Module M15

La programmation orientée objet JAVA

Chapitre 5

Les classes abstraites et les interfaces

Plan du cours

1. Héritage (rappel)
2. Les classes abstraites
3. Les interfaces
4. Classe abstraite ou interface
5. Classe abstraite vs interface

Héritage

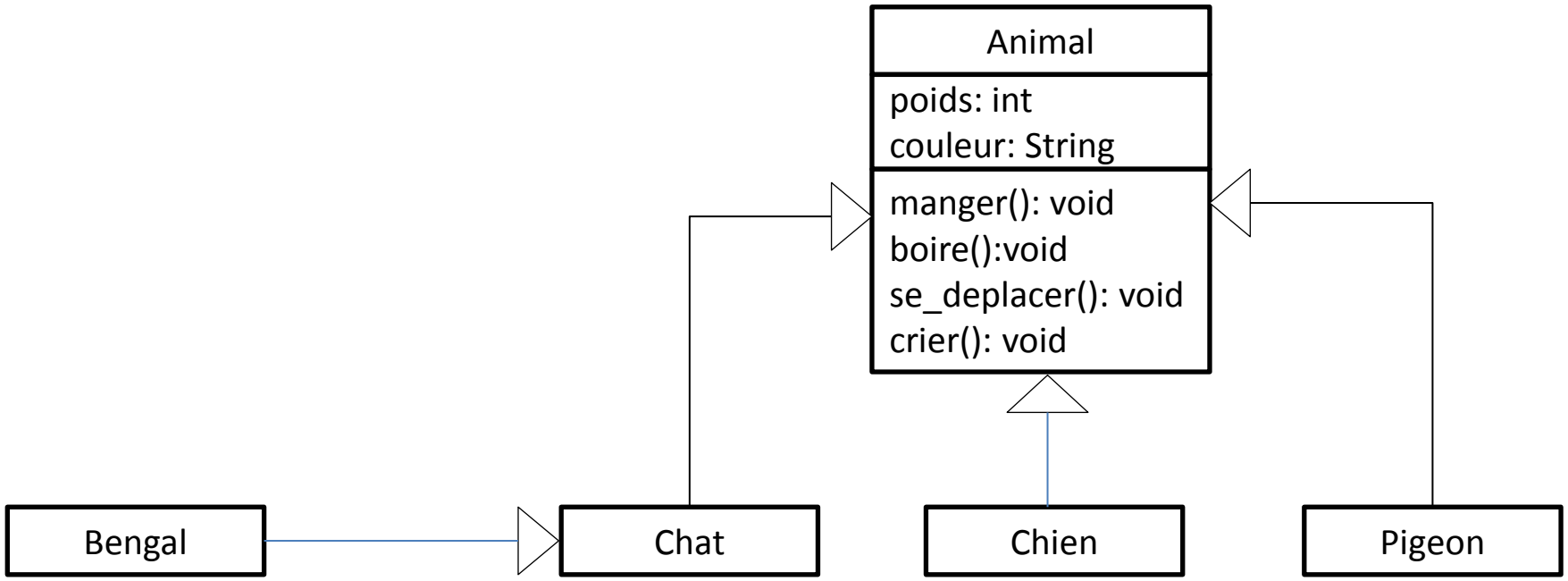
Rappel

- L'héritage est l'un des fondements de la programmation orientée objets
- À partir d'une superclasse (classe mère), nous pouvons créer autant de classe filles (classe dérivées).

Héritage

Exemple d'utilisation

- **Gestion des différents types d'animaux**



Implémenter les classes et donner un exemple d'utilisation

Héritage

Exemple d'utilisation

Superclasse instanciable

```
public class Animal{
int poids;
String couleur;
public  void manger(){
System.out.println("je mange de xxx");
}
public  void boire(){
System.out.println("je bois de l'eau ");
}
public  void crier(){
System.out.println("je  xxx");
}
public  void se_deplacer(){
System.out.println("je  xxx");
}
public void identite(){
System.out.println("poids:"+poids + "Couleur:"+couleur);
}
}
```

```
public class chien extends Animal{
}
```

```
public class chat extends Animal{
}
```

```
public class Pigeon extends Animal{
}
```

```
public class Bengal extends Chat{
}
```

```
public class Test {
Animal animal=new Animal();
animal.manger();
animal.crier();
}
```

L'objet crée ne représente pas une réalité

Héritage

Exemple d'utilisation- constats

- Un système gère les objets concrets
- Souvent, la superclasse n'a pas un "sens" ou elle n'est pas liée à des choses réelles
- Dans des cas, la superclasse ne doit pas donner lieu à des objets concrêts : elle ne doit pas avoir la possibilité d'être instanciée

Les classes abstraites

Classes abstraites

Définition

- Une classe abstraite est identique à une classe normale
- Elle est déclarée en utilisant le terme abstract:

```
public abstract class nomClasseAbstraite{
    ....
    }
```

- Elle caractérise un concept générique
- Elle ne peut pas être instanciée
- Elle contient au moins une méthode abstraite
  - La classe fille (non abstraite) doit implémenter la méthode abstraite.

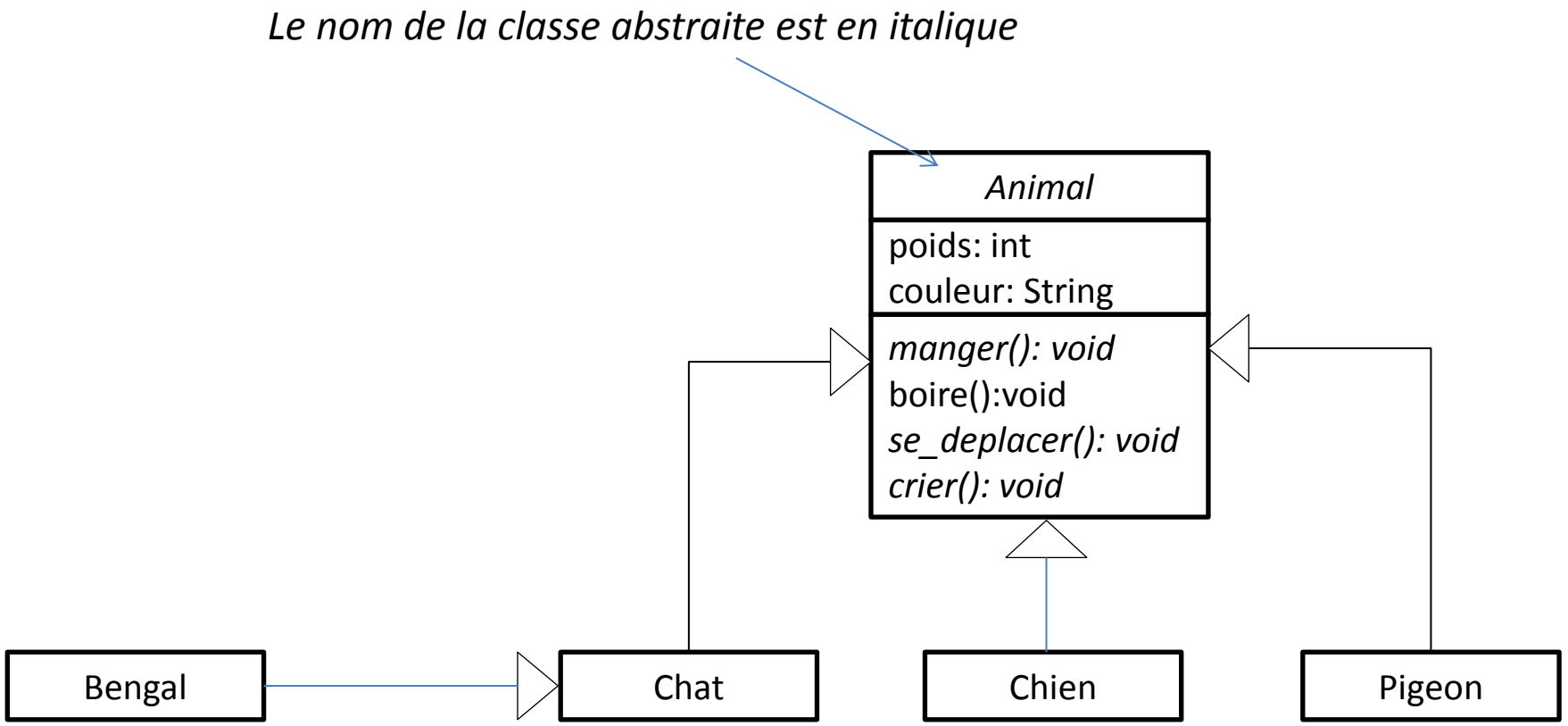
Classes abstraites

Une méthode abstraite

- **Une méthode abstraite ne contient que la signature.**
- **Une méthode abstraite ne peut exister que dans une classe abstraite**
- **Une classe qui contient une méthode abstraite doit être abstraite**

Classes abstraites

Représentation en UML



Les classes abstraites

Le code java

```
public classe NomClasseAbstraite{
public  abstract type1 methode1(liste param1);
Public type2 methode2(liste_param2){
[…]
}

}
```

```
public class  nomClasse extends NomClasseAbstraite{
public  type1 methode1(liste param1){
[……]
}
}
```

Héritage

Exemple d'utilisation

Superclasse non instanciable

```
public abstract  class Animal{
int poids;
String couleur;
public  void manger();
 public  void boire(){
System.out.println("je bois de l'eau ");
}
public  void crier();
public  void se_deplacer();
public void identite(){
System.out.println("poids:"+poids + "Couleur:"+couleur);
}
}
```

Classes abstraites

Exemple d'utilisation

```
public class Bengal extends Chat{
public void manger() {
System.out.println("je mange la viande hachée");
}
}
```

```
public class Pigeon extends animal{
public  void manger(){
System.out.println("je mange des grains");
}
public  void crier(){
System.out.println("je roucoule");
}
public  void se_deplacer(){
System.out.println("je  volle");
}
}
```

```
public class Chat extends animal{
public  void manger(){
System.out.println("je mange de la viande");
}
public  void boire(){
System.out.println("je bois de l'eau et du lait");
}
public  void crier(){
System.out.println("je miaule: miaw miaw");
}
public  void se_deplacer(){
System.out.println("je  marche sur quatre pattes!");
}
}
```

```
public class Chien extends animal{
Public  void manger(){
System.out.println("je mange de la viande");
}
public  void boire(){
System.out.println("je bois de l'eau et du lait");
}
public  void crier(){
System.out.println("j'aboie: haw haw");
}
public  void se_deplacer(){
System.out.println("je  marche sur quatre pattes!");
}
}
```

Classes abstraites

Exemple d'utilisation

```
public class Test{
public  static void main(String arg[]){
Animal animal1=new Animal();// instruction refusée (classe abstraite)
Animal  chat1=new Chat(); //instruction acceptée (classe concrète)
Chat chat2=new Chat();
Chien chien=new Chien();
}
}
```

Les classes abstraites

- **Faites attention de ne pas trop utiliser les classes abstraites**
    - **Pour ne pas compliquer le modèle**

Les classes abstraites

- **Une classe fille d'une classe abstraite qui ne définit pas toutes les méthodes de sa classe mère doit être abstraite et ne peut pas être instanciée.**
- **Il est possible de définir une classe abstarct sans contenir aucune méthode abstraite**

Les classes abstraites

Exercice d'application

- **Les formes géométriques**
- **Gestion de stock??**

Les interfaces

Définition

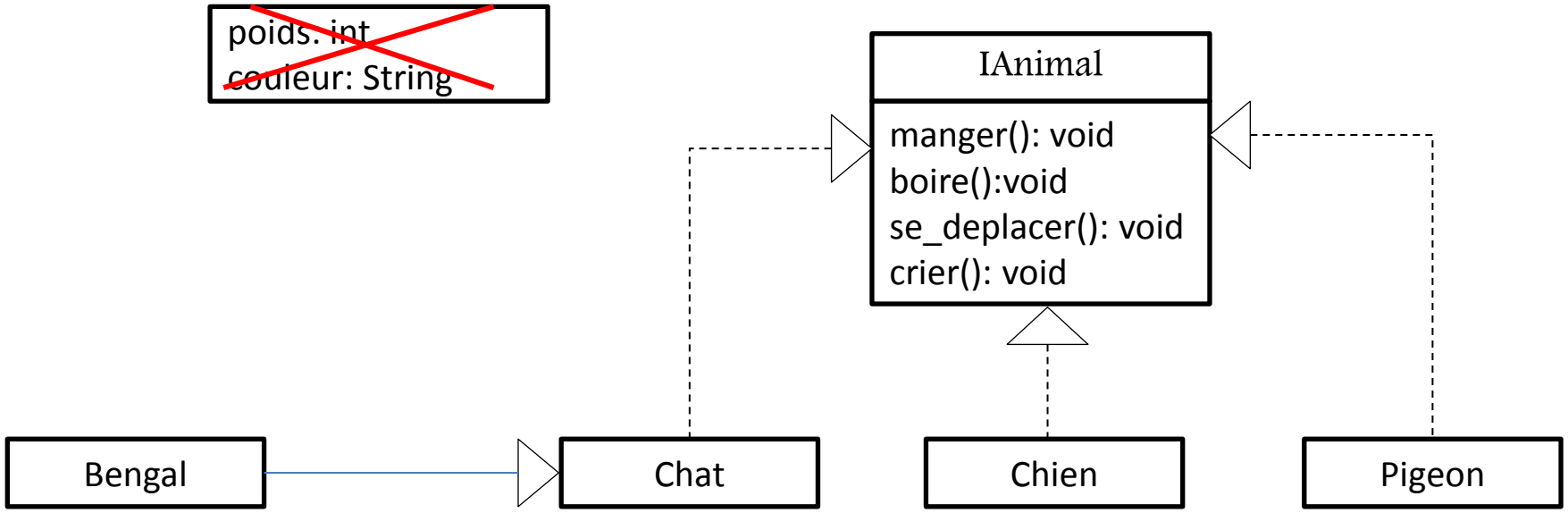
- **Une interface est similaire à une classe abstraite avec les exceptions suivantes:**
  - **Toutes les méthodes d'une interface sont abstraites**
  - **Elle ne peut pas contenir des attributs d'instances, cependant elle peut contenir des variables de type** *public static final*

- **Une interface est définie en utilisant le mot clé interface au lieu de class**
- **Les interfaces sont implémentés par des classes à l'aide du mot clé *implements***
- **Une classe qui implémente une interface doit implémenter toutes les méthodes de cette interface.**

Classes abstraites

Représentation graphique (exemple 1)

Le poids et la couleur ne sont pas des membres statiques



Les interfaces

Implémentation

- **Une classe peut hériter d'une seule classe. Cependant, une classe peut implémenter plusieurs interfaces séparées par des virgules.**
- **Chaque classe implémentant une interface doit fournir une implémentation à toutes les méthodes de cette interface**
- **Si une classe implémente plusiuers interfaces elle doit implémenter toutes les méthodes de toutes les interfaces**

Les interfaces

Implémentation

- **Note:**
  - **Si une classe abstraite implèmente une interface, elle n'a pas besoin d'implémenter toutes les méthodes de cette interface. Cependant, chaque sous classe concrète de la classe abstraite elle doit implémenter toutes les méthodes abstraites non implémentées par leur classe mère.**
- **Les interfaces peuvent hériter la signature des méthodes à partir d'une ou plusieurs autres interface**

Les interfaces

Le code java

Modificateur public ou absent (public par défaut)

```
public interface InomInterface{
public  [abstract]type1 methode1(liste param1);
Public [abstract] type2 methode2(liste_param2);

}
```

Interface

```
public class  nomClasse implements InomInterface{
public  type1 methode1(liste param1){
[……]
}
public  type2 methode2(liste param2){
[……]
}
}
```

Implémentation de l'interface

Les interfaces

Le code java de l'exemple 1

```
public interface IAnimal {
  //attributs statiques s'il y en a
[…]
//les méthodes abstraites
 public abstract void manger();
   public void se_deplacer();
   public void crier();
   public void dormir();
}
```

```
public class Chien implements IAnimal {
public void manger() {
System.out.println("je mange la viande ! et pas celle des souris");
}
public void crier() {
System.out.println("j'aboie");
}
public void se_deplacer(){
System.out.println("je marche!");
}
public void dormir() {
//il est possible de ne rien mettre dans le corps de la fonction,
mais  au moins il faut la définir !!!
}
}
```

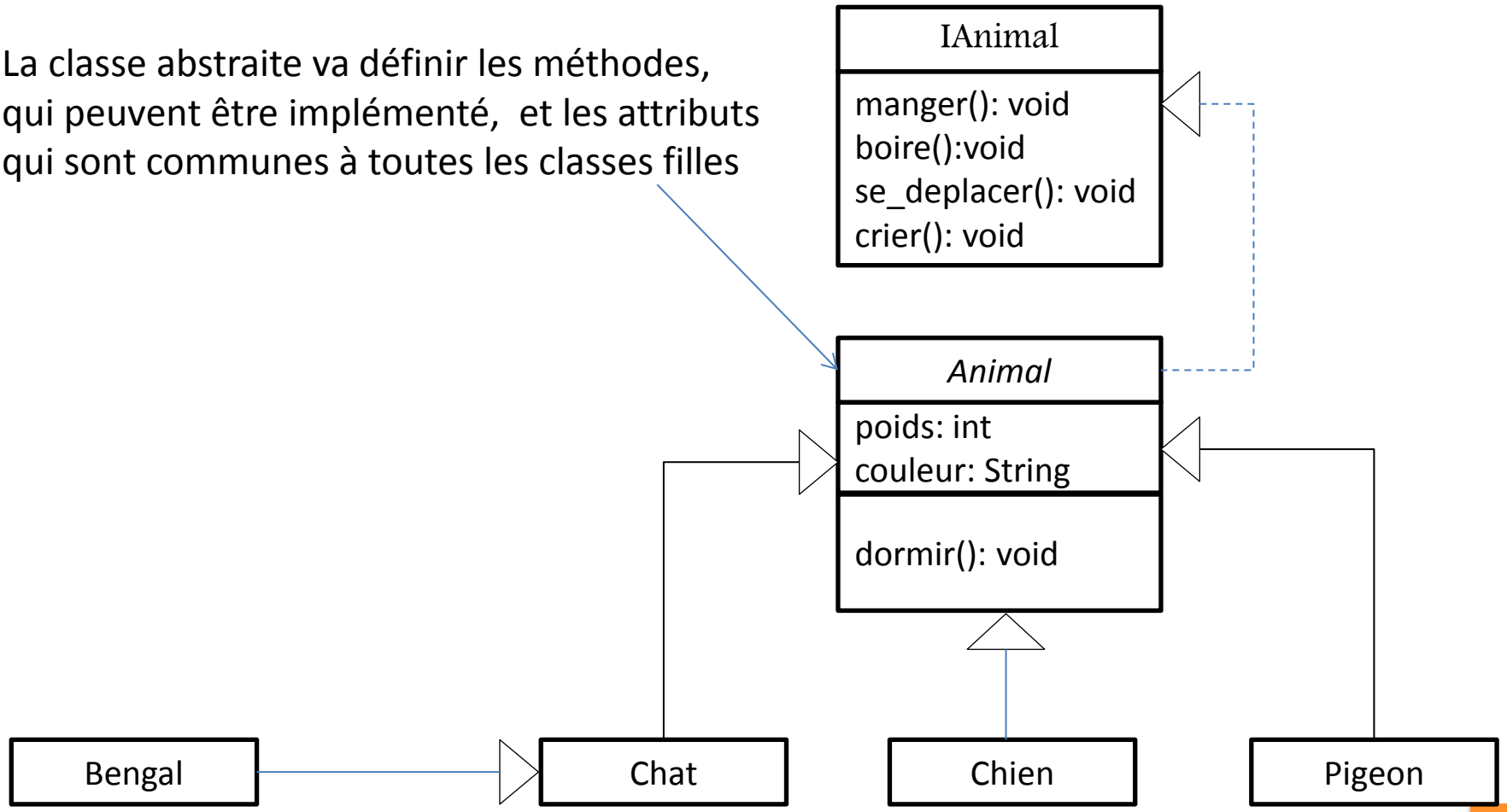
[la même chose pour le reste des classes:
- Chat
- Pigeon

]

Classes abstraites

Exemple d'utilisation (exemple 2)

La classe abstraite va définir les méthodes, qui peuvent être implémenté, et les attributs qui sont communes à toutes les classes filles



Les interfaces

Le code java de l'exemple 2

```
public interface IAnimal {
  //variables  de type final s'il y en a
[…]
//les méthodes abstraites
 public abstract void manger();
   public void se_deplacer();
   public void crier();
   public void dormir();
}
```

```
public abstract class Animal implements IAnimal{
private String couleur;
private int poids;
public abstract void manger();
public abstract void crier();
public void se_deplacer(){
System.out.println("je me deplace");
}
public Animal(String couleur,int poids){
this.couleur=couleur;
this.poids=poids;
}
public String identite(){
return this.getClass()+ " "+ couleur + " "+ poids+ "KG";
}
}
```

Classe abstraite qui implémente l'interface

```
public class Chien extends Animal{
public Chien(String couleur, int poids) {
super(couleur, poids);
}
public void manger() {
System.out.println("je mange la viande ! et pas celle des
souris");
}
public void crier() {
System.out.println("j'aboie");
}
public void se_deplacer(){
System.out.println("je marche!");
}
public void dormir() {
}
}
```

Classes abstraites vs Interfaces

- **Quand utiliser une classe abstraite au lieu d'une interface?**
  - **Si la relation sousclasse – superclasse est véritablement une relation de type "est une"**
  - **Si la classe abstraite peut fournir une implémentation au niveau approprié d'abstraction**

- **Quand utiliser une interface au lieu d'une classe abstraite?**
  - **Lorsque les méthodes définies représentent une petite partie d'une classe**
  - **Lorsque la sous-classe doit hériter d'une autre classe !!!**
  - **Lorsque vous ne pouvez pas raisonnablement implémenter l'une des méthodes**
  - **Lorsqu'il ya besoin de séparer complètement spécification / comportement de l'implémentation**

Différences entre classe abstraite et interfaces

- Les méthodes d'une inteface sont implicitement abstract et ne peuvent pas être implémentés (dans l'interface). Tandis que les méthodes d'une classe abstraite peuvent être implémentés (dans la classe abstraite) et définissent le comportement par défaut.

- Les variables déclarées dans une interface sont par défaut public static final. Tandis que une classe abstraite peut définir des attributs d'instance avec les type de portée usuels (private, protected...).

Différences entre classe abstraite et interfaces

- Les interfaces sont implémentées à l'aide du mot clé implements. Tandis que les classes abstraites sont étendues à l'aide du mot clé extends.

- Une interface java peut **étendre plusieurs interfaces**. tandis que une classe abstraite peut étendre une classe et implémente plusieurs interfaces

Différences entre classe abstraite et interfaces

- Une classe java peut étendre une seule classe abstraite. Mais elle peut implémenter plusieurs interfaces.

- Une interface ne peut pas être instanciée. De même une classe abstraite, elle ne peut pas être instanciée mais elle peut être invoquée si elle contient un main().

Discussions

Classes vs Interfaces

instanceof

```
if (a4 instanceof Chien)
System.out.println("traitement 1");
else
System.out.println("traitement 2");
```