

# Programmation objet

Héritage  
Interface

# Interface

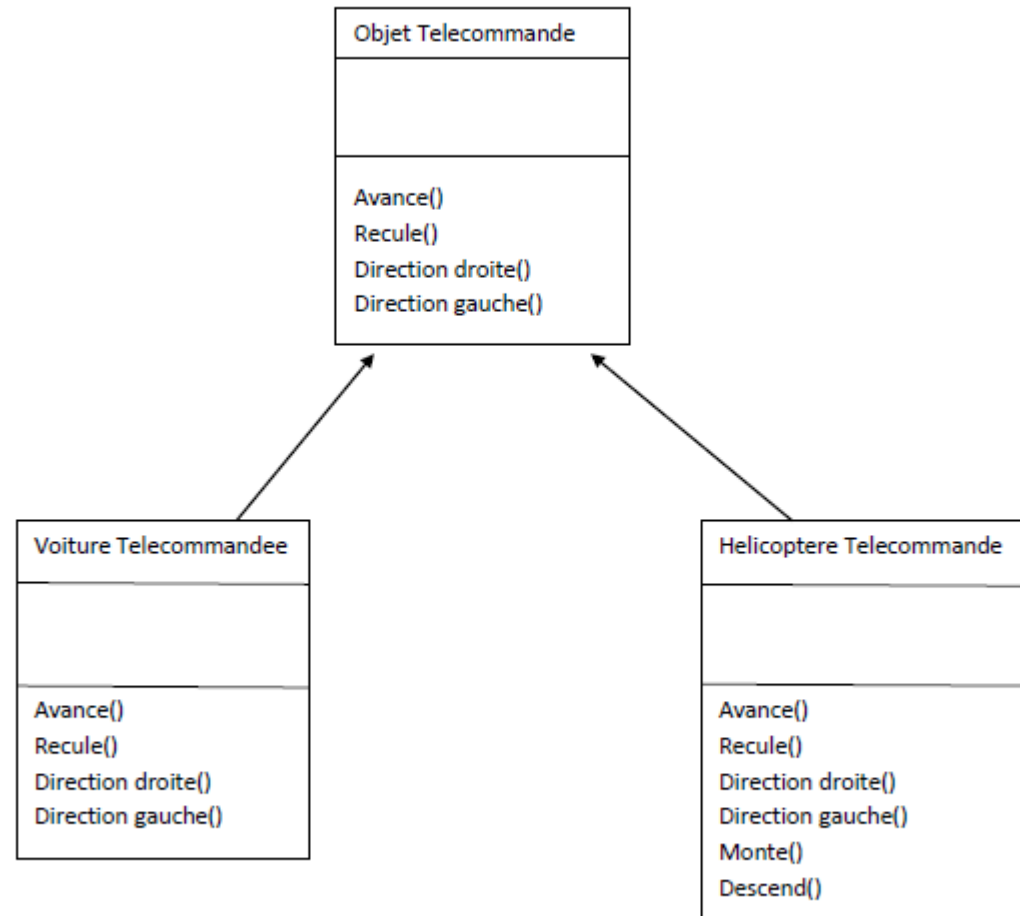
# Une interface

- Une interface est une classe qui contient uniquement des méthodes abstraites
- Elle est différentes d'une classe abstraite car
  - Elle ne contient aucun attribut
  - aucune des méthodes de la classe n'est implémentée
- Elle ne peut pas être utilisée pour créer un objet

# Une interface

- Permet de définir un comportement qui devra être respecté par les classes dérivées
- Les classes dérivées devront obligatoirement implémenter chaque méthode définie dans l'interface

# Exemple



# Classe Objet Telecommande

- Les 4 méthodes sont des méthodes abstraites
  - Représentent le comportement commun des objets télécommandés
- Aucun code de méthodes n'est créé dans la classe

# Classes dérivées

- Les classes Voiture Telecommandee et Helicoptère Telecommande auront obligatoirement les méthodes : avance, recule, directionDroite, directionGauche
- Le code de ces 4 méthodes devra être défini dans chacune des classes dérivées
- Les classes dérivées peuvent avoir d'autres méthodes, exemple monte et descend pour la classe Helicoptere Telecommande

# Intérêt des interface

- Définir un comportement de base qui devra être respecté par toutes les classes dérivées
- Permet d'avoir un comportement identique (même nom de méthode, même paramètre...) pour différentes classes ayant des points communs



# Comparaison

## Classe abstraite

- Une classe ne peut hériter que d'une seule classe (abstraite ou non)
- Peut contenir des attributs
- A un constructeur
- Peut contenir des méthodes et leur code
- Utilisée pour optimiser l'écriture du code (factorisation + imposer des méthodes aux classes dérivées)

## Interface

- Une classe peut implémenter plusieurs interfaces
- Ne contient que des méthodes abstraites
- Utilisée pour définir un comportement commun (même signature de méthode pour plusieurs classes)

# Jeu des héros - bonus

# Principe

- Au début du jeu, chaque joueur choisit un héros et reçoit 3 bonus (1 de chaque type)
- Lorsqu'il lance une attaque, le joueur peut jouer simultanément un bonus

# Bonus Bouclier



- Selon le niveau du bouclier, il peut bloquer une attaque
  - Niveau supérieur à 50 : blocage
- Son utilisation fait baisser le niveau du bouclier et peut augmenter le nombre de points gagnés par le défenseur
  - Chaque utilisation provoque 20% de dommages
  - Fait gagner un point supplémentaire au défenseur

# Bonus Arme



- Selon le niveau de l'arme, peut permettre de lancer 2 attaques.
  - Niveau supérieur à 100 : 2 attaques
- Son utilisation fait baisser le niveau de l'arme et peut augmenter le nombre de points gagnés par le défenseur
  - Chaque utilisation provoque 40 points de dommages
  - Fait gagner un point supplémentaire à l'attaquant tant que son niveau est supérieur à 40

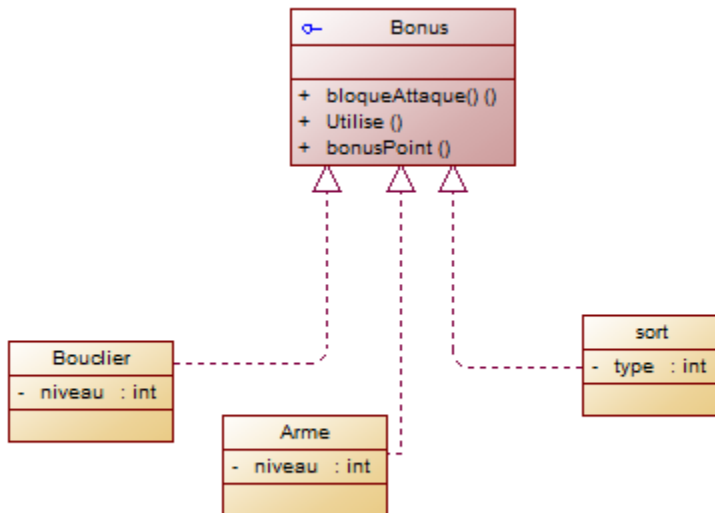
# Bonus Sort



- Selon le sort, peut bloquer une attaque ou augmenter le nombre de points gagnés
- Ne s'utilise qu'une seule fois
  - Maléfice ou enchantement : blocage
  - ModeSupra : +2 points pour l'attaquant ou le défenseur

# Structure des classes

- L'interface Bonus permet de s'assurer du comportement des bonus



# Exercice

- Créer un package pour gérer les bonus appelé option
- Créer une interface Bonus



# Exercice

- Créer les classes Arme, Bouclier et Sort qui implémentent l'interface Bonus
- Créer les attributs, les constructeurs, les accesseurs et les méthodes des 3 classes
- Ajouter une classe de test dans le package option pour vérifier le bon fonctionnement des classes