TP : Gestion d’un Zoo

Implémenter un système de gestion d'animaux de zoo illustrant les concepts d'héritage, d'abstraction, les méthodes equals/hashCode/toString, et les mécanismes d'upcasting/downcasting en Java.

Classes à implémenter

1. Classe abstraite Animal

Attributs:

- `id` (int): identifiant unique de l'animal
- `nom` (String): nom de l'animal
- `age` (int): âge de l'animal en années
- `poids` (double): poids de l'animal en kg

Méthodes:

- Constructeur avec tous les attributs
- Getters et setters pour tous les attributs
- `calculerNourriture()` : méthode abstraite qui calcule et retourne la quantité de nourriture quotidienne en kg
- `faireUnSon()` : méthode abstraite qui retourne une chaîne de caractères représentant le son de l'animal
- `toString()` : redéfinition qui retourne une chaîne au format "Animal [id=X, nom=X, age=X, poids=X]"
- `equals(Object obj)` : redéfinition qui compare deux animaux (deux animaux sont égaux s'ils ont le même id)
- `hashCode()` : redéfinition qui retourne un code de hachage basé sur l'id

2. Classe Carnivore (hérite d'Animal)

Attributs supplémentaires:

- `quantiteViande` (double): quantité de viande consommée par jour en kg
- `estDangereux` (boolean): indique si l'animal est dangereux

Méthodes supplémentaires:

- Constructeur avec tous les attributs (y compris ceux hérités)
- Getters et setters pour les nouveaux attributs
- Redéfinition de `calculerNourriture()` : implémentation qui retourne quantiteViande * 1.5

- Redéfinition de `faireUnSon()` : implémentation qui retourne un son spécifique aux carnivores (ex: "Grrr")
- Redéfinition de `toString()` : format "Carnivore [id=X, nom=X, age=X, poids=X, quantiteViande=X, estDangereux=X]"
- `chasser()` : méthode qui retourne un message décrivant comment l'animal chasse

3. Classe Herbivore (hérite d'Animal)

Attributs supplémentaires:

- `quantiteFoin` (double): quantité de foin consommée par jour en kg
- `estDomestique` (boolean): indique si l'animal est domestique

Méthodes supplémentaires:

- Constructeur avec tous les attributs (y compris ceux hérités)
- Getters et setters pour les nouveaux attributs
- Redéfinition de `calculerNourriture()` : implémentation qui retourne quantiteFoin * 1.2
- Redéfinition de `faireUnSon()` : implémentation qui retourne un son spécifique aux herbivores (ex: "Meuh" ou "Bêê")
- Redéfinition de `toString()` : format "Herbivore [id=X, nom=X, age=X, poids=X, quantiteFoin=X, estDomestique=X]"
- `brouter()` : méthode qui retourne un message décrivant comment l'animal broute

4. Classe GestionZoo

Attributs:

- `animaux` (Animal[]): tableau des animaux du zoo
- `nombreAnimaux` (int): nombre actuel d'animaux dans le tableau

Méthodes:

- `GestionZoo(int capacite)` : constructeur qui initialise un tableau vide avec la capacité spécifiée
- `boolean ajouterAnimal(Animal animal)` : ajoute l'animal au tableau et retourne true si réussi, false si le tableau est plein ou si un animal avec le même id existe déjà
- `void afficherAnimaux()` : parcourt le tableau et affiche tous les animaux en utilisant toString()
- `double calculerCoutTotal()` : calcule et retourne la somme des coûts de nourriture de tous les animaux
- `void traiterAnimal(Animal animal)` : méthode qui:
    - Affiche les informations générales de l'animal

    - Vérifie si l'animal est un Carnivore ou un Herbivore avec instanceof
    - Effectue un downcasting approprié pour appeler chasser() ou brouter()
    - Inclut un cas par défaut si l'animal n'est ni l'un ni l'autre
- `boolean peuventCohabiter(Animal a1, Animal a2)`: méthode qui:
    - Retourne false si un des animaux est un Carnivore dangereux et l'autre un Herbivore
    - Retourne false si les deux animaux sont des Carnivores dangereux
    - Retourne true dans les autres cas
    - Utilise instanceof pour vérifier les types
- `Animal trouverAnimalParId(int id)`: parcourt le tableau et retourne l'animal avec l'id spécifié, null si non trouvé
- `void compterTypeAnimaux()`: compte et affiche le nombre de carnivores et d'herbivores dans le zoo

5. Classe Main (Programme principal)

Méthodes:

- `public static void main(String[] args)`: méthode principale qui:
    - Crée une instance de GestionZoo avec une capacité définie
    - Crée plusieurs instances de Carnivore et Herbivore avec différents attributs
    - Les ajoute au zoo
    - Teste toutes les méthodes de GestionZoo
    - Démontre le fonctionnement des méthodes equals() et toString()
    - Montre un exemple d'upcasting (assignation d'un Carnivore ou Herbivore à une variable de type Animal)
    - Montre un exemple de downcasting (conversion d'une référence Animal vers Carnivore ou Herbivore)