

CONCEPTION ORIENTÉE OBJET - COO

DIAGRAMME DE CLASSES UML VERS JAVA

🎓 2A - Bachelor Universitaire de Technologie

🏛️ IUT d'Orsay - Université Paris-Saclay - 2025/2026



Idir AIT SADOUNE

idir.ait-sadoune@universite-paris-saclay.fr

PLAN

- [Transcription des classes](#)
- [Transcription des associations](#)
- [Transcription des collections](#)

[Retour au plan](#) - [Retour à l'accueil](#)

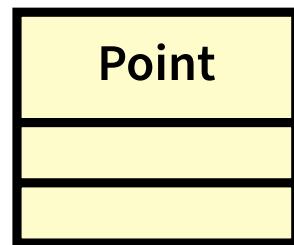
PLAN

-  [Transcription des classes](#)
-  [Transcription des associations](#)
-  [Transcription des collections](#)

[Retour au plan](#) - [Retour à l'accueil](#)

TRANSCRIPTION DES CLASSES

A chaque **classe UML** correspondra une **classe Java**.

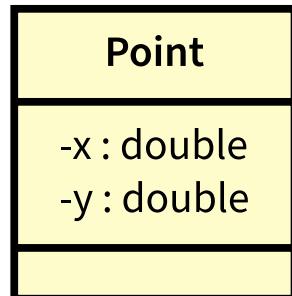


```
1 class Point{  
2     ...  
3 }
```

TRANSCRIPTION DES CLASSES

A chaque **attribut** d'une **classe UML** correspondra un **attribut** d'une **classe Java**.

- cardinalité $\leq 1 \rightarrow$ Type nomAttribut;
- cardinalité $> 1 \rightarrow$ Collection<Type> nomAttribut;



```
1 class Point{  
2     private double x;  
3     private double y;  
4 }
```

REMARQUE

Les visibilités des éléments de modélisation UML seront les mêmes dans leurs correspondants en Java.

TRANSCRIPTION DES CLASSES

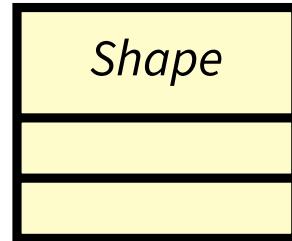
A chaque **opération** d'une **classe UML** correspondra
une **méthode** d'une **classe Java**.

Point
-x : double
-y : double
+ distanceTo(autre : Point) : double

```
1 class Point{  
2     private double x;  
3     private double y;  
4  
5     public double distanceTo(Point autre){  
6         // TO DO  
7     }  
8 }
```

TRANSCRIPTION DES CLASSES

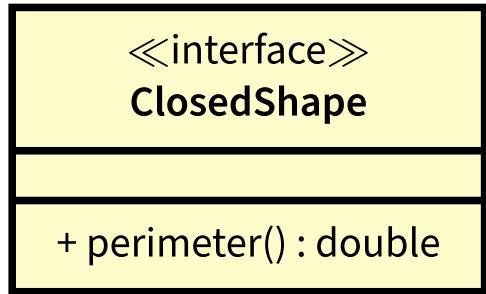
A chaque **classe UML abstraite** correspondra
une **classe Java abstraite**.



```
1 abstract class Shape{  
2     ...  
3 }
```

TRANSCRIPTION DES CLASSES

A chaque **interface UML** correspondra une **interface Java**.



```
1 interface ClosedShape{  
2     public abstract double perimeter();  
3 }
```

PLAN

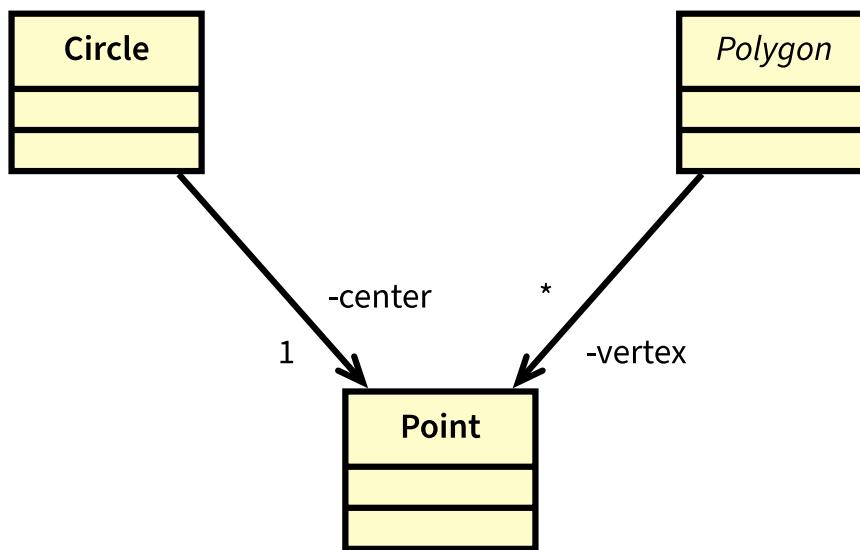
- Transcription des classes
- Transcription des associations
- Transcription des collections

[Retour au plan](#) - [Retour à l'accueil](#)

TRANSCRIPTION DES ASSOCIATIONS

- A chaque **association UML** correspondra **un attribut** dans les **classes Java** participantes à l'association.
- L'**attribut** est ajouté uniquement si l'**association est navigable**.
- Le **nom de l'attribut** est le nom de l'**extrémité navigable**.
 - cardinalité $\leq 1 \rightarrow$ **Type nomExtremite;**
 - cardinalité $> 1 \rightarrow$ **Collection<Type> nomExtremite;**

TRANSCRIPTION DES ASSOCIATIONS



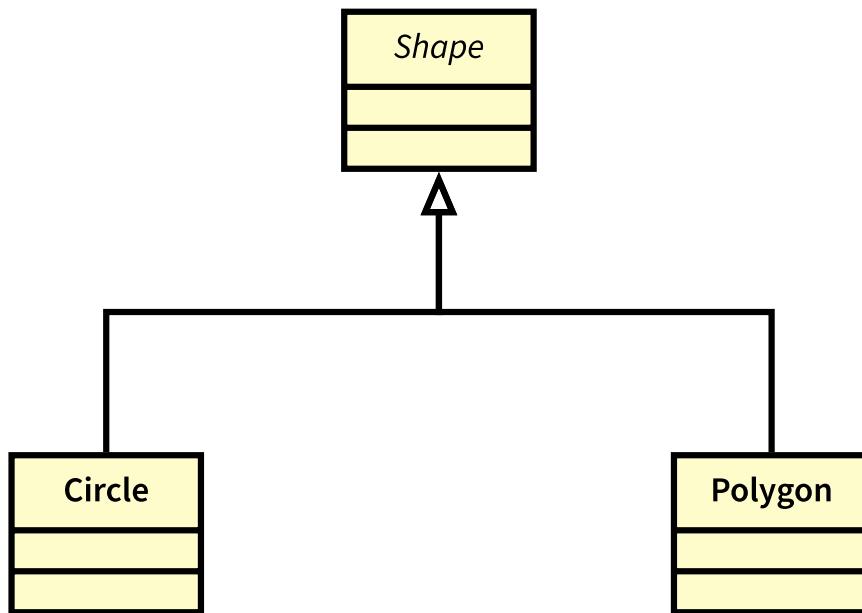
```
1 class Point{  
2   ...  
3 }
```

```
1 class Circle{  
2   private Point center ;  
3 }
```

```
1 abstract class Polygon{  
2   private Collection<Point> vertex;  
3 }
```

TRANSCRIPTION DES ASSOCIATIONS

La relation de **généralisation en UML** est traduite
par une relation d'**héritage en Java**.



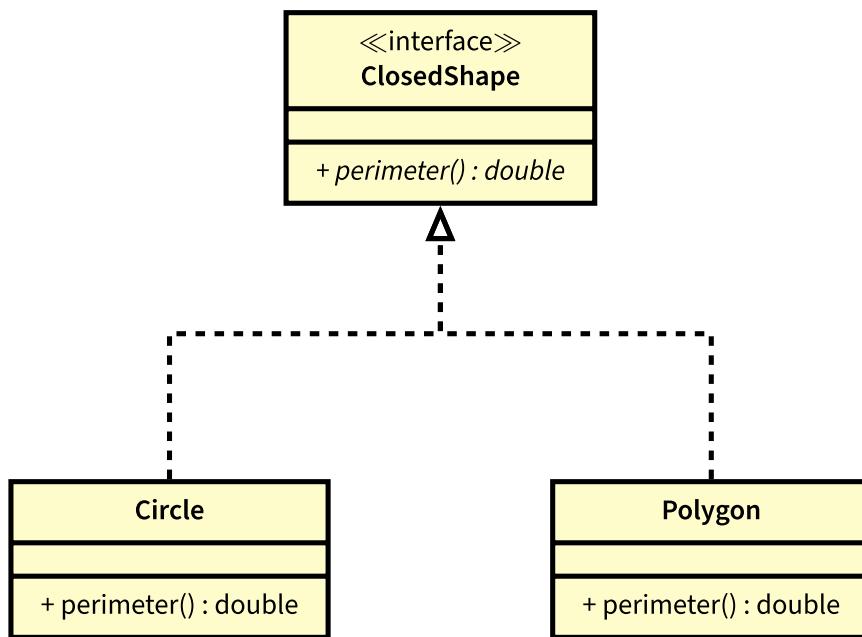
```
1 abstract class Shape{  
2     ...  
3 }
```

```
1 class Circle extends Shape{  
2     ...  
3 }
```

```
1 class Polygon extends Shape{  
2     ...  
3 }
```

TRANSCRIPTION DES ASSOCIATIONS

La relation de **réalisation en UML** est traduite par une relation d'**implémentation en Java**.



```
1 interface ClosedShape{
2     public abstract double perimeter();
3 }
```

```
1 class Circle implements ClosedShape{
2     public double perimeter(){
3         // TO DO
4     }
5 }
```

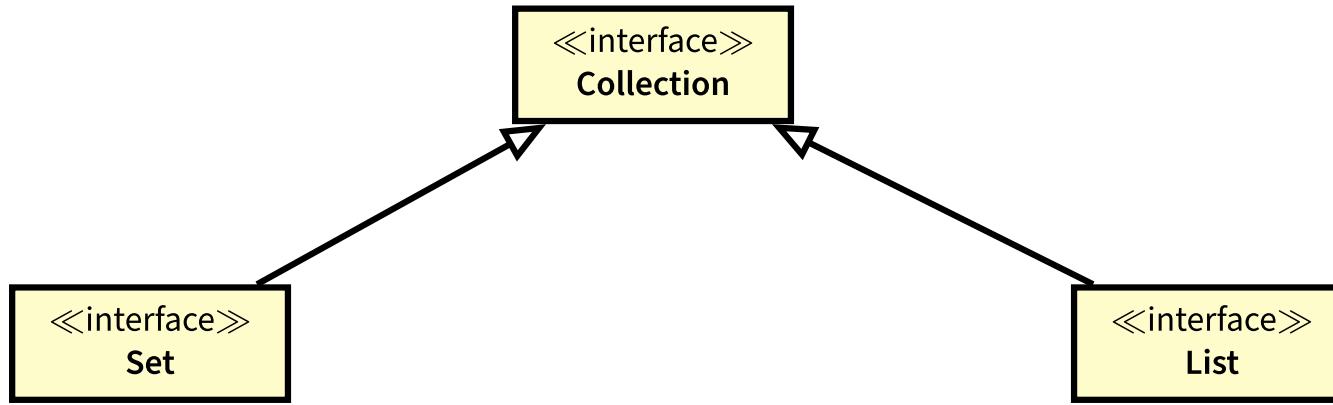
```
1 class Polygon implements ClosedShape{
2     public double perimeter(){
3         // TO DO
4     }
5 }
```

PLAN

- [Transcription des classes](#)
- [Transcription des associations](#)
- [Transcription des collections](#)

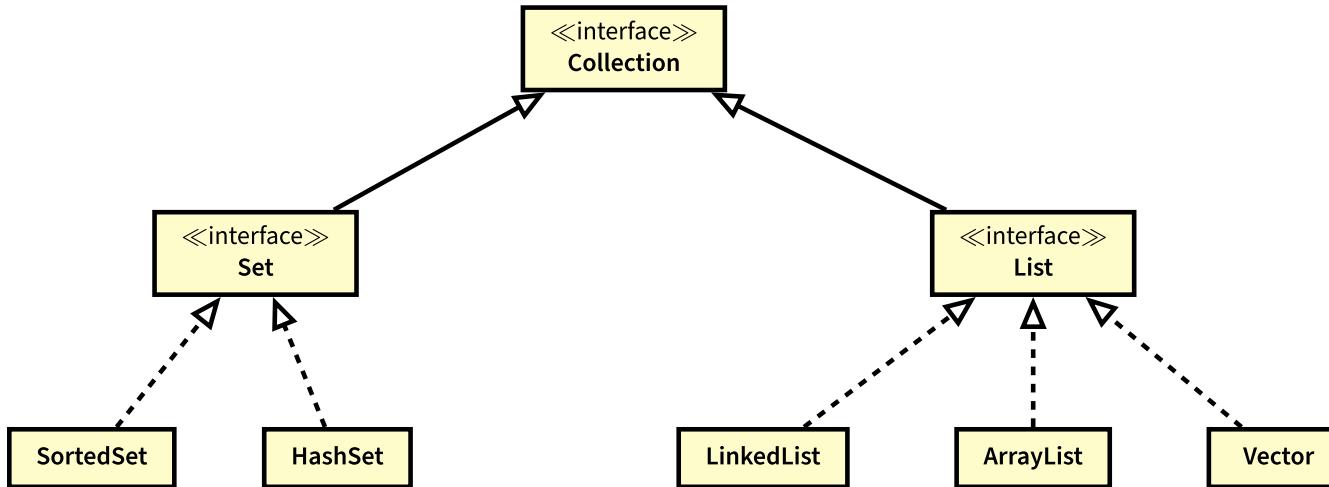
[Retour au plan](#) - [Retour à l'accueil](#)

LES COLLECTIONS EN JAVA



- **Collection** → interface qui gèrent toutes les **collections** dynamiques.
- **Set** → interface pour des **collections** qui n'autorisent **pas de doublons**.
- **List** → interface pour des **collections** qui **autorisent des doublons** et un **accès direct** à un élément.
- ...

LES COLLECTIONS EN JAVA



- `Vector` → implémente l'interface `List` (synchronisé)
- `ArrayList` → implémente l'interface `List` (non synchronisé)
- `LinkedList` → implémente l'interface `List` (doublement chaînée)
- `HashSet` → implémente l'interface `Set`
- `SortedSet` → implémente l'interface `Set` (ordonné)

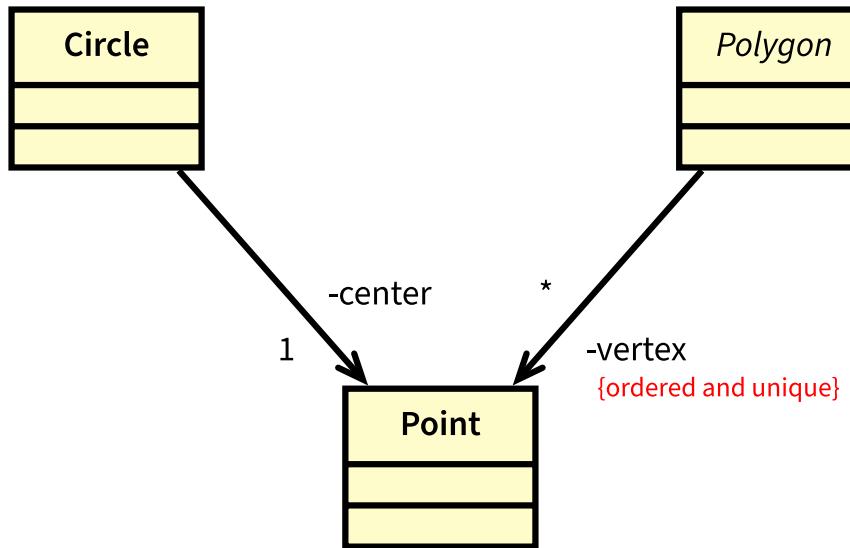
TRANSCRIPTION DES COLLECTIONS

Le type de la **Collection** affecté à un **attribut** en **Java**, dépend des **propriétés** de l'**attribut/association UML**:

- aucune propriété → à configurer manuellement.
- **ordered** → traduit en **ArrayList** ou **LinkedList**.
- **unique** → traduit en **HashSet**.
- **ordered** et **unique** → traduit en **SortedSet**.

TRANSCRIPTION DES COLLECTIONS

EXAMPLE



```
1 class Point{  
2   ...  
3 }
```

```
1 class Circle{  
2   private Point center ;  
3 }
```

```
1 abstract class Polygon{  
2   private SortedSet<Point> vertex;  
3 }
```

MERCI

[Version PDF des slides](#)

[Retour à l'accueil](#) - [Retour au plan](#)