HE2B ESI

Analyse 1

ANA2

Geneviève Cuvelier (CUV)
Christine Leignel (CLG)
Thibaut Nicodème (TNI)
Pantélis Matsos (PMA)

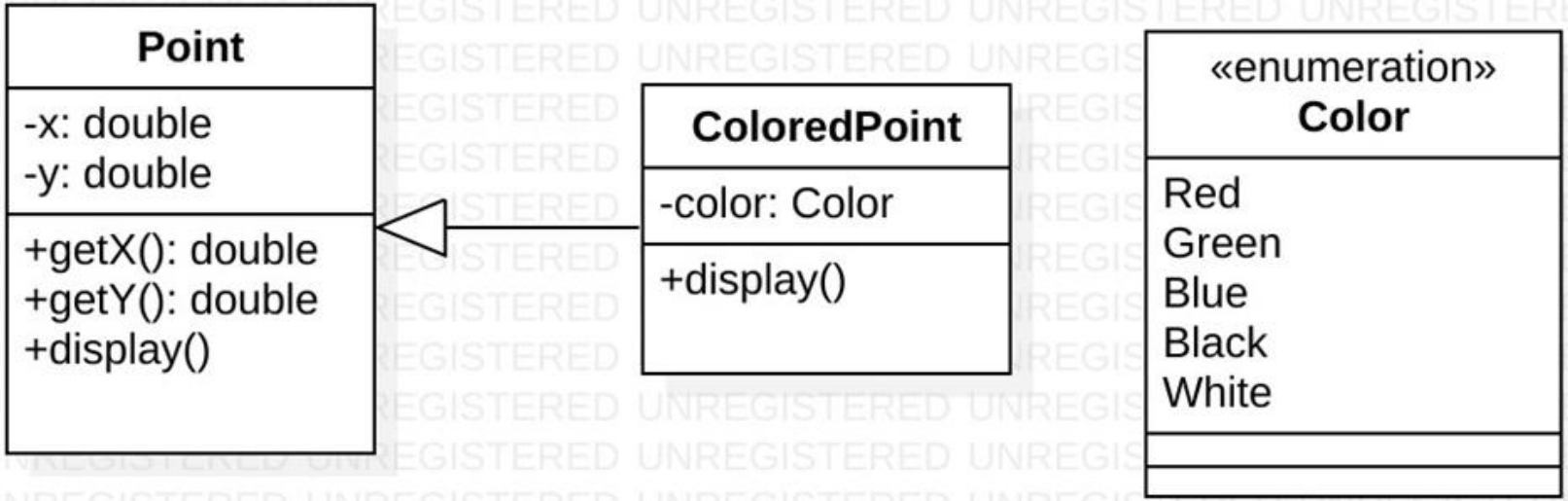
v1.2

2020-2021

Où en sommes-nous ?

1. Qu’est-ce que l’analyse?
2. Diagramme d’activités
3. Les classes et objets
4. Les associations 1-1 et 1-N
5. Les associations N-N
6. Les compositions et énumérations
7. Les classes associations
8. L’héritage
9. Les interfaces

Héritage - Rappel de Java



Héritage - Rappel de java

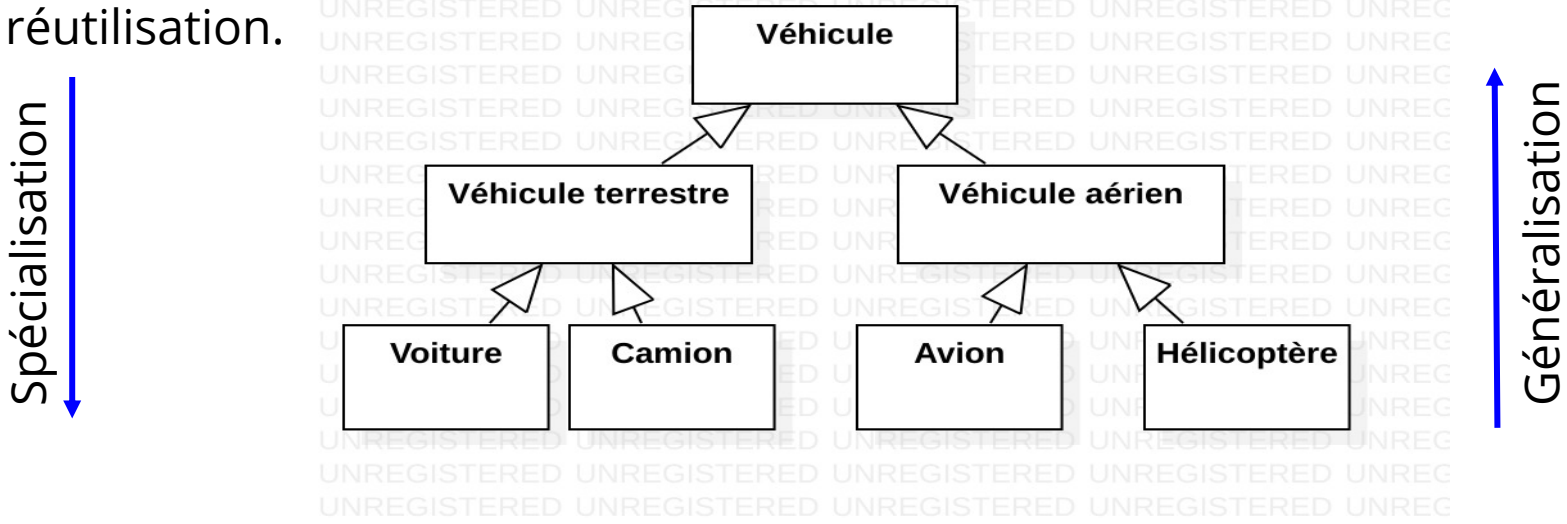
```java
public class Point
{
    public static final Point ORIGIN = new Point(0,0);

    private double x;
    private double y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
    public void display() {
        System.out.println("(" + x + "," + y + ")");
    }
    public double getX() {
        return x;
    }
    public double getY() {
        return y;
    }
}
```

```java
public class ColoredPoint extends Point
{
    private Color color;

    public ColoredPoint(double x, double y, Color color) {
        super(x, y);
        this.color = color;
    }

    public Color getColor() {
        return color;
    }

    @Override
    public void display() {
        super.display();
        System.out.println("… et je suis " + color);
    }
}
```

sources pbt

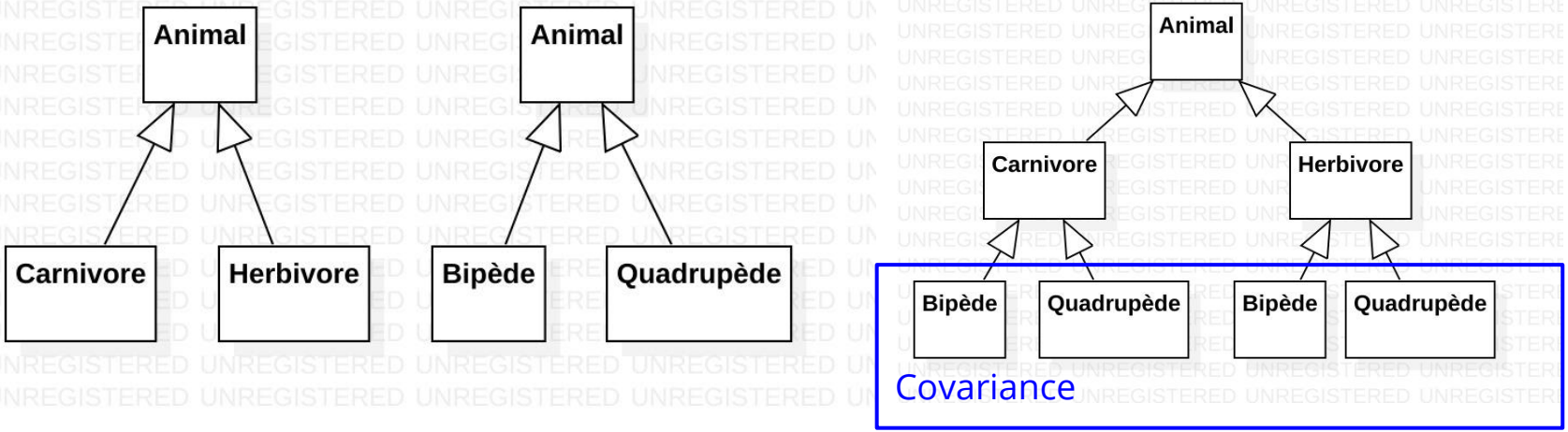
Héritage - Principes

- L’héritage permet de classifier les classes dans une hiérarchie.
- On peut ainsi définir dans une « super-classe » les attributs et les méthodes communs à plusieurs « sous-classes ». Cela permet donc la réutilisation.



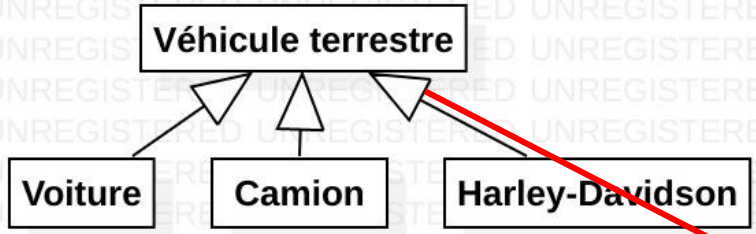
Héritage - La difficulté de classer

- Les bonnes classifications sont stables et extensibles.
- Il n’y a pas toujours une classification unique, parfois plusieurs classifications possibles.

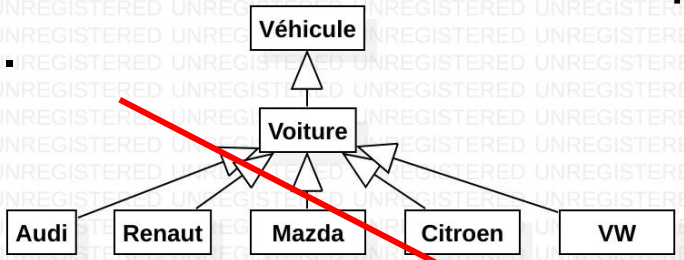


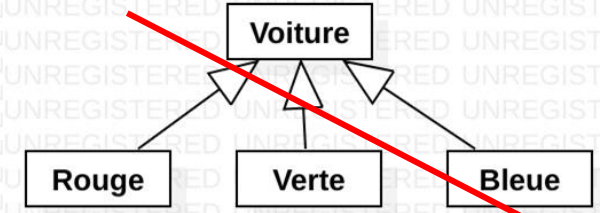
Héritage - La difficulté de classer

- Les bonnes classifications comportent des niveaux d'abstraction équilibrés.

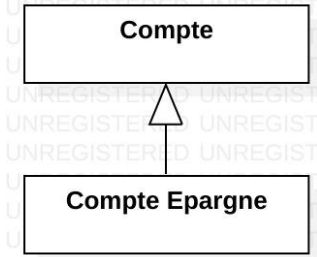


- Les bonnes classifications comportent peu de sous-classes.



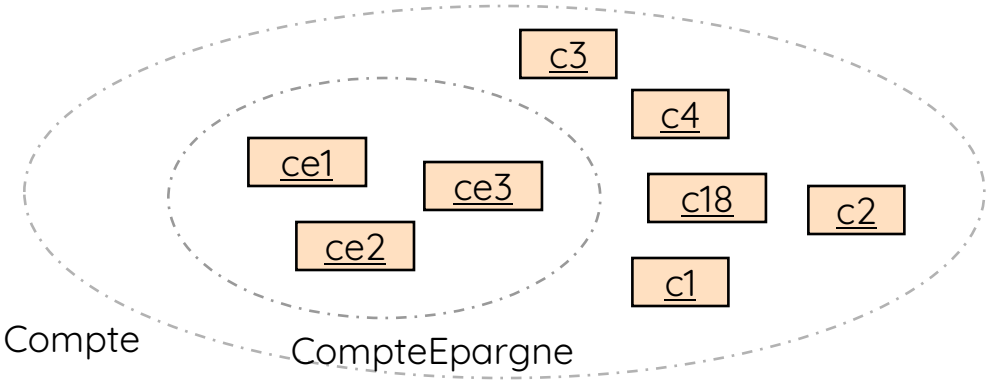


Héritage - La difficulté de classer



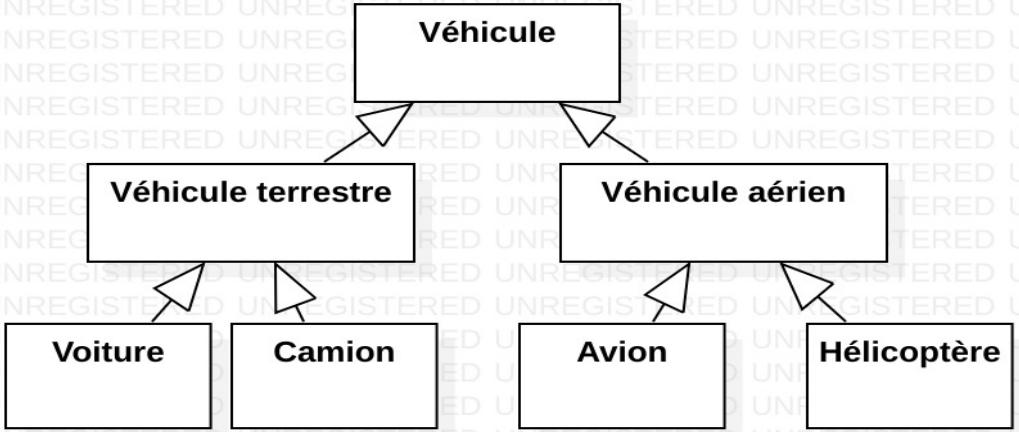
Un compte épargne est un compte.

Les objets ce1, ce2, ce3 forment un sous-ensemble de l'ensemble Compte

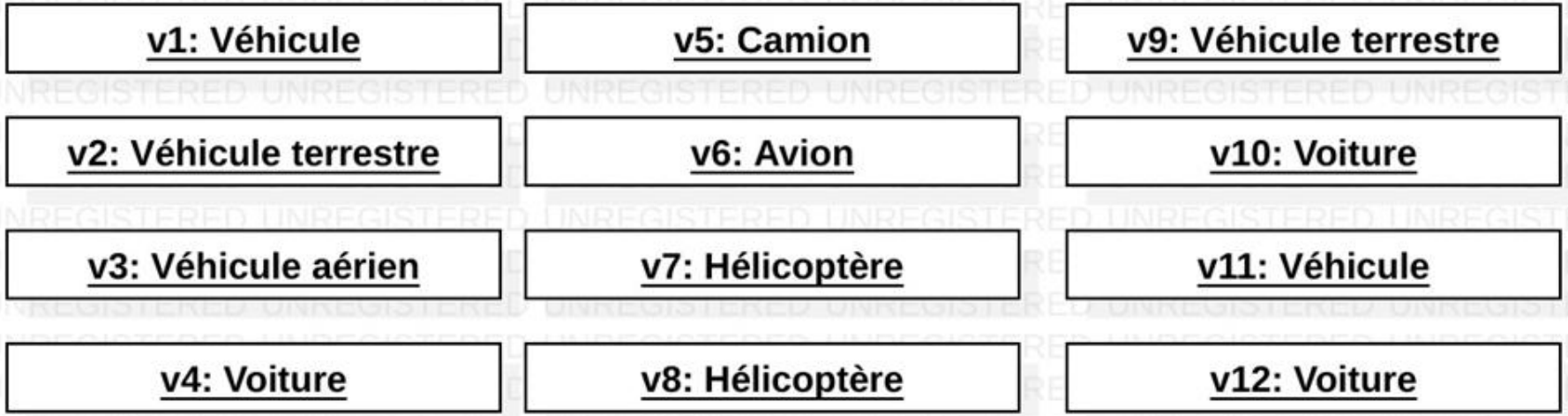


Héritage - Principes

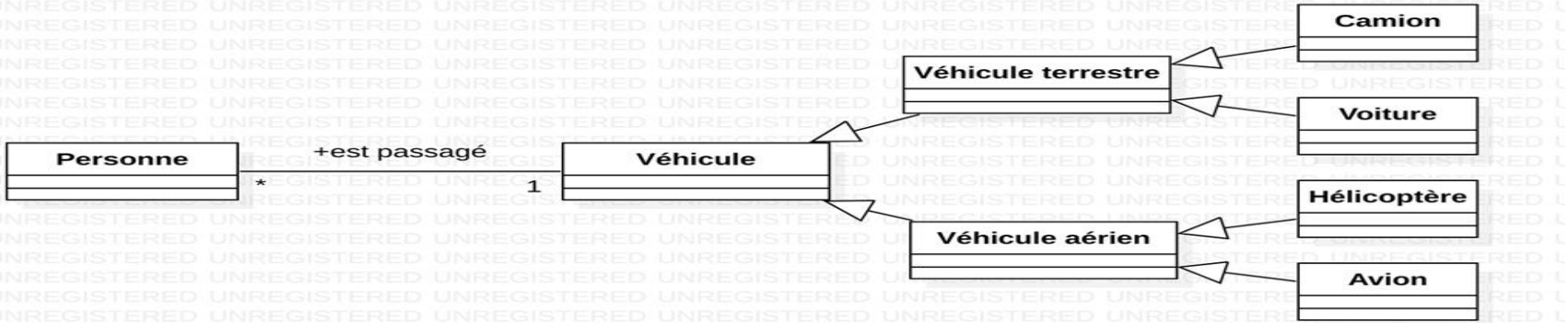
- On a une hiérarchie de classes quand on peut dire: « est un(e) »
- L’association d’héritage (de généralisation/Spécialisation)
  - ne porte pas de nom
  - n’a pas de multiplicité

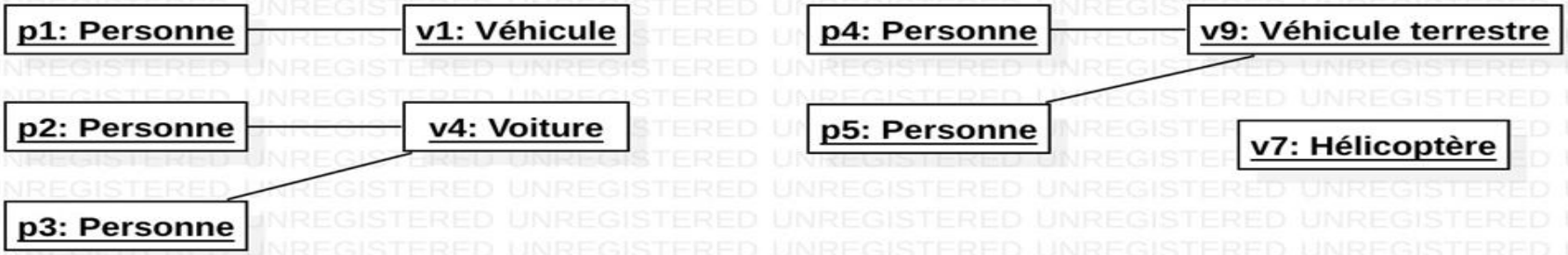


Héritage - Diagramme d’objets



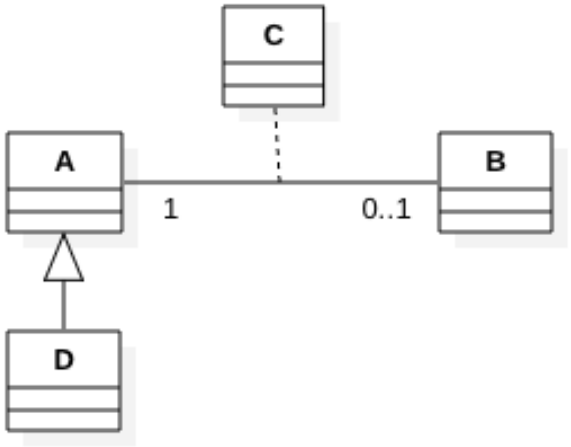
Héritage - Héritage des associations





Héritage - Exercice 1

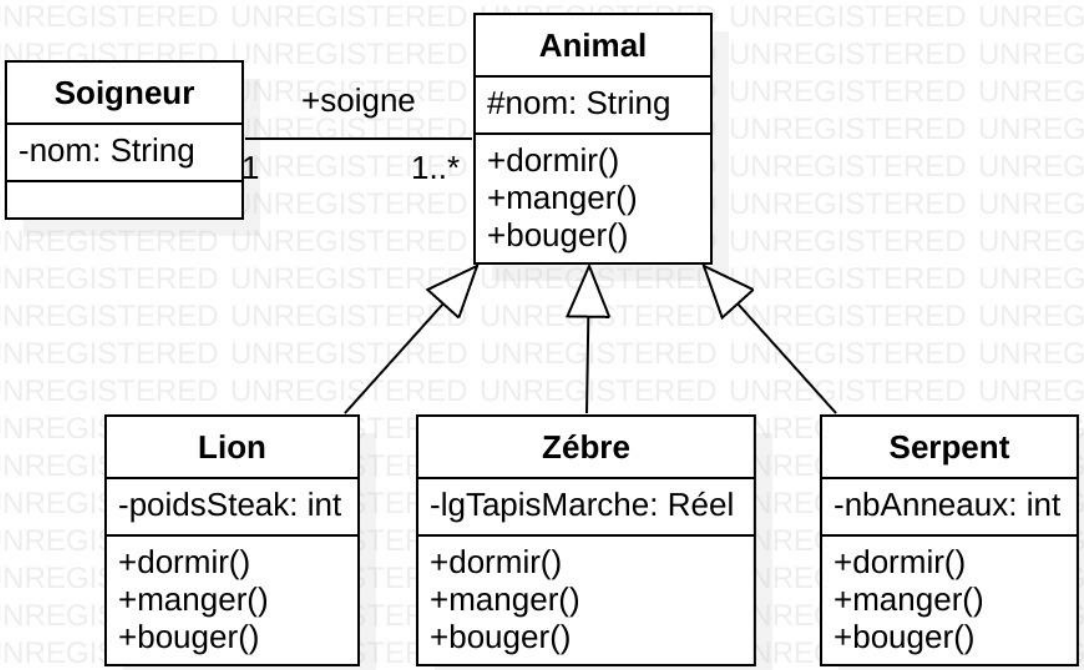
Voici un diagramme de classes. On vous demande de nous produire un diagramme d'objets qui reprend au moins une instance de chaque classe et au moins un lien pour chaque association bidirectionnelle et paire de classes concernées.



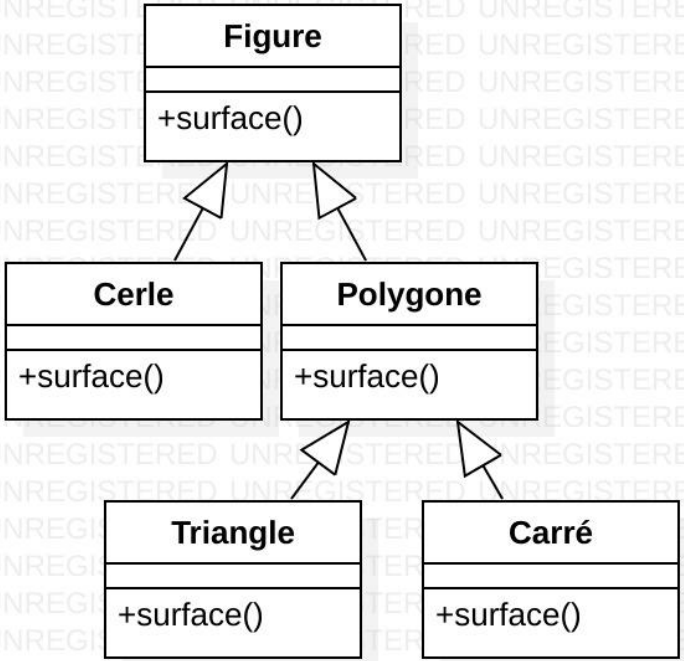
Héritage et polymorphisme

- Certaines opérations décrites au niveau d’une super-classe peuvent être redéfinies au niveau des sous-classes.
- On a donc plusieurs formes (polymorphe) d’une même entité (ici une méthode).
- Il existe du polymorphisme de méthodes, de variables,...

Héritage et polymorphisme

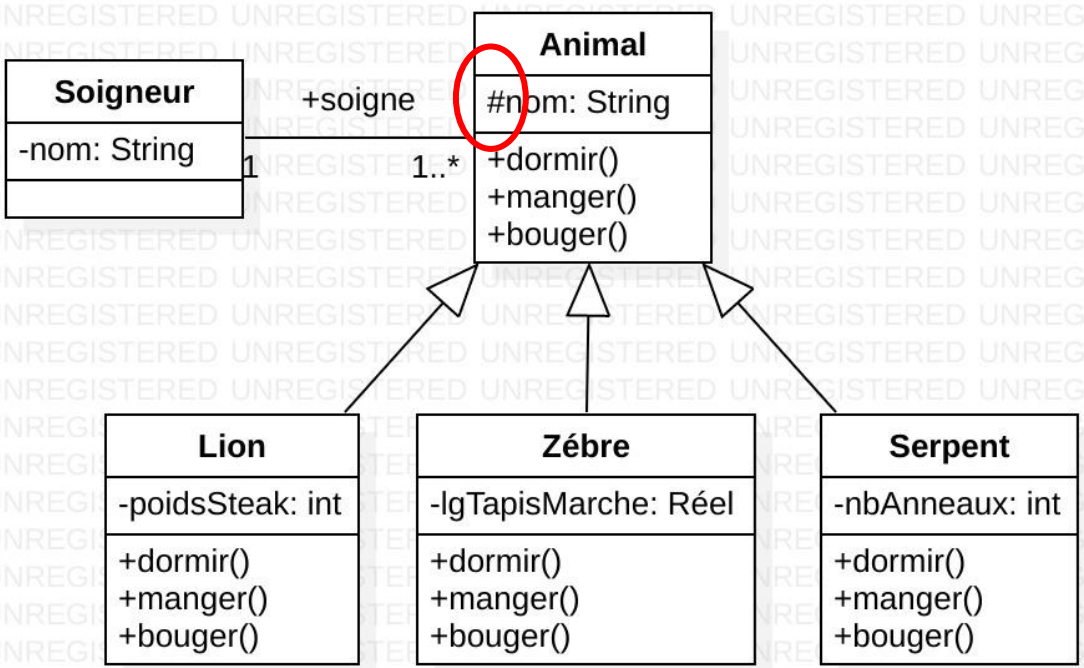


Héritage et polymorphisme



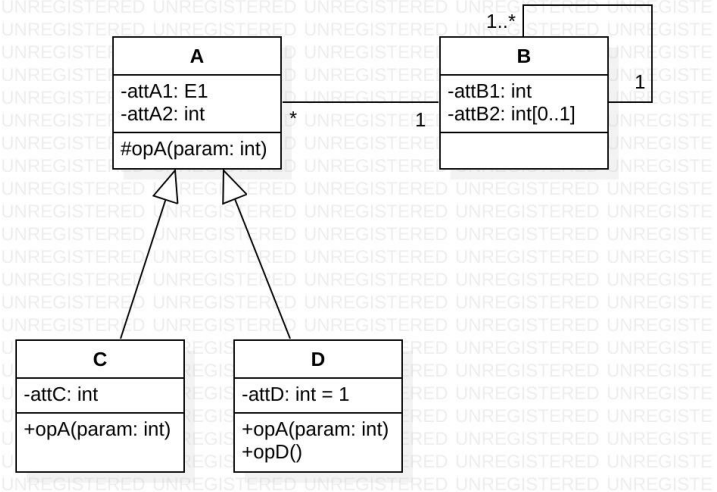
Une sous classe peut redéfinir une méthode, à condition toutefois de rester « compatible » avec la définition originale

Héritage - Visibilité



Héritage - Exercice 2

Voici un diagramme de classes. On vous demande de nous produire un diagramme d'objets qui reprend au moins une instance de chaque classe et au moins un lien pour chaque association bidirectionnelle et paire de classes concernées.

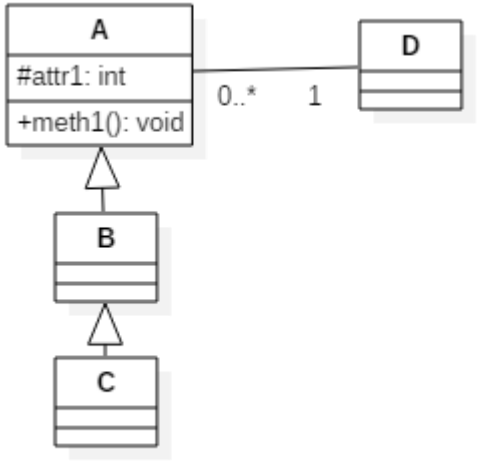


Ecrivez le code Java correspondant.

Héritage: propriétés

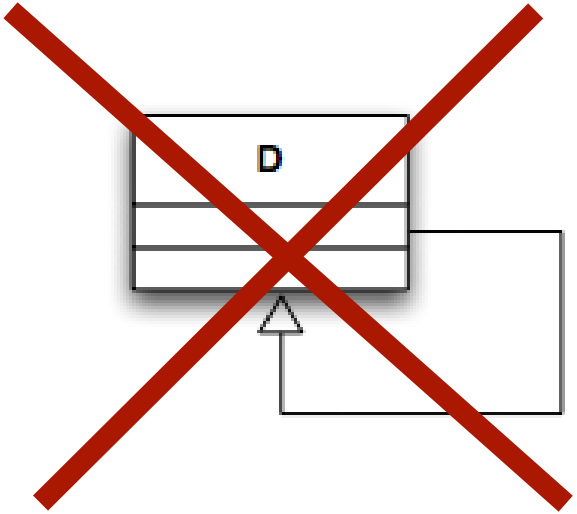
- Relation transitive
  - pour les attributs
  - pour les méthodes
  - pour les associations

- A, B et C ont:
  - un attribut entier protégé *attr1*
  - une méthode *meth1*
  - une association vers D



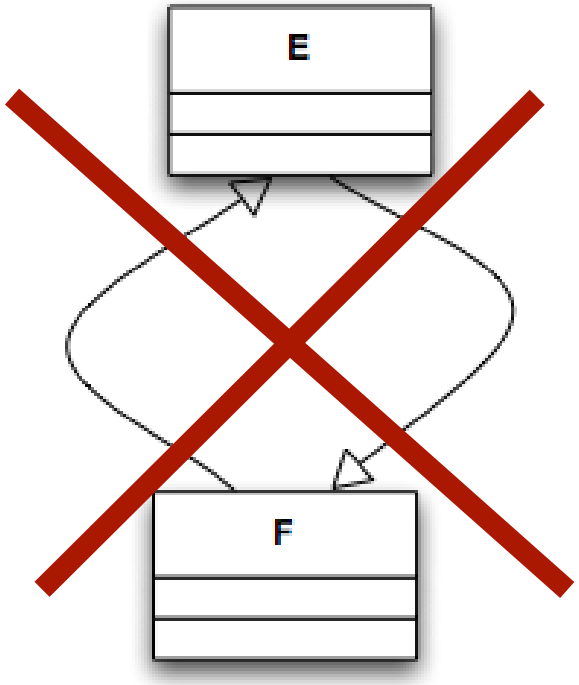
Héritage: propriétés

- Relation non réflexive
  - Une classe ne peut pas hériter d'elle-même.

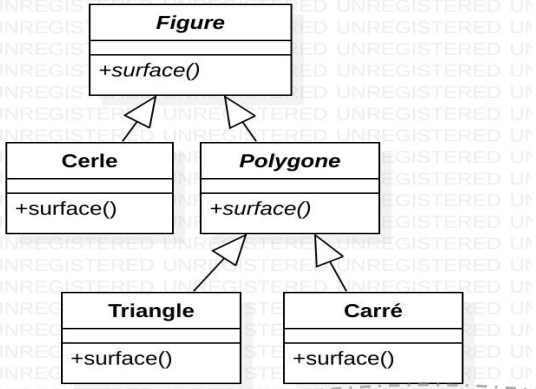


Héritage: propriétés

- Relation non symétrique
  - Si F hérite de E
  - Alors E ne peut pas hériter de F.
- La hiérarchie ne peut pas former de cycle.

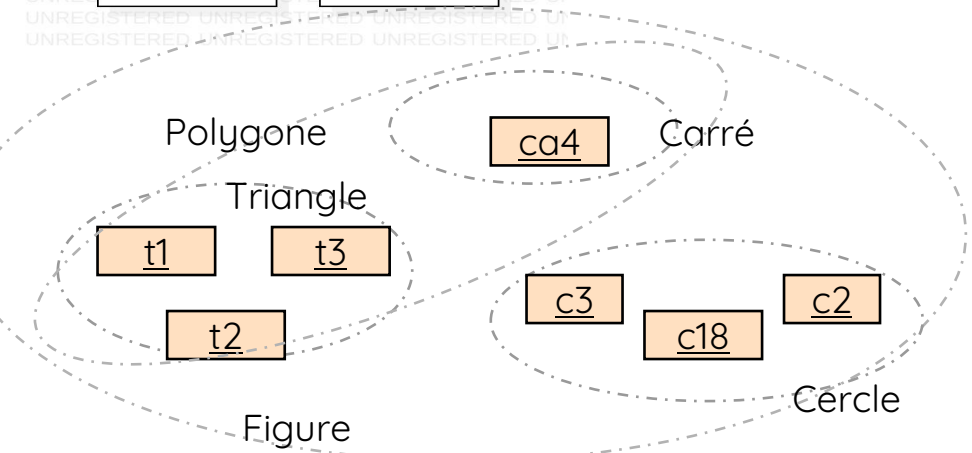


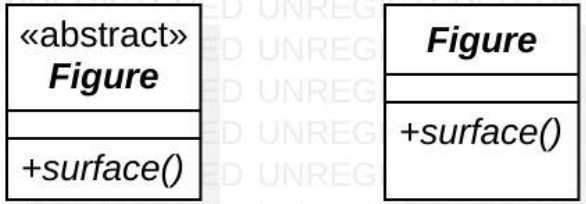
Héritage: classe abstraite



Une classe abstraite

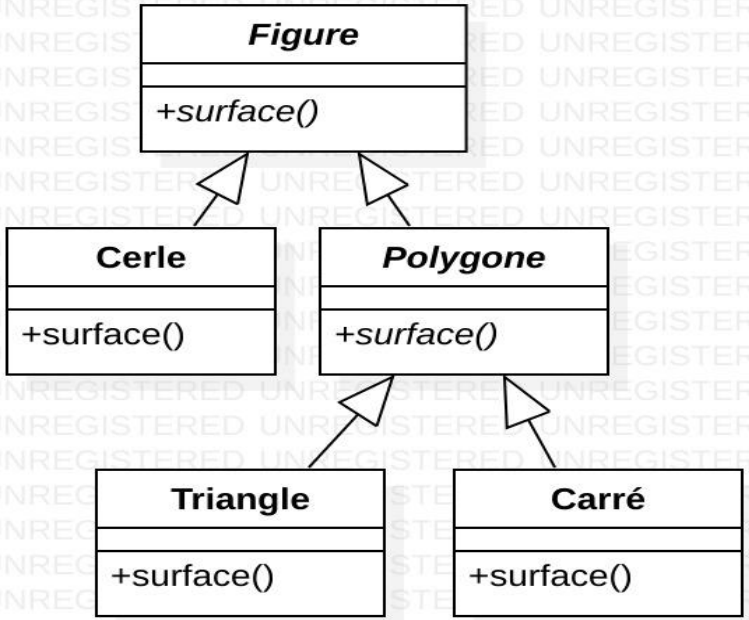
- ne peut pas être instanciée
- utile pour définir un comportement abstrait
- doit être étendue





Notations équivalentes

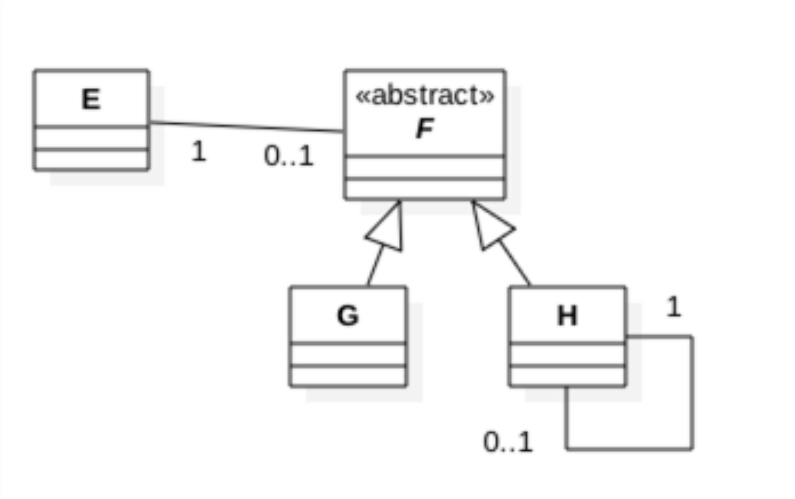
Héritage: méthode abstraite



Une méthode abstraite
- doit être dans une classe abstraite,
- n'a pas de corps,
- doit être redéfinie.

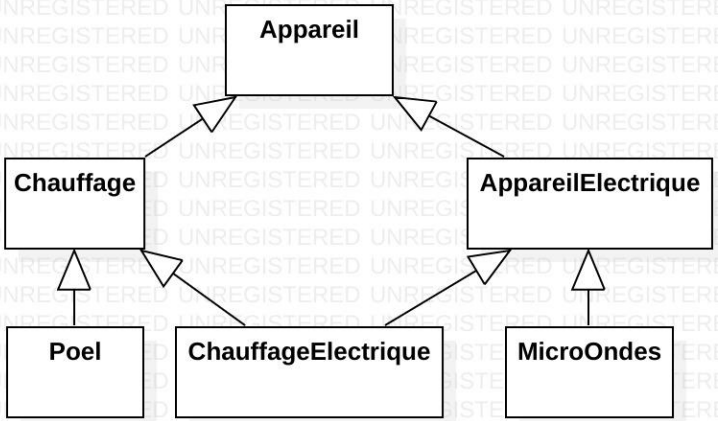
Héritage - Exercice 3

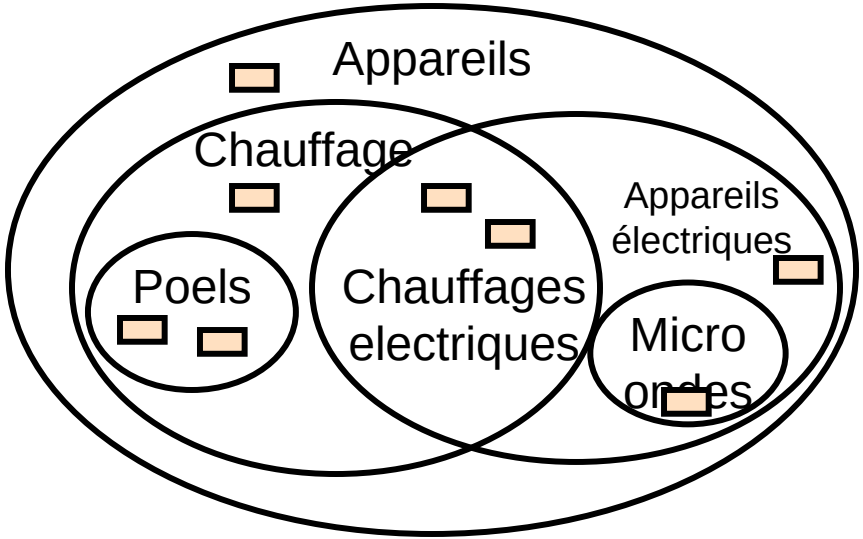
Voici un diagramme de classes. On vous demande de nous produire un diagramme d'objets qui reprend au moins une instance de chaque classe et au moins un lien pour chaque association bidirectionnelle et paire de classes concernées.



Héritage multiple

Une classe peut hériter de plusieurs super-classes





Interdit dans certains langages de programmation (p.e., Java et C#)

Héritage - Pour aller plus loin...

Les modèles orientés-objets ne font pas tous les mêmes hypothèses:

- Héritage simple vs. héritage multiple
  - Une classe peut-elle hériter de plusieurs classes ?
- Classification simple vs. classification multiple
  - Un objet peut-il être simultanément instance de plusieurs classes?
- Classification statique vs. classification dynamique
  - Un objet peut-il changer de classe pendant l'exécution ?

Héritage - Hypothèses UML par défaut

Sauf si le contraire est indiqué explicitement,
en UML les hypothèses par défaut sont :

- Héritage multiple
  - une classe peut hériter de plusieurs classes
- Classification simple
  - un objet est instance d'une seule classe
- Classification statique
  - un objet est créé à partir d'une classe donnée et n'en change pas

Héritage - Références

Ses slides sont inspirés de

- F Pluquet
- http://fr.slideshare.net/megaplanet20
- “The UML user guide” G. Booch, J. Rumbaugh, I. Jacobson
- “Modélisation objet avec UML” P-A Muller, N Gaertner
- “Cours UML” L. Audibert sur developpez.com
- “Java Tête la première” Kathy Sierra & Bert Bates - Ed. O’Reilly