

# Java

## Héritage

University of Geneva  
www.miralab.ch

## Hérarchie des classes

3

On peut organiser les classes sous forme de hiérarchie

En *Java*, une classe peut avoir au plus une classe parente directe

Syntaxe :

```
qualificatif class X extends Y  
{ }
```

la classe X **hérite** de la classe Y

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

# Héritage

4

## Une sous-classe hérite

Des paramètres des classes parentes

Des méthodes des classes parentes (super classes)

## Elle peut

Soit redéfinir des méthodes de la classe parente (overload)

Soit les réutiliser

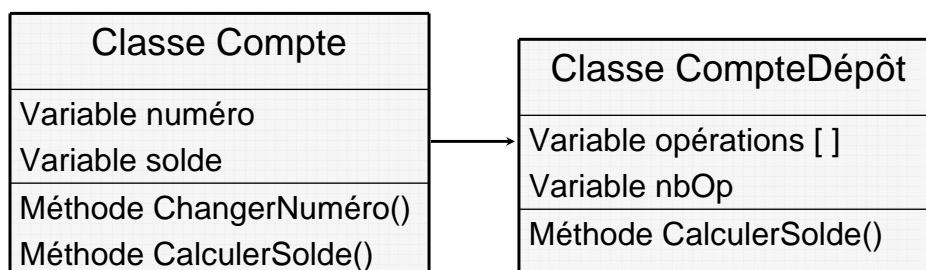
Et ajouter ses propres paramètres et méthodes

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

# Exemple : compte

5



University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

6

```

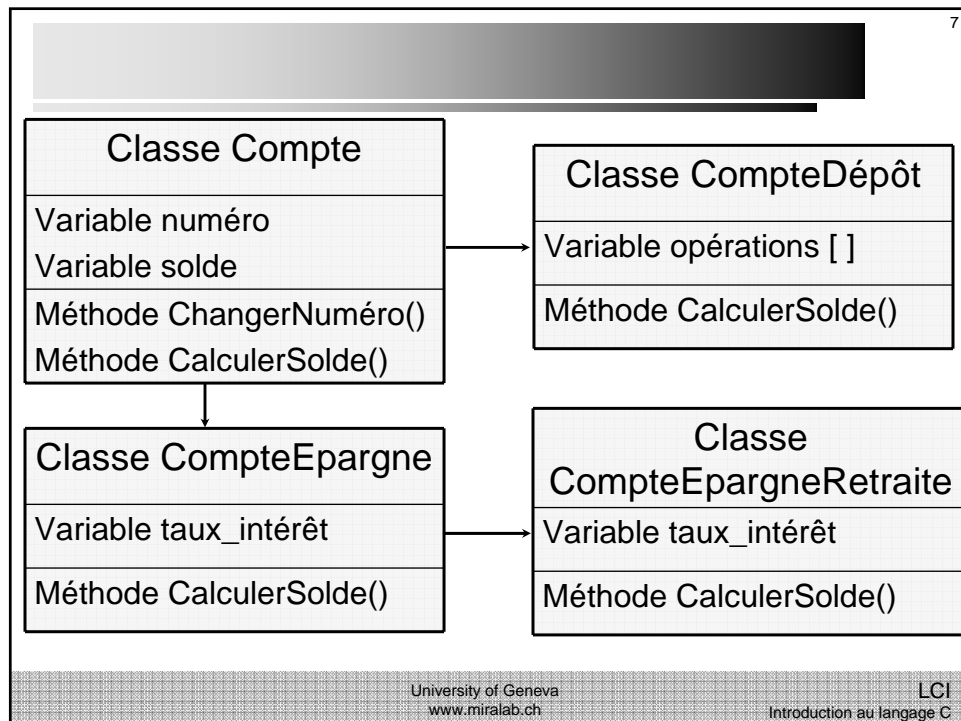
class Compte
{
    private int numéro;
    public float solde;
    public Compte()
    {
        numéro = 0;
        solde = 0.0f;
    }
    public float
    CalculerSolde()
    { return solde; }
    public void
    ChangerNuméro(int n)
    { numéro = n; }
}

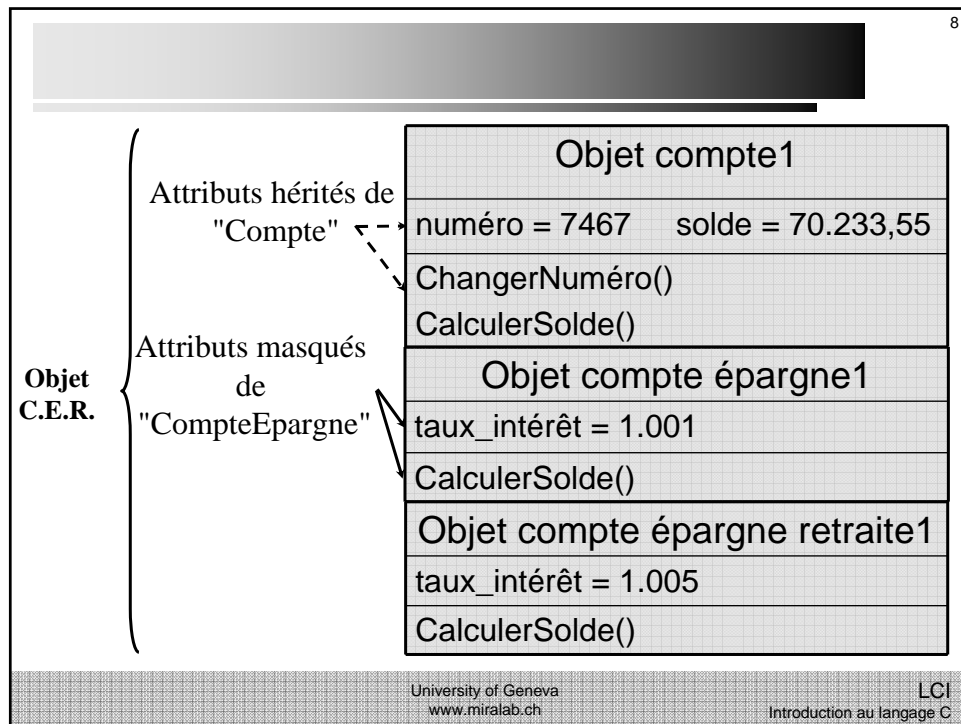
class CompteDépôt extends Compte
{
    public float operations[];
    public int nbOp = 0;
    public CompteDépôt()
    { }
    public float CalculerSolde()
    {
        for(int i = 0; i < nbOp; ++i)
            solde += operations[i];
        nbOp = 0;
        return solde;
    }
}

```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C





## Utilité de l'héritage

Généralement utilisé pour modifier le comportement d'une classe

Ajout de fonctionnalité à une classe de base

Spécialisation

Réutilisation de méthodes communes

Modification d'une classe existante et utilisée ailleurs

Empêchant la modification directe de la classe existante

## Héritage versus composition

10

Ne pas confondre héritage et composition

Héritage se fait sur des classes partageant des mêmes fonctionnalités

une classe *autobus* pourrait hériter de la classe *auto*

réutilisation de méthode *nombreDeRoues()* etc

extension *nombreDePassagerMax* etc

mais *autobus* ne devrait pas hériter d'une classe *roue*

Le lien entre des classes sans rapport se fait par simple composition

une classe *autobus* peut **utiliser** une classe *roue*

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Constructeur & Héritage

11

Lors de la création d'une instance (objet)

les constructeurs de toutes les classes parentes sont appelés

dans l'ordre parent -> enfant

```
class Parent {}
```

```
class Enfant extends Parent {}
```

```
Enfant e = new Enfant();
```

donne lieu à l'appel de *Parent()* ; puis  
exécution du bloc {} du constructeur  
*Enfant()*

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Constructeur

12

On peut rediriger l'appel d'un constructeur vers un autre avec

**this**(*param*, ...) pour un constructeur de la même classe et

**super**(*param*, ...) pour un constructeur de la classe parente (super classe)

au début du corps du constructeur

## Exemple : Entier

13

```
class Entier
{ public int _e;
  Entier(int i)
  { _e = i; }
  Entier()
  { this(0); }
}
```

```
class EntierPos
  extends Entier
{ EntierPos(int i)
  {
    super(i);
    if(i >= 0) _e = 0;
  }
  EntierPos()
  { super(0); }
}
```

## Appel *super()* implicite

14

```

class EntierPos extends Entier
{ EntierPos(int i)
  { //super(); // implicite
    if(i < 0)
      _e = 0;
    else
      _e = i;
  }
  EntierPos()
  { super(); }
}

```

```

EntierPos()
{ }

```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Conséquences

15

Il faut

Soit créer un constructeur sans paramètre dans la super classe

ou utiliser son constructeur par défaut

Soit appeler explicitement *super(params, ...)* dans le(s) constructeur(s) de la classe

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Erreur ...

16

```

class Entier
{ public int _e;
  Entier(int i)
  { _e = i; }
}

class EntierPos
  extends Entier
{ EntierPos(int i)
  { super(i); //OK
  }
  EntierPos()
  { } //erreur : ...
//... pas de constr. Entier();
}

```

## Corrections possibles

17

```

class Entier
{ //...
  //ajouter
  Entier()
  { _e = 0; }
}

class EntierPos
  extends Entier
{ //...
  // ou ajouter
  EntierPos()
  { super(0); }
}

```



## Autre utilisation de super

18

On peut utiliser ce mot clef pour appeler des méthodes de la classe parente directe

```
class Comptepôt extends Compte
{
    //...
    public Comptepôt(int num)
    { super.ChangerNuméro(num);
    }
}
```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Redéfinition de variable

19

Lorsqu'une variable d'une super classe est redéfinie dans une sous classe en respectant exactement son nom elle masque la variable de la super classe

```
class Entier
{ public int _e;
}

class EntierPos
    extends Entier
{ public int _e;
}
```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Qualificatif additionnel

20

### Rappel

*private* seul les méthodes de la classe ont accès à la variable

*public* toutes les méthodes de toutes les classes peuvent accéder à la variable

*protected* toutes les méthodes de toutes les sous classes et des classes du même *package* peuvent accéder à la variable

Est aussi utilisé pour qualifier les méthodes

## Exemple : Qualificatifs

21

```

class Entier
{
    public int e;
    protected int f;
    private int g;
    Entier()
    {
        e = 0; //OK
        f = 1; //OK
        g = 2; //OK
    }
}

class EntierPos extends Entier
{
    EntierPos()
    {
        e = 0; //OK
        f = 1; //OK
        g = 2; //Non!
    }
}

```

## Exemple : Qualificatifs

22

```

class Entier
{public int e;
 protected int f;
 private int g;
 Entier()
 { e = 0; //OK
   f = 1; //OK
   g = 2; //OK
 }
}

class Autre
{
  Autre(Entier ent)
  { ent.e = 0; //OK
    ent.f = 1; //non!
    ent.g = 2; //Non!
  }
}

```

ne dérive pas de

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Redéfinition de méthode

23

Lorsqu'une méthode d'une super classe est redéfinie dans une classe en respectant son nom  
le nombre, l'ordre et les types des paramètres  
elle remplace la méthode de la super classe pour les objets de cette sous-classe

```

class Entier
{ public int a()
  { return 0; }
}

class EntierPos extends Entier
{ public int a()
  { return 1; }
}

```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Qualificatifs de classe

24

### Syntaxe de déclaration d'une classe

```
qualif. class X [extends superX]
{ }
```

#### Les qualificatifs

**final** indique que la classe ne peut pas être dérivée (pas de sous-classe)

**abstract** utilisé pour fournir des méthodes et des champs communs à toutes les classes qui en dérivent

impossible de créer un objet d'une telle classe

**public** désigne la classe correspondant au nom de fichier

une seule par .java

## Autres qualificatifs de méthode

25

### Rappel

```
public protected private static native
synchronized
```

#### En +

**final** indique que la méthode ne peut pas être redéfinie dans une sous-classe

**abstract** définit un modèle de méthode sans corps qui doit être redéfini dans toute sous-classe dérivée

Ceci implique de qualifier la classe qui contient la méthode d'**abstract**

⇒ classes abstraites !

## Utilité des classes abstraites

26

Permet de modéliser des concepts qui n'ont pas de sens en terme d'existence

### Exemple de la nourriture

C'est un concept abstrait, qui prend forme concrète seulement si on prend une instance comme une carotte, une pomme, ...

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Exemple

27

```

abstract class Food
{
    abstract public
    boolean isFruit();
}

class Carrot extends Food
{
    public boolean
        isFruit()
    { return true; }
}

class Apple
    extends Food
{
    public boolean isFruit()
    { return true; }
}

class Chocolate
    extends Food
{
    public boolean isFruit()
    { return false; }
}

```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C