KHEMIR ISSRA
3A45

Prosit 2

Instruction 5 :

```
public class Animal {

    private String family;
    private String name;
    private int age;
    private boolean isMammal;


    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getFamily() {
        return family;
    }

    public void setFamily(String family) {
        this.family = family;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public boolean isMammal() {
        return isMammal;
    }

    public void setMammal(boolean mammal) {
        isMammal = mammal;
    }
}
```

```
public class Zoo {
    private Animal[] animals;
    private String name;
    private String city;
    private int nbrCages;
```

```
    public Animal[] getAnimals() {
        return animals;
    }

    public void setAnimals(Animal[] animals) {
        this.animals = animals;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public int getNbrCages() {
        return nbrCages;
    }

    public void setNbrCages(int nbrCages) {
        this.nbrCages = nbrCages;
    }

}
```

```
import java.util.Scanner;
public class ZooManagement {
    private int nbrCages;
    private String zooName;
    public ZooManagement() {}

    public void setNbrCages(int nbrCages) {
        this.nbrCages = nbrCages;
    }

    public void setZooName(String zooName) {
        this.zooName = zooName;
    }

    public static void main(String[] args) {

        Animal lion = new Animal();
        lion.setFamily("Felidae");
        lion.setName("Lion");
        lion.setAge(3);
```

```
        lion.setMammal(true);

        Zoo myZoo = new Zoo();
        myZoo.setName("Mon Zoo");
        myZoo.setCity("Tunis");
        myZoo.setNbrCages(25);

        System.out.println("Animal : " + lion.getName() + ", Age : " +
lion.getAge());
        System.out.println("Zoo : " + myZoo.getName() + ", Ville : " +
myZoo.getCity());

    }
}
```

```
C:\Users\dell\.jdks\jbr-17.0.8.1\bin\java.exe "-
Animal : Lion, Age : 3
Zoo : Mon Zoo, Ville : Tunis
```

Instruction 6 :

La principale différence réside dans l'utilisation de constructeurs pour créer et initialiser les objets de manière plus concise et directe dans la deuxième version de la méthode main. Cela rend le code plus lisible et facilite l'initialisation des objets avec des valeurs cohérentes dès le début.

Instruction 7:

```
public static void main(String[] args) {
Animal lion = new Animal("Felidae","Lion",3,true);
Animal elephant = new Animal("Elephantidae", "Elephant", 8, true);
Animal giraffe = new Animal("Giraffidae", "Giraffe", 5, false);

Animal[] animals = new Animal[25];
animals[0] = lion;
animals[1] = elephant;
animals[2] = giraffe;

Zoo myZoo = new Zoo("Mon Zoo", "Tunis", 10, animals);

}
```

Instruction 8 :

Dans la classe Zoo:

```
public void displayZoo() {
    System.out.println("Nom du zoo : " + name);
    System.out.println("Ville du zoo : " + city);
    System.out.println("Nombre de cages : " + nbrCages);
}
```

Dans la classe Main :

```
System.out.println("displayZoo : ");
myZoo.displayZoo();
System.out.println("myZoo : ");
System.out.println(myZoo);
System.out.println("myZoo.toString : ");
System.out.println(myZoo.toString());
```

```
C:\Users\dell\.jdks\jbr-17.0.8.1\bin\java.exe
displayZoo :
Nom du zoo : Mon Zoo
Ville du zoo : Tunis
Nombre de cages : 10
myZoo :
Zoo@27f674d
myZoo.toString :
Zoo@27f674d
```

Sans redéfinir la méthode toString(), l'affichage par défaut est une représentation de l'objet qui comprend le nom de la classe Zoo et la valeur hexadécimale qui identifie cet objet dans la mémoire.

Instruction 9 :

Pour obtenir une sortie lisible et significative, il faut redéfinir la méthode toString() dans la classe Zoo.

```
@Override
public String toString() {
    return "Nom du zoo : " + name + "\nVille du zoo : " + city + "\nNombre de
cages : " + nbrCages;
}
```

```
C:\Users\dell\.jdks\jbr-17.0.8.1\bin\java.exe "
displayZoo :
Nom du zoo : Mon Zoo
Ville du zoo : Tunis
Nombre de cages : 10
myZoo :
Nom du zoo : Mon Zoo
Ville du zoo : Tunis
Nombre de cages : 10
myZoo.toString :
Nom du zoo : Mon Zoo
Ville du zoo : Tunis
Nombre de cages : 10
```

Même pour la classe Animal:

```
@Override
public String toString() {
    return "Famille : " + family + "\nNom : " + name + "\nÂge : " + age +
"\nMammifère : " + isMammal;
}
```

Instruction 10:

```
public class Zoo {
    private Animal[] animals;
    private String name;
    private String city;
    private int nbrCages;
    private int animalCount;


    public Zoo(String name, String city, int nbrCages ) {
        this.name = name;
        this.city = city;
        this.nbrCages = nbrCages;
        this.animals = new Animal[nbrCages];
        this.animalCount = 0;
    }
```

```
    public boolean addAnimal(Animal animal) {
        if (animalCount < nbrCages) {
            animals[animalCount] = animal;
            animalCount++;
            return true;
        } else {
            return false;
        }
    }
```

```
public static void main(String[] args) {
    Animal lion = new Animal("Felidae","Lion",3,true);
    Animal elephant = new Animal("Elephantidae", "Elephant", 8, true);
    Animal giraffe = new Animal("Giraffidae", "Giraffe", 5, false);

    Zoo myZoo = new Zoo("Mon Zoo", "Tunis",2);
    boolean ajout1 = myZoo.addAnimal(lion);
    boolean ajout2 = myZoo.addAnimal(elephant);
    boolean ajout3 = myZoo.addAnimal(giraffe);
    System.out.println("Ajout 1 : " + ajout1);
    System.out.println("Ajout 2 : " + ajout2);
    System.out.println("Ajout 3 : " + ajout3);
}
```

```
C:\Users\dell\.jdks\jbr-17.0.8.1\bin\java.exe '
Ajout 1 : true
Ajout 2 : true
Ajout 3 : false
```

Dans cet exemple, le zoo a une capacité de 2 cages. Les trois premiers ajouts (ajout1et ajout2) sont réussis, car ils ne dépassent pas la capacité maximale. Le quatrième ajout (ajout3) échoue car le zoo est plein, et la méthode addAnimal retourne false. Cela montre que la méthode fonctionne comme prévu pour empêcher les ajouts lorsque la capacité maximale est atteinte.

```
public void displayAnimals() {
    System.out.println("Animaux dans le zoo :");
    for (int i = 0; i < nbrCages; i++) {
        if (animals[i] != null) {
            System.out.println("Cage " + (i + 1) + ": " + animals[i]);
        } else {
            System.out.println("Cage " + (i + 1) + ": Vide");
```

```
        }
    }
}
public int searchAnimal(Animal animal) {
    for (int i = 0; i < nbrCages; i++) {
        if (animals[i] != null &&
animals[i].getName().equals(animal.getName())) {
            return i;
        }
    }
    return -1;
}
```

```
public static void main(String[] args) {
    Animal lion = new Animal("Felidae","Lion",3,true);
    Animal elephant = new Animal("Elephantidae", "Elephant", 8, true);
    Animal giraffe = new Animal("Giraffidae", "Giraffe", 5, false);

    Zoo myZoo = new Zoo("Mon Zoo", "Tunis", 3);
    boolean ajout1 = myZoo.addAnimal(lion);
    boolean ajout2 = myZoo.addAnimal(elephant);
    boolean ajout3 = myZoo.addAnimal(giraffe);
    myZoo.displayAnimals();
    int indice = myZoo.searchAnimal(lion);
    System.out.println("Indice de l'animal trouvé : " + indice);
}
```

```
C:\Users\dell\.jdks\jbr-17.0.8.1\bin\java.exe "
Animaux dans le zoo :
Cage 1: Famille : Felidae
Nom : Lion
Âge : 3
Mammifère : true
Cage 2: Famille : Elephantidae
Nom : Elephant
Âge : 8
Mammifère : true
Cage 3: Famille : Giraffidae
Nom : Giraffe
Âge : 5
Mammifère : false
Indice de l'animal trouvé : 0
```

KHEMIR ISSRA
3A45

Instruction 11:

```
public static void main(String[] args) {
    Zoo myZoo = new Zoo("Mon Zoo", "Tunis", 3);

    Animal lion1 = new Animal("Felidae", "Lion", 3, true);
    myZoo.addAnimal(lion1);

    Animal lion2 = new Animal("Felidae", "Lion", 3, true);
    myZoo.addAnimal(lion2);

    int indice1 = myZoo.searchAnimal(lion1);
    int indice2 = myZoo.searchAnimal(lion2);

    System.out.println("Indice du 1 lion trouvé : " + indice1);
    System.out.println("Indice du 2 lion trouvé : " + indice2);
}
```

```
Indice du 1 lion trouvé : 0
Indice du 2 lion trouvé : 0
```

Ce que l'on remarque dans le code précédent, c'est que la méthode searchAnimal trouve le premier animal (lion1) avec le même nom qu'elle rencontre.

Les deux variables indice1 et indice2 ont la même valeur (0) car elles pointent vers le même animal dans le zoo. Cela montre que la méthode de recherche fonctionne en comparant les noms des animaux.

Donc la méthode searchAnimal renverra l'indice du premier animal trouvé avec ce nom. Elle ne fait pas la distinction entre les animaux identiques en fonction de leur position dans le tableau ou de leur référence mémoire, mais plutôt en fonction de leur nom.

Instruction 12:

```
public boolean addAnimal(Animal animal) {
    for (int i = 0; i < animalCount; i++) {
        if (animals[i].getName().equals(animal.getName())) {
            return false;
        }
    }
    if (animalCount < nbrCages) {
        animals[animalCount] = animal;
        animalCount++;
        return true;
    } else {
        return false;
    }
}
```

KHEMIR ISSRA
3A45

Instruction 13:

```
public boolean removeAnimal(Animal animal) {
    for (int i = 0; i < animalCount; i++) {
        if (animals[i] != null && animals[i].equals(animal)) {
            for (int j = i; j < animalCount - 1; j++) {
                animals[j] = animals[j + 1];
            }
            animals[animalCount - 1] = null;
            animalCount--;
            return true;
        }
    }
    return false;
}
```