Classes Abstraites et Interfaces

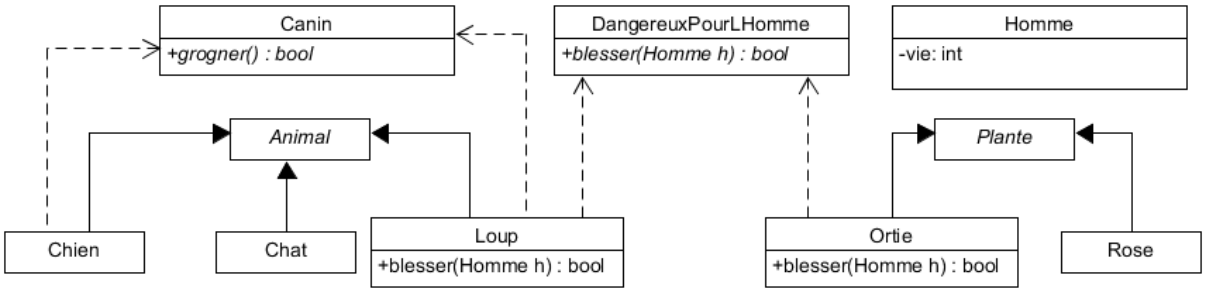
INFO-H200

Les classes abstraites s'utilisent pour faire de l'héritage, et l'héritage s'utilise quand un objet **est un** autre objet.

Par exemple : un chien est un animal, une personne est un humain, une rose est une plante. De ce fait, l'objet fils hérite des méthodes de son objet père.

Les interfaces servent à regrouper les objets par comportement et à obliger des objets à avoir certaines méthodes, on dit qu'un objet **a un** comportement de *interface*.

Par exemple : un loup a un comportement DangereuxPourLHomme, une ortie a un comportement DangereuxPourLHomme, Un chien et un loup ont un comportement Canin.



Cela a beaucoup d'utilités, mais en particulier on peut en trouver deux :

Listes : Une interface nous permet de regrouper des objets dans des listes en fonction de leur comportement et pas en fonction de leur classe mère.

```
ArrayList<Animal> animaux;
animaux.add(new Chien());
animaux.add(new Chat());
animaux.add(new Loup());

ArrayList<Plante> plantes;
plantes.add(new Rose()) ;
plantes.add(new Ortie()) ;

ArrayList<DangereuxPourLHomme> dangereux;
dangereux.add(new Ortie());
dangereux.add(new Loup());

Homme h = new Homme();
for (DangereuxPourLHomme element : dangereux) {
  element.blesser(h);
}
```

Fonctions : Des fonctions qui prennent en paramètre un objet par son comportement et pas par son type.

```
void pousse_un_cri(Canin c) {
  c.grogner();
}

pousse_un_cri(new Loup());
pousse_un_cri(new Chien());
```