TRAVAIL PRATIQUE 01 (15%)

Le Zoo

Vous allez développer, dans cet exercice, un programme Java pour gérer les informations d'un **Zoo**. Vous devrez suivre le nombre de visiteurs quotidiens, la liste des animaux avec leurs poids, ainsi que des statistiques associées au zoo. Le programme devra inclure un **menu interactif** qui permet à l'utilisateur de naviguer entre différentes fonctionnalités.

Le programme doit permettre de :

- Gérer une liste d'**animaux** avec leur nom, espèce, et poids.
- Enregistrer le **nombre de visiteurs** par jour.
- Calculer des **statistiques** telles que le poids moyen des animaux et le nombre total de visiteurs.
- Trouver les **animaux les plus lourds et les plus légers**.
- Rechercher un animal par son nom.
- Afficher un résumé complet du zoo incluant les visiteurs et les animaux.

PARTIE 0 : créer le menu principal

La première étape consiste à créer un menu principal qui permet à l'utilisateur de choisir entre différentes options. Vous devez implémenter un menu qui reste actif jusqu'à ce que l'utilisateur choisisse de quitter le programme. Le menu doit contenir les options suivantes :

1. Ajouter un nouvel animal
2. Afficher la liste des animaux et leurs informations
3. Vérifier si un animal est dans le zoo
4. Modifier le poids d'un animal
5. Ajouter des visiteurs pour une journée
6. Afficher le nombre total de visiteurs
7. Afficher le poids moyen des animaux
8. Trouver l'animal le plus lourd et le plus léger
9. Afficher un résumé complet du zoo (visiteurs et animaux)
10. Quitter le programme

Objectif : Le menu doit permettre à l'utilisateur de choisir une option en saisissant un numéro, et le programme doit exécuter l'action correspondante.

Important : Le menu doit gérer les **entrées invalides** de l'utilisateur. Si l'utilisateur entre une option qui n'est pas disponible, le programme doit afficher un message d'erreur approprié et redemander une entrée valide.

PARTIE 1 : AJOUTER UN NOUVEL ANIMAL

Créez une méthode nommée ajouterNouvelAnimal() qui permet à l'utilisateur d'**ajouter un nouvel animal**. Chaque animal aura les informations suivantes :

- Le **nom** de l'animal (par exemple, « simba »).
- Le **type** ou l'**espèce** de l'animal (par exemple, "Lion").
- Le **poids** en kilogrammes (par exemple, 200,00 kg).

Vous devez stocker ces informations correctement afin de pouvoir y accéder ultérieurement.

PARTIE 2 : AFFICHER LA LISTE DES ANIMAUX ET LEURS INFORMATIONS

Créez une méthode nommée `afficherListeAnimaux()` qui permet d'**afficher tous les animaux** avec leurs informations (nom, espèce, et poids). Utilisez la bonne structure conditionnelle pour afficher chaque animal avec ses données respectives.

PARTIE 3 : RECHERCHER UN ANIMAL PAR SON NOM

Créez une méthode nommée `rechercherAnimal()` qui permet à l'utilisateur de **rechercher un animal en entrant soit son nom, soit son espèce**. Le programme doit parcourir les tableaux et, si un animal correspond au **nom** ou à l'**espèce** saisis, afficher ses informations (**nom**, **espèce**, **poids**).

- Si plusieurs animaux appartiennent à la même espèce, affichez tous les animaux de cette espèce avec leurs informations (nom, espèce, poids).
- Si aucun animal ne correspond à la recherche, affichez un message indiquant que l'animal ou l'espèce n'a pas été trouvé(e).

PARTIE 4 : MODIFIER LE POIDS D'UN ANIMAL

Créez une méthode nommée `modifierPoidsAnimal()` qui permet de **modifier le poids** d'un animal existant. L'utilisateur entre le nom de l'animal à modifier et la nouvelle valeur du poids. Vérifiez d'abord si l'animal existe dans le tableau et offrez ensuite la possibilité de mettre à jour son poids.

PARTIE 5 : AJOUTER DES VISITEURS POUR UNE JOURNÉE

Créez une méthode nommée `ajouterVisiteursParAnimal()` qui permet d'enregistrer et de suivre le **nombre de visiteurs** qui viennent voir chaque animal dans le zoo. À chaque visite, vous devrez ajouter le nombre de visiteurs associés à un animal donné, en vous assurant de conserver le cumul des visites déjà enregistrées. L'utilisateur pourra ainsi ajouter des visiteurs pour chaque animal autant de fois que nécessaire.

Instructions détaillées :

1. **Stockage des visiteurs par animal** :
   Créez un tableau où chaque élément représente le **nombre total de visiteurs** pour un animal particulier. Le tableau aura la même taille que le tableau des animaux et permettra de suivre le cumul des visiteurs pour chaque animal.
2. **Ajouter des visiteurs pour un animal** :
   La méthode doit permettre à l'utilisateur de sélectionner un animal par son **nom** et d'ajouter des visiteurs à ce dernier. Si l'animal est trouvé, le nombre de visiteurs est ajouté au total déjà enregistré pour cet animal.
3. **Cumul des visiteurs** :
   À chaque appel de la méthode, l'utilisateur peut ajouter des visiteurs pour un animal, et cette valeur s'ajoutera au total existant. Ainsi, chaque fois que des visiteurs viennent pour un animal donné, le cumul est mis à jour.
4. **Afficher le total des visiteurs** :
   Après chaque mise à jour, affichez le **nombre total de visiteurs** pour l'animal sélectionné.

PARTIE 6 : CALCULER ET AFFICHER LE NOMBRE TOTAL DE VISITEURS

Créez une méthode nommée `calculerTotalVisiteursZoo()` qui permet de **calculer le nombre total de visiteurs** ayant visité tous les animaux du zoo. Pour cela, vous devez additionner le nombre de visiteurs pour chaque animal, et ensuite afficher le total cumulé.

PARTIE 7 : CALCULER ET AFFICHER LE POIDS MOYEN DES ANIMAUX

Créez une méthode nommée `calculerPoidsMoyenAnimaux()` qui calcule et affiche le **poids moyen** des animaux présents dans le zoo. Vous devrez déterminer la valeur moyenne en fonction des poids enregistrés pour chaque animal et afficher le résultat final.

PARTIE 8 : TROUVER L'ANIMAL LE PLUS LOURD ET LE PLUS LÉGER

Créez une méthode nommée `trouverAnimauxExtremes()` qui permet de **trouver l'animal le plus lourd** et l'**animal le plus léger**.

PARTIE 9 : AFFICHER UN RÉSUMÉ COMPLET DU ZOO

Créez une méthode nommée `afficherResumeCompletZoo()` qui affiche un **résumé complet du zoo**, y compris :

- Le **nombre total de visiteurs**.
- La **liste des animaux**, avec leurs noms, espèces, et poids.
- Le **poids moyen** des animaux.
- L'animal le plus lourd et le plus léger.

PARTIE 10 : QUITTER LE PROGRAMME

Le programme doit continuer à afficher le menu et permettre des actions jusqu'à ce que l'utilisateur sélectionne l'option **Quitter**. Utilisez une structure adéquate pour gérer les choix de l'utilisateur dans le menu.

GRILLE DE CORRECTION

CRITÈRES D'ÉVALUATION	DESCRIPTION	POINTS
1. Fonctionnalité et complétion (9 points)	Le programme compile correctement sans erreurs de syntaxe (exécution sans bugs).	
a) L'ajout d'un nouvel animal	La méthode `ajouterNouvelAnimal()` est correctement implémentée et fonctionne comme prévu.	/ 2
b) L'affichage de la liste des animaux	La méthode `afficherListeAnimaux()` affiche correctement tous les animaux avec leurs informations.	/ 0.5
c) La recherche d'un animal par nom/espèce	La méthode `rechercherAnimal()` permet de rechercher un animal par nom ou espèce et affiche tous les animaux correspondants.	/ 1
d) La modification du poids d'un animal	La méthode `modifierPoidsAnimal()` permet de modifier le poids d'un animal existant correctement.	/ 1
e) L'ajout de visiteurs	La méthode `ajouterVisiteursParAnimal()` permet d'ajouter des visiteurs pour un animal de manière cumulative.	/ 1
f) Le calcul du total des visiteurs	La méthode `calculerTotalVisiteursZoo()` calcule et affiche correctement le nombre total de visiteurs du zoo.	/ 0.5
g) Le calcul du poids moyen des animaux	La méthode `calculerPoidsMoyenAnimaux()` calcule et affiche correctement le poids moyen des animaux du zoo.	/ 0.5
h) La recherche de l'animal le plus lourd et le plus léger	La méthode `trouverAnimauxExtremes()` permet de trouver l'animal le plus lourd et l'animal le plus léger.	/ 1
i) Le résumé complet du zoo	La méthode `afficherResumeCompletZoo()` affiche un résumé complet avec les visiteurs et les informations sur les animaux.	/ 1.5
2. Le choix des structures de données et algorithmes	Les boucles et conditions sont bien choisies et efficaces pour les opérations demandées.	/ 2
3. La qualité et propreté du code (3 points)		

a) Lisibilité et indentation	Le code est bien indenté, facile à lire et structuré de manière claire.	/ 0.5
b) L'utilisation de noms de variables significatifs	Les noms des variables, méthodes, et classe sont clairs, significatifs et respectent les conventions.	/ 0.5
d) La documentation (commentaires explicatifs)	Les principales sections du code sont correctement commentées pour expliquer le fonctionnement.	/ 1
4. La gestion des erreurs et validation (1.5 points)		
a) Validation des entrées utilisateur	Le programme vérifie les entrées de l'utilisateur et gère les erreurs ou entrées invalides.	/ 1
b) Gestion des cas limites	Les cas particuliers (zoo vide, animal non trouvé, etc.) sont correctement pris en compte.	/ 0.5
5. Originalité et créativité (0.5 point)		
a) Améliorations ou fonctionnalités supplémentaires	L'étudiant a ajouté des fonctionnalités pertinentes non demandées ou amélioré l'interface utilisateur.	/0.5