Programmation Orientée Objet – TP3

S.l. Alexandru Mitrea

S.l. Iulia-Cristina Stănică

iulia.stanica@gmail.com

Objectifs pour aujourd'hui

- Interfaces
- Classes abstraites vs Interfaces
- Comparator
- Ecrire dans les fichiers

Les Interfaces

- **Une interface** est une «classe » purement abstraite dont **toutes les méthodes sont abstraites et publiques.**
- Elle peut aussi contenir des constantes, des méthodes statiques.
- Une interface est implicitement abstraite; toutes les méthodes sont implicitement abstraites
- Mot clef dans la classe: **implements**
- Une classe peut implémenter plusieurs interfaces; elle doit implémenter toutes les méthodes des interfaces

Les Interfaces - Exemples

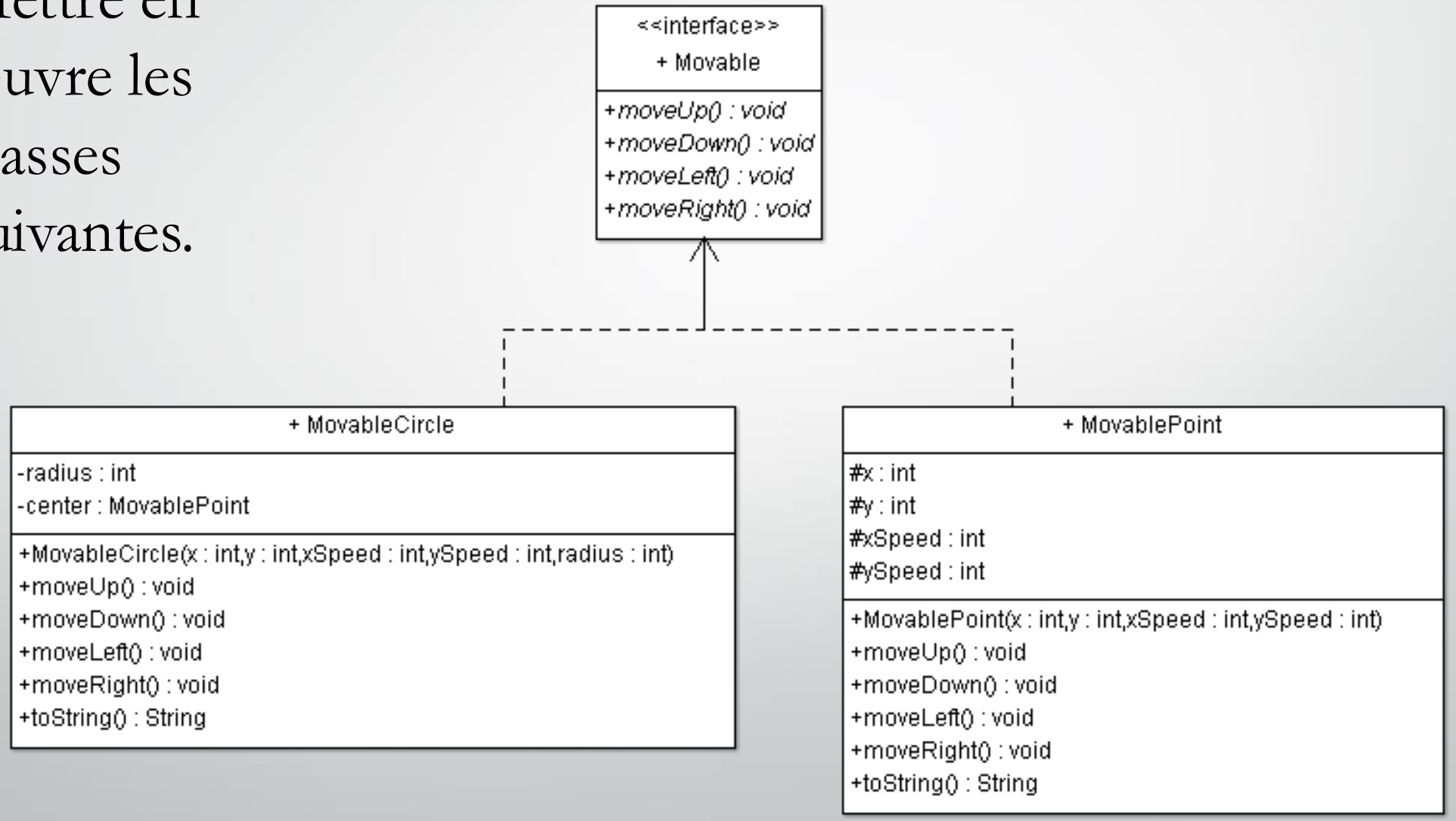
```
interface Vehicle {

   // all are the abstract methods.
   void changeGear(int a);
   void speedUp(int a);
   void applyBrakes(int a);
}

class Bicycle implements Vehicle{

   int speed;
   int gear;

    // to change gear
   @Override
   public void changeGear(int newGear){

      gear = newGear;
   }

   // to increase speed
   @Override
   public void speedUp(int increment){

      speed = speed + increment;
   }

   // to decrease speed
   @Override
   public void applyBrakes(int decrement){

      speed = speed - decrement;
   }

   public void printStates() {
       System.out.println("speed: " + speed
           + " gear: " + gear);
   }
}
```

Classe Abstraite	Interface
1) Peut avoir les deux: **méthodes abstraites et non-abstraites**. Peut avoir constructeur.	L'interface ne peut avoir que des **méthodes abstraites**. Depuis Java 8, il peut également avoir des méthodes statiques. N'a pas de constructeurs.
2) N**e supporte pas l'héritage multiple**.	**Supporte l'héritage multiple**.
3) Peut avoir des variables **final, non-final, static et non-static**.	Seulement **variables static et final (constantes)**.
4) Peut contenir l'**implémentation d'une interface**.	Ne peut pas contenir l'**implémentation d'une classe abstraite**.
5) Déclaration: **abstract keyword**	Déclaration: **interface keyword**
6) Une classe **abstraite** peut étendre une autre classe Java et implémenter plusieurs interfaces Java.	Une **interface** peut étendre seulement une autre interface Java.
7) On utilise "extends".	On utilise "implements".
8) Peut avoir des membres **private, protected**, etc.	Les membres sont **public** par défaut0

9)**Exemple:**
public abstract class Shape{
public abstract void draw();
}
	Exemple:
	public interface Drawable{
	void draw();
	}

Ex 1

- Mettre en œuvre les classes suivantes.



Interface Comparator

- L’interface **Comparator** permet d’ordonner les objets des classes définies par l’utilisateur. Le comparateur d'objets est capable de comparer deux objets de deux classes différentes.
- L’interface contient une méthode ***compare (arg1, arg2)*** qui contient les deux objets qui doivent être comparés
- La méthode ***compare*** renvoie un **int < 0, == 0 ou > 0** pour indiquer si le premier objet est inférieur, égal ou supérieur à l'autre.

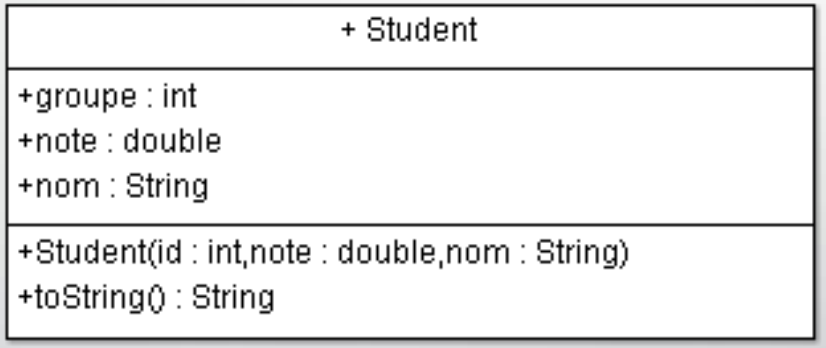
Interface Comparator – Ex:

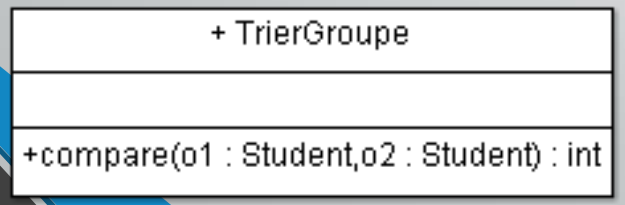
```
public class Player {
    private int ranking;
    private String name;
    private int age;

    // constructor, getters, setters  }
```

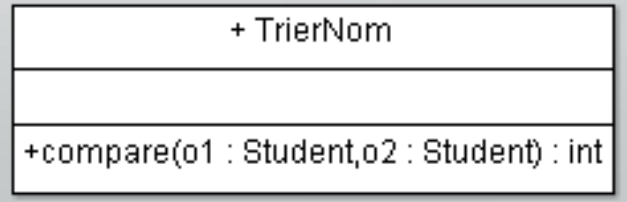
```
public class PlayerRankingComparator implements Comparator<Player> {

    @Override
    public int compare(Player firstPlayer, Player secondPlayer) {
        return (firstPlayer.getRanking() - secondPlayer.getRanking());
    }
}
```

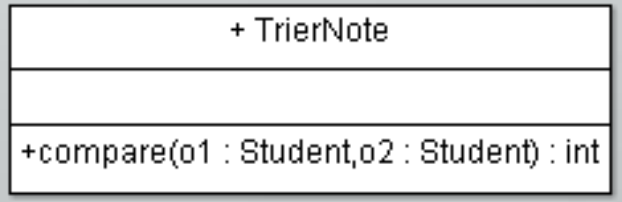
Ex 2

- Mettre en œuvre les classes suivantes. TrierGroupe, TrierNom et TrierNote implémentent Comparator.
- Vous avez la classe Test sur moodle. Afficher le résultat dans un fichier.









Rappel – écriture fichier

```
FileWriter fw = new FileWriter(“output.txt”);
BufferedWriter out = new BufferedWriter(fw);
out.write(“test”);
out.newLine();
out.close();
```

Ex 3

- Créer un petit système de gestion des commandes d'un restaurant.
- Utiliser le comparateur pour trier les commandes selon différents critères de votre choix.
- Afficher les resultats du tri dans un fichier externe.