Asie– 2024 – sujet1 - Correction

Exercice 3 (8 points)

Partie A – Programmation orientée objet

1. Création de `personneA`

```
personneA = Personne(112, "LESIEUR", "Isabelle", 1982, 2005)
```

👉*Commentaire : On respecte l'ordre des paramètres du constructeur : numéro, nom, prénom, année de naissance, année d'entrée.*

---

2. Accéder au numéro de badge

```
personneA.num_badge
```

👉*Commentaire : On accède directement à l'attribut de l'objet.*

---

3. Méthode `annee_anciennete`

```
def annee_anciennete(self):
    return 2024 - self.annee_entree
```

👉*Commentaire : L'ancienneté correspond à la différence entre l'année courante et l'année d'entrée.*

---

4. Méthode `ajouter`

```
def ajouter(self, personne):
    self.liste.append(personne)
```

👉*Commentaire : On ajoute l'objet `Personne` à la liste du personnel.*

---

5. Méthode `effectif`

```
def effectif(self):
    return len(self.liste)
```

👉 *Commentaire : Le nombre d'employés correspond à la taille de la liste.*

---

6. Méthode `donne_nom`

```
def donne_nom(self, num):
    for elt in self.liste:
        if elt.num_badge == num:
            return elt.nom
    return None
```

👉 *Commentaire : On parcourt la liste et on compare les numéros de badge. Si aucun ne correspond, on renvoie* `None`.

---

7. Méthode `nb_personne_honneur`

```
def nb_personne_honneur(self, annee):
    compteur = 0
    for elt in self.liste:
        if annee - elt.annee_entree == 10:
            compteur += 1
    return compteur
```

👉 *Commentaire : On compte les personnes ayant exactement 10 ans d'ancienneté à l'année donnée.*

---

8. Méthode `plus_anciens`

```
def plus_anciens(self, annee):
    max_anciennete = 0
    for elt in self.liste:
        anciennete = annee - elt.annee_entree
        if anciennete > max_anciennete:
            max_anciennete = anciennete

    liste_badges = []
    for elt in self.liste:
        if annee - elt.annee_entree == max_anciennete:
            liste_badges.append(elt.num_badge)

    return liste_badges
```

👉*Commentaire **:** On cherche d'abord l'ancienneté maximale, puis on récupère tous les badges correspondants.*

Partie B – Base de données relationnelle

9. Requête SQL

```
SELECT nom, prenom FROM Personnel
WHERE num_centre = 2;
```

👉*Commentaire : La requête affiche le nom et le prénom des personnes travaillant dans le centre numéro 2.*

---

10. Mutation de HADJI Hakim

```
UPDATE Personnel
SET num_centre = 3
WHERE num_badge = 135;
```

👉*Commentaire : On modifie le centre uniquement pour la personne ayant le badge 135.*

---

11. Intérêt de deux tables

Utiliser deux tables permet :

- d'éviter les redondances,
- de réduire la taille de la base,
- de garantir la cohérence des données,
- de respecter le principe de normalisation.

---

12. Relation entre les tables

Les tables sont reliées par l'attribut `num_centre`.

- `Personnel.num_centre` est une **clé étrangère**
- `Centre.num` est la **clé primaire**

13. Requête demandée (centre de Lille)

```
SELECT nom
FROM Personnel
JOIN Centre ON Personnel.num_centre = Centre.num
WHERE Centre.ville = 'Lille'
AND annee_debut BETWEEN 2015 AND 2020;
```

👉*Commentaire :*

- *On utilise une jointure pour relier les deux tables.*
- `BETWEEN` *inclut les bornes.*

---

14. Erreur de la requête DELETE

Requête donnée :

```
DELETE *
FROM Centre
WHERE nom = 'Normandie';
```

L'erreur vient de `DELETE *`.
La syntaxe correcte est :

```
DELETE FROM Centre
WHERE nom = 'Normandie';
```

👉*Commentaire : En SQL, on ne met pas* `*` *avec DELETE.*