le cnam

Année universitaire 2019-2018

Sujet UE DIU EIL:

Examen première session : 10/2019

Responsable : P. Courtieu – C. Picouleau – P. Rigaux

Durée : 3h

Consignes

Tous les documents sont autorisés.

Sujet de 4 pages, celle-ci comprise.

→ *Vérifiez que vous disposez de la totalité des pages du sujet en début d'épreuve et signalez tout problème de reprographie le cas échéant.*

Algorithmique

Exercice 1 manipulation de tas

Soit T un tas maximum comportant au moins $n = 3$ éléments distincts.

Question 1.1
Donner un algorithme de complexité $O(1)$ qui retourne la valeur du troisième plus grand élément.

Programmation objet

Utilisation d'objets (répondre sur papier)

Exercice 2 Utilisation d'objet

On suppose écrites les classes `Recette` et `Ingrédient`, dont on vous donne les en-têtes de méthodes (notez que `Recette` correspond plutôt à la liste de courses pour une recette) :

```
class Ingredient:
    # nom est une string et prixUnitaire est un float
    def __init__(self, nom, prixUnitaire):
        ...
    # Accesseurs
    def getNom(self):
        ...

    def getPrixUnitaire(self):
        ...

class Recette:
    # titre est une string
    def __init__(self,titre):
        ...
    # ingredient est un Ingredient et quantité un int
    def ajouter(self,ingredient, quantite):
        ...
    # ingredient est un Ingredient
    def retirer(self,ingredient):
        ...

    # retourne le prix total de la recette (un float)
    def getPrixTotal(self):
        ...

    # retourne la liste des ingrédients
    def getListeIngredients(self):
        ...

    # retourne la quantité de de l'Ingredient ingredient nécessaire à la recette
    def getQuantiteIngredient(self, ingredient):
        ...
```

Notez que les types attendus en argument et les types de retour des fonctions sont précisés en commentaire.

Question 2.1

Écrire un programme principal qui va :
- — créer un ingrédient « oignon », de prix « 1 euro »,
- — créer un ingrédient « carotte », de prix « 2 euros » ,
- — créer une recette de titre « carotte à l'oignon », qui contiendra trois carottes et deux oignons ;
- — afficher le prix total de la recette.

Écriture de classe (répondre sur machine)

Exercice 3

Dans une course à pieds, on suppose que chaque coureur est identifié par son numéro de dossard. On crée un objet de type `Course` en passant la liste des noms de coureurs, rangés dans l'ordre des numéros de dossards (le premier dans la liste correspond au dossard 1, le second au dossard 2, etc.)

Quand un joueur franchit la ligne d'arrivée, on l'enregistre en appelant la méthode `arriver()`.

Enfin à tout moment la méthode `afficher` affiche la liste des coureurs arrivés dans l'ordre d'arrivée. Voici un exemple de scenario d'utilisation de la classe :

Un scenario Le programme principal suivant :

```
# Liste initiale composée de 4 coureur
l = ["Dupont", "Durand", "Turing", "Church"]
# On crée la course avec ces 4 coureurs
c = Course(l)
c.arriver(3) # le dossard 3 (Turing) arrive en premier
c.arriver(1) # le dossard 1 (Dupont) arrive en deuxième
c.afficherResultat() # affichage du résultat provisoire de la course
```

devrait afficher :

```
1 - Turing (dossard 3)
2 - Dupont (dossard 1)
```

On propose un début d'implémentation pour représenter les résultats d'une course :

```
class Course:
    # initialise la liste des coureurs avec listNomsCoureurs,
    # et la liste des arrivées à la liste vide.
    def __init__(self, listNomsCoureurs):
        # liste de noms de coureur (string) dans l'ordre des dossards
        # self.nomsCoureur[0] correspond au dossard no 1.
        self.nomsCoureur = listNomsCoureurs

        # Une liste d'entiers représentant l'ordre d'arrivée des
        # dossards seuls les dossards arrivés apparaissent dans cette
        # liste. Attention ce sont bien les numéros de dossards qui
        # sont stockés, pas l'indice du coureur dans self.nomsCoureur
        self.ordreArrivee = []

    def getNom(self, numeroDossard):
        ...
    def arriver(self, numeroDossard):
        ...
    def afficherResultat(self):
        ...
```

Dans cette implémentation :

- `nomsCoureurs` contient les noms des coureurs, donnés dans leur ordre de dossard. Attention il y a un décalage : `nomsCoureur[0]` correspond au dossard no 1 ;
- `ordreArrivee` permet d'enregistrer l'ordre d'arrivée des coureurs. Par exemple la case 0 de ce tableau contiendra le numéro de dossard du gagnant. Les appels à `ordreArrivee` sont nécessairement faits dans l'ordre d'arrivée.
- `nombreCoureursArrives` contient le nombre de coureurs qui sont déjà arrivés. Au départ, il n'y en a aucun.

Question 3.1

Écrire la méthode `arriver(numoDossard)` qui enregistre l'arrivée du coureur de dossard `numeroDossard`.

Question 3.2

Écrire la méthode afficherResultats, qui affiche les résultats sous la forme :

```
1 - toto (dossard 4)
2 - titi (dossard 2)
...
```

Rappels :

- pour concaténer 2 listes `l1` et `l2` on fait `l1 + l2` ;
- pour concaténer deux chaînes de caractère `s1` et `s2` on fait `s1 + s2` ;
- la longueur d'une liste `l` s'obtient avec `len(l)`.

Bases de données

Exercice donné oralement.