Programmation objet

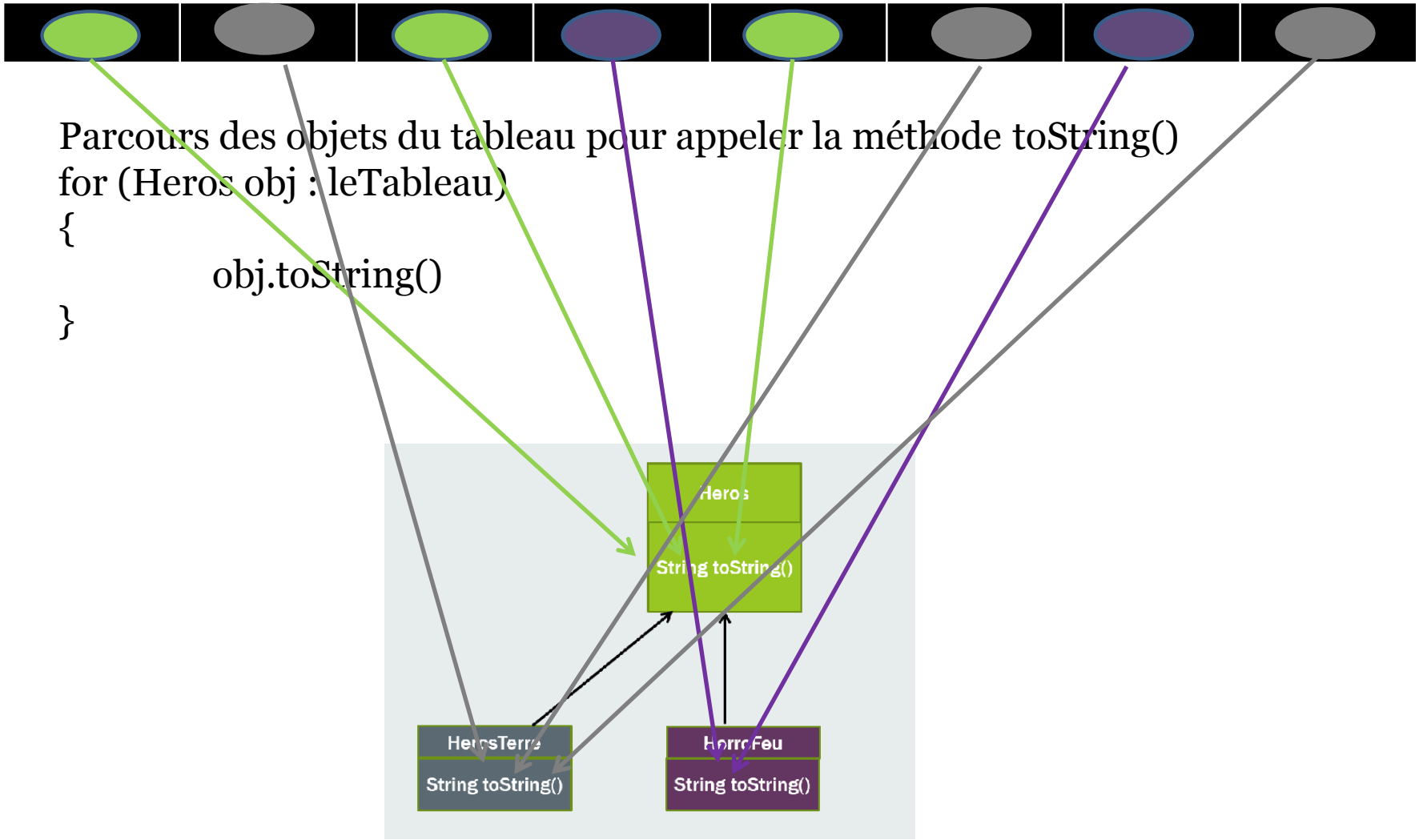
Héritage
Classe abstraite
Collections

Polymorphisme

- Création d'un tableau de Heros contenant des HerosTerre et HerosFeu
- Boucle pour parcourir les héros du tableau
  - Appel de la méthode toString()

- En fonction du type précis de l'objet Heros
  - Appel de toString() de Heros
  - Appel de toString() de HerosTerre
  - Appel de toString() de HerosFeu

Mise en œuvre du polymorphisme



Parcours des objets du tableau pour appeler la méthode toString()

```
for (Heros obj : leTableau)
{
        obj.toString()
}
```

L'opérateur instanceof

- Permet de comparer un objet avec une classe
- Syntaxe
  - Object instance of Classe →true ou false

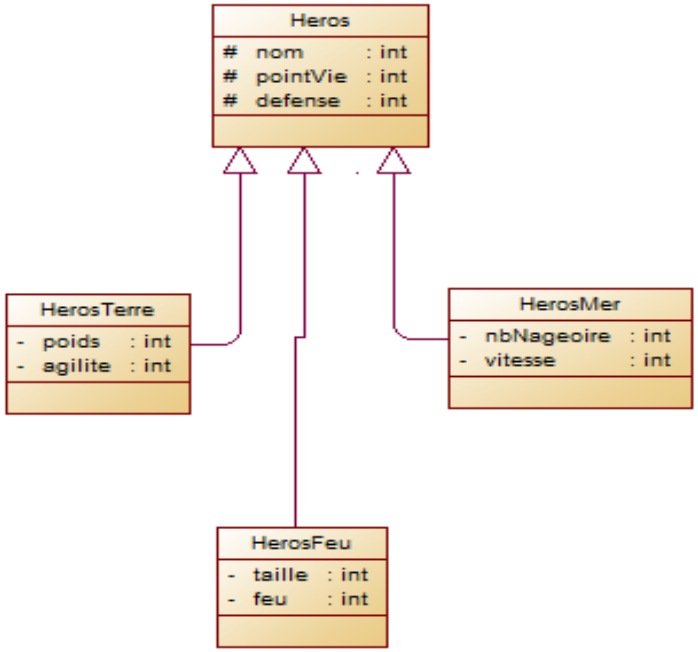
- Afficher un message indiquant le type de Héros dans la boucle précédente

Classe abstraite

Les héros

- Gestion de types de héros ayant des caractéristiques différentes

Les différents héros



Héros de terre

- Ces héros sont caractérisés par un poids (en kg) et un degré d'agilité.
- La puissance de ces héros sera calculée grâce à la formule suivant : *puissance =poids/50 * degré d'agilité*

Héros de feu

- Ces héros sont caractérisés une taille (en centimètres) et une puissance de feu.

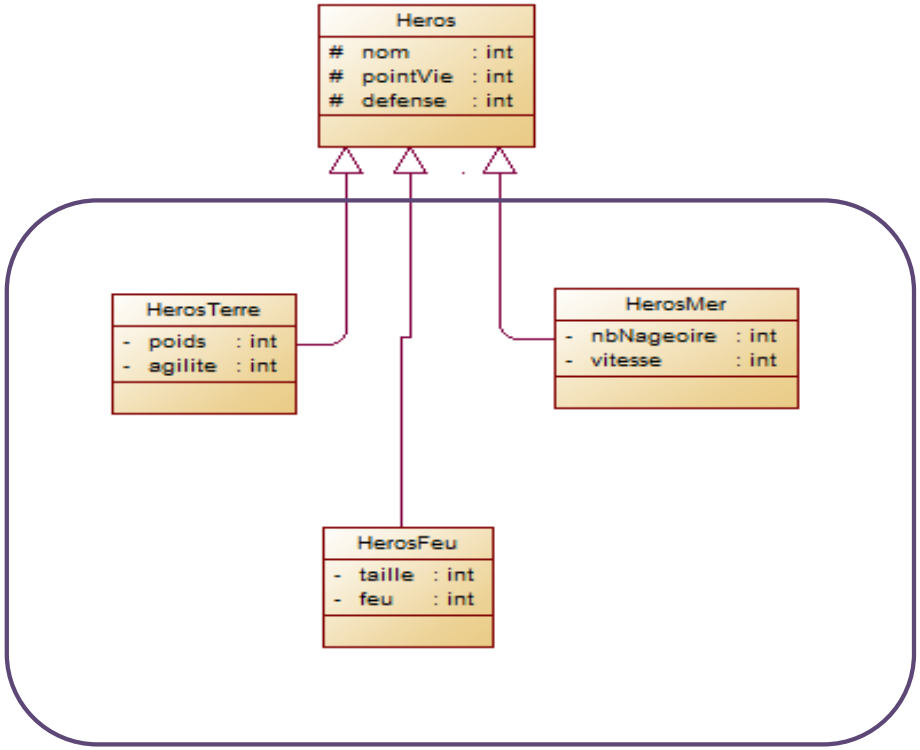
- La puissance des ces héros sera calculée grâce à la formule suivante : *puissance = taille/100 * puissance de feu*

Héros de mer

- Ces héros sont caractérisés un nombre de nageoire et une vitesse (en km/h)

- La puissance des ces héros sera calculée grâce à la formule suivante : *puissance = nombre de nageoires * vitesse*

Les héros du jeu



La classe Heros

Bilan

- Utile pour optimiser l'écriture du code
- Inutile pour la création d'objets

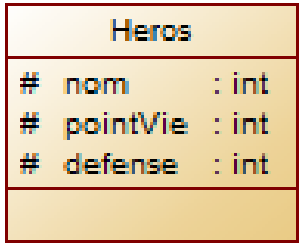
Evolution

- La classe Héros devient une classe abstraite
- Impose la création de méthodes par les classes filles

Classe abstraite

Particularités

- Une classe est abstraite si elle contient au moins une méthode abstraite
- On ne peut pas créer un objet d'une classe abstraite
- Ajout du mot clé abstract



Méthode abstraite

- Définie avec le mot clé abstract
- N'a pas de corps dans la classe mère

- Doit être redéfinie dans toutes les classes filles
  - Une classe fille qui ne redéfinie par une méthode abstraite doit définir la méthode comme abstraite et être elle-même abstraite

Structure de Heros

- Pour que le jeu fonctionne, chaque type de Heros doit avoir
  - Une méthode puissance qui retourne la puissance du héros (utilisée dans la méthode combat)
  - Une méthode combat qui définit le mode de combat du type de Heros

Exercice

- Evolution de la classe Heros
  - La classe devient abstraite
  - Les méthodes puissance et combat sont définies comme méthodes abstraites

- Une erreur est générée pour toute création d'un objet Heros
  - Le type Heros peut être utilisé pour le polymorphisme

Exercice

- Créer la classe HerosMer qui hérite de Heros avec
  - Ses attributs spécifiques
  - Un constructeur qui initialise tous les attributs
  - Un constructeur par défaut avec le nom en paramètres
  - La méthode toString()

Effet de la classe abstraite

Héritage classe abstraite

- Erreur sur la classe HerosMer
  - HerosMer is not abstract and does not override abstract method
- Génération des méthodes obligatoires
  - Puissance
  - Combat

Code généré

- Mot clé Override
  - Définit une redéfinition
  - Vérification par le compilateur de la signature de la méthode

Combat

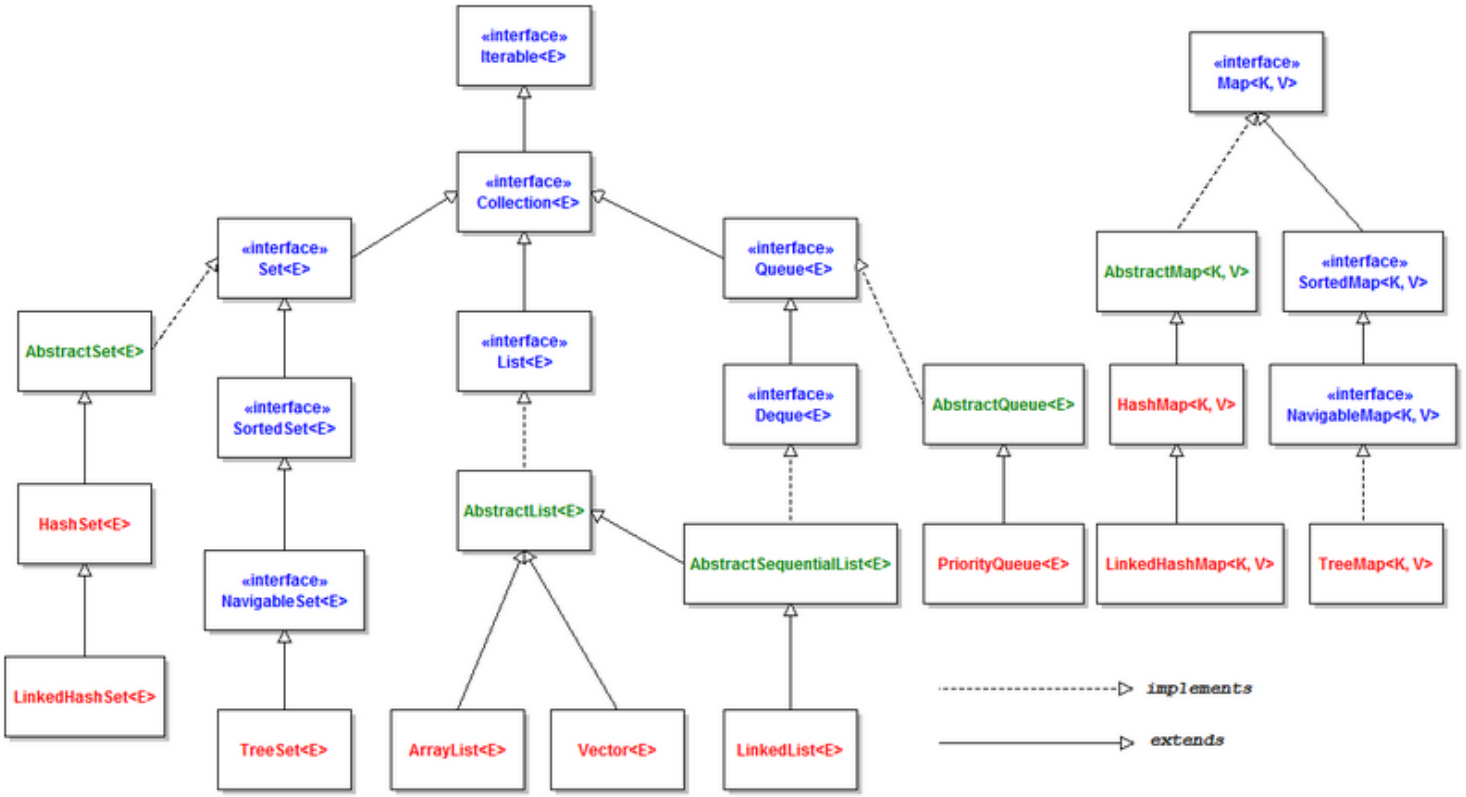
- La méthode combat fonctionne de la façon suivante :
  - Il doit rester au héros de mer au moins 20 points de vie pour pouvoir porter une attaque, sinon l'attaque n'aura pas lieu.
  - Si la puissance du héros qui attaque est supérieure à la puissance du héros en défense, l'attaquant fait perdre 60 points au défenseur.
  - Si les deux puissances sont égales, il fait perdre 20 points au défenseur.
  - Si la puissance du défenseur est supérieure à celle de l'attaquant, aucun ne perd de points.
- A la fin de la méthode combat, on affiche le résultat du combat si un des deux héros a été tué.

Test de la classe

- Créer 2 HerosMer à ajouter dans le tableau de Heros
  - Tester la méthode toString
  - Tester la méthode puissance

Collections

Classes du framework Java



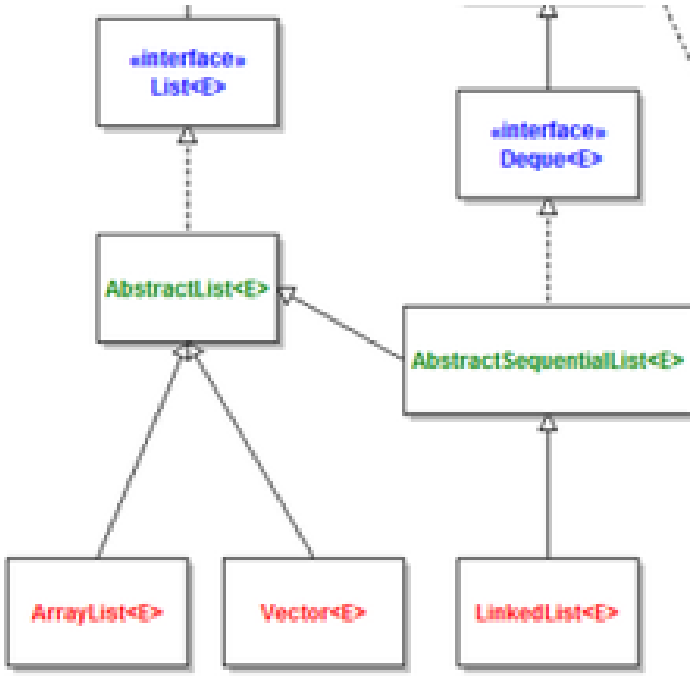
Class diagram of Java Collections framework

Intérêt de leur utilisation

- Facilité d'utilisation (même structure de classe)
- Méthodes validées
- Gestion automatique des manipulations sur les listes

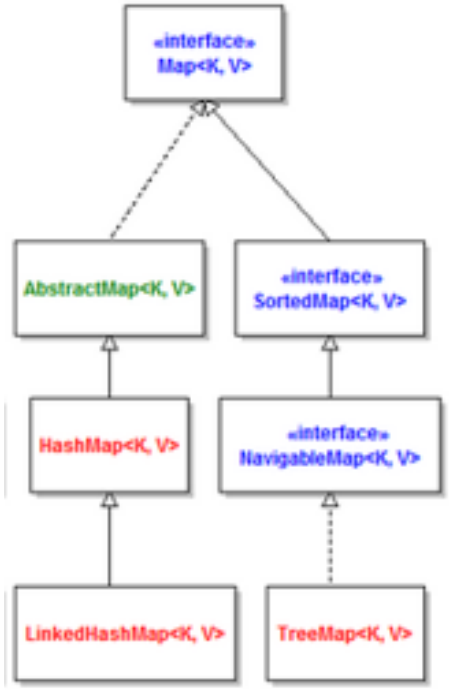
Différentes collections

- Objets organisés sous le format d'un tableau
  - Indice pour accéder à un objet
  - Taille extensible



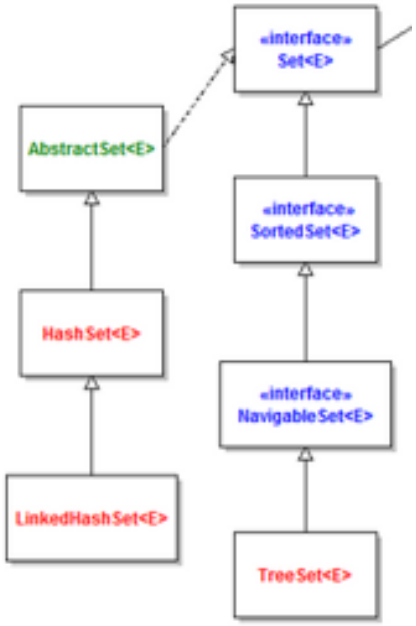
Différentes collections

- Liste basée sur des couples clé-valeur
  - Permet de chercher une valeur à partir de sa clé



Différentes collections

- Liste d'objets sans doublons
  - Liste triée TreeSet



ArrayList

Tableau classique

Ajout de tous types de valeurs

```
ArrayList tab = new ArrayList();
tab.add("test");
tab.add(3);
System.out.println(tab.get(0));
```

Tableau typé

Ajout d'objets Heros (ou sous-type) uniquement

```
ArrayList<Heros> tabHeros = new ArrayList<Heros>();
tabHeros.add(ht1);
tabHeros.add(hf1);
tabHeros.add(ht2);
tabHeros.add(hf2);

//test du polymorphisme avec toString()
for (Heros h : tabHeros)
    System.out.println(h.toString());
```

Utilisation de ArrayList

Classes à utiliser

- ArrayList
  - https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html
- ListIterator
  - https://docs.oracle.com/javase/7/docs/api/java/util/ListIterator.html

A faire

- Rechercher si un héros est présent dans la liste
- Ajouter un héros en 3ème position
- Afficher la liste de héros à l'aide d'un iterateur
- Supprimer un Heros de la liste
- Afficher le nombre de Heros de la liste

Evolution du moteur du jeu

Utilisation ArrayList

- Remplacer les tableaux de Joueur et Heros par des ArrayList

Création des héros

- Modifier la méthode Heros pour ajouter des héros de terre, feu ou mer selon le choix de l'utilisateur
- Ajouter une méthode creationHerosDefaut qui créé une liste de quinze héros avec les propriétés de votre choix
- L'utilisateur pourra décider de créer les héros lui-même ou d'utiliser les héros par défaut.