirez seront donc des fonctions et non pas des méthodes.

Question 4

Implémentez votre module sans utiliser de `list` Python.

Question 5

Dessinez la mémoire d'une pile contenant du haut vers le bas `"partez"`, `3.0`, `(1, 2)`. C'est à dire une pile dans laquelle nous avons empilé `(1, 2)` puis `3.0` puis `"partez"`.

Question 6

Justifiez vos choix auprès de *NiEmacsNiVim*.

Question 7

Sans écrire de code, comment feriez-vous pour implémenter votre module avec une `list` Python ? Quels seraient les avantages et les inconvénients par rapport à votre implémentation précédente ?

Exercice 3 : première arrivée, première servie

Question 1

Analysez le module Python ci-dessous et notez toutes les remarques (positives et négatives) et questions que vous pouvez avoir.

```python
#!/usr/bin/env python3

"""Implémentation de l'ADT file en python."""

import datetime
import time

def arrive(l, b):
    l.append(b)

def premier(l):
    assert l, "file vide!"
    preums = l[0]
    l.pop(0)
    return preums

# Quelques tests
file_devant_la_poste = []
if datetime.date.today().weekday() == 5: # Samedi, c'est blindé !
    for i in range(1, 43):
        arrive(file_devant_la_poste, "client" + str(i))
else:
    arrive(file_devant_la_poste, "Micheline")
    arrive(file_devant_la_poste, "Karim")
    arrive(file_devant_la_poste, "Bao")
print(file_devant_la_poste)
try:
    while True:
        print(premier(file_devant_la_poste) + " au guichet SVP")
        time.sleep(5)
except:
    pass
print(file_devant_la_poste)
```

Exercice 4 : il y a des piles et des files partout (pour aller plus loin)

Question 1

Listez toutes les utilisations de le TA pile au sein d'un ordinateur que vous avez déjà rencontrées.

Question 2

Même question pour le TA file.

Question 3

Faites un dessin/schéma pour illustrer le plus simplement et clairement possible selon vous les notions de pile et de file. L'objectif est d'expliquer ces concepts à vos collègues absents aujourd'hui.