420-2N6-RO

**EXAMEN Formatif**

**Programmation 2   
Profil réseautique**

%

Durée : 2h00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nom :** |  | |  | **Note:** |
|  |  | |  | **/ 50** |
|  |  |  |  |  |

**ATTENTION !** Il y Cinq (5) questions dans cet examen FORMATIF, chaque question est sur une page différente.

Vous avez le droit aux notes de cours qui vous sont fournies.

## Création d'une classe simple /5

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

À partir du diagramme UML, compléter la classe **Employe** dans le fichier **Q1\_classe\_simple.py**

La classe possède trois attributs publics.

Le constructeur prend 3 paramètres, toutefois le **temps\_plein** est optionnel et sa valeur par défaut est **True**.

La méthode **calculer\_salaire()** retourne un float, correspondant à 2000 \* **taux** si la variable **temps\_plein** est **True** et 1200 \* **taux** sinon. La valeur retournée ne contient que 2 chiffres après la virgule, vous pouvez utilisé la fonction **round().**

Quand la classe est terminée, l’exécution du script devrait donnée le résultat suivant dans le terminal :

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

## Création d'une classe un peu plus complexe /15

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

À partir du diagramme UML, complétez la classe dans le fichier Q2\_definition\_de\_classe.py

Cette classe comprend 4 attributs et 2 propriétés.

1. Le constructeur prend 4 paramètres en plus du self ; le nom, le prenom, l’id, et la date de naissance
2. Le **\_nom,** le **\_prenom**, et le **\_id** sont des attributs privés
3. La **date\_de\_naissance** est un objet de type datetime
4. La propriété **nom** retourne le nom complet de la personne. Donc le self.\_prenom suivi d’un espace suivi du self.\_nom
5. La propriété id retourne le self.\_id et possède un setter.
   1. Ce setter va convertir la valeur reçue en paramètre en un int avant de changer la valeur de self.\_id

Exécuter le script donne le résultat suivant :



## Création d'une classe parent /15

Vous avez déjà les classes **Cercle** et **Rectangle** dans le fichier Q3\_heritage.py, ces deux classes sont très similaires.

Vous devez compléter deux classes supplémentaires : **Forme** et **Carré**.

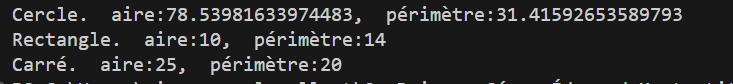
Diagramme de classes souhaité :

Une image contenant texte, capture d’écran, conception

Description générée automatiquement

1. Vous devez compléter la classe **Forme** et modifier les classes **Cercle** et **Rectangle** afin qu'ils héritent de la classe **Forme**.
   1. **Forme** est une classe abstraite
   2. La classe **Forme** possède deux méthodes abstraites : **aire()** et **périmètre()**
2. Vous devez compléter la classe **Carré**.
   1. Elle hérite de **Rectangle**
   2. Le constructeur de la classe **Carré** ne prend qu’une variable **longueur** en paramètre en plus du **self**.
   3. Le constructeur de la classe **Carré** fait appel au constructeur du parent et lui passe la valeur de **longueur** reçu comme paramètre pour la **longueur** et la **hauteur** du Rectangle.

Exécuter le script donne le résultat suivant :



## Relation de classe /20

Le fichier Q4\_relation.py contient deux classes que vous devez compléter.

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

1. Compléter la classe **Patient**. **(5 points)**  
   Elle ne comporte qu’un constructeur prenant 3 paramètres : une valeur d’id, de nom, et de numéro d’assurance social (NAS).
   1. Le paramètre **id** est obligatoire
   2. Les paramètres **nom** et **NAS** sont optionnels avec une valeur par défaut de « Inconnu »
   3. La **date\_admission** a la valeur None lors de l’instanciation.
2. Compléter la classe **Departement. (15 points)**
   1. Elle comporte les attributs suivants :
      1. **liste\_departement** est une variable de classe
      2. **nom** est un attribut public
      3. **\_liste\_patients** est un attribut privé qui consiste en une liste vide lors de l’instanciation
   2. Le constructeur prend 1 seul paramètre : le nom du département.
      1. La **\_liste\_patients** devient une liste vide lors de l’instanciation
      2. Le self est ajouté à la **liste\_departements**, une liste contenant toutes les instances de la classe **Departement**
   3. **liste\_patients** est une propriété qui retourne \_**liste\_patients**.
   4. **ajouter\_patient()** est une méthode d’instance qui prend en paramètre un objet de la classe **Patient**.
      1. Elle change la valeur de **date\_admission** du patient passé en paramètre pour qu’elle ait la valeur d’aujourd’hui (vous pouvez utiliser date.today() )
      2. Elle ajoute le patient passé en paramètre à la variable d’instance **\_liste\_patients**
   5. **imprimer\_liste\_des\_patients()** est une méthode de classe. Elle comporte deux boucles imbriquées
      1. Une première boucle qui imprimer le nom de chaque département dans **liste\_departements**.
      2. Une deuxième boucle à l’intérieur de la précédente qui imprime pour chaque patient dans la **liste\_patients** de ce département : son **id**, son **nom**, et sa **date\_admission**.

Lorsque le script terminé et exécuté, l’appel de la méthode **imprimer\_liste\_des\_patients()** devra afficher les lignes suivantes dans le terminal :

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

## Levé d’erreurs /15

Le script Q5\_levé\_erreur.py est un début de système de gestion d’un chenil (élevage de chiens).

Il contient une classe très simple appelé **Chien**.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Il y a malheureusement beaucoup d’erreurs de frappe lorsque les employés ajoutent des chiens au système, on veut donc ajouter une série de vérifications au constructeur afin de réduire les erreurs.

Vous devez ajouter des conditionnels qui vont soulever des erreurs si les valeurs passées en paramètres ne sont pas bonnes.

Vous devez :

1. Soulevez une erreur de type **ValueError** avec un message « Age trop élevé » si la valeur de pAge est supérieure à 20.
2. Soulevez une erreur de type **ValueError** avec un message « Nom trop court » si la longueur de la chaine pNom passée en paramètre est de moins de 3 charactères
3. Soulevez une erreur de type **TypeError** si pNom OU pRace n’est pas de type **str**.   
   Vous pouvez utiliser la fonction **type()** pour obtenir le type d’une variable ou bien la fonction **isinstance()** pour obtenir une booléenne indiquant si une variable fait partie d’une classe.
4. Créez une nouvelle classe appelée **ValeurNégative** qui hérite de **ValueError**
5. Soulever une erreur de type **ValeurNégative** si le paramètre pAge est inférieur à 0.

FIN !

Enregistrez ce fichier Word et remettez-le au professeur selon ses directives.