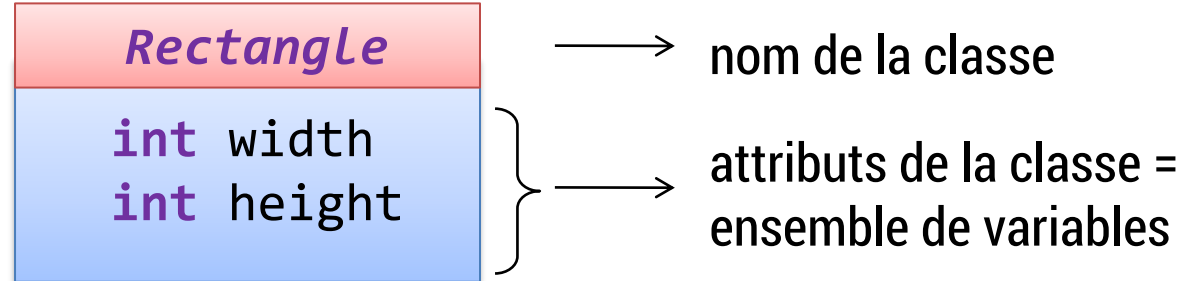


Commençons par le début

## **7.1 CLASSES ET OBJETS**

# Créer une classe : 1<sup>er</sup> exemple

Un rectangle



```
class Rectangle{  
    int width;  
    int height;  
}
```

# Définition de classe

Une classe est un **type généralisé** que l'on peut **définir nous-même**.

Définition

Attributs

- Variables internes

Méthodes

- Fonctions en relation avec la classe

*interface de la classe*

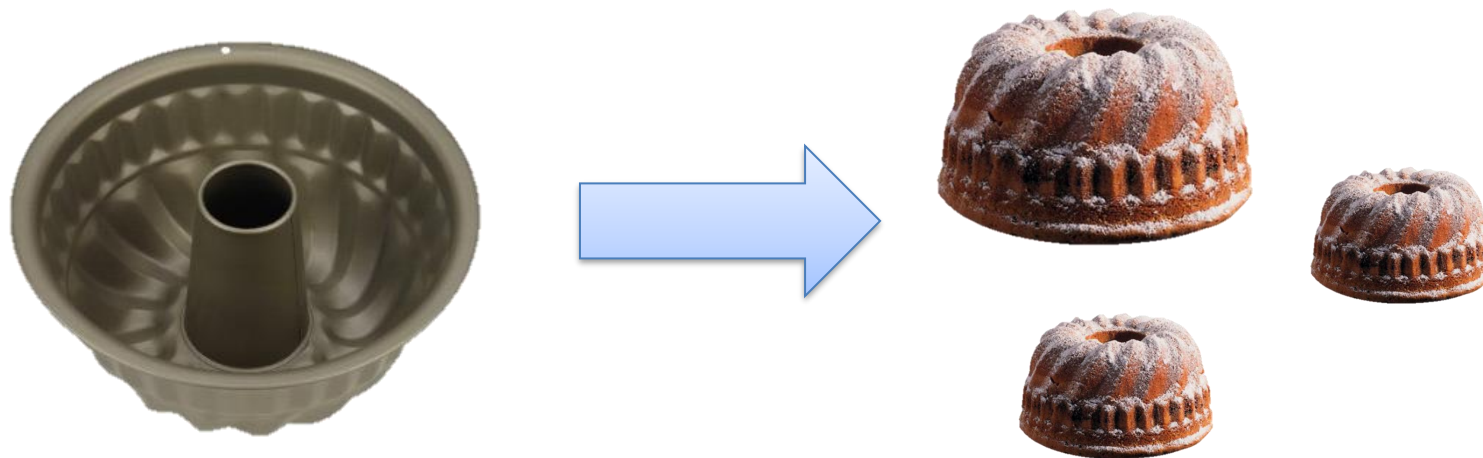
# Implémentation

<i>Rectangle</i>
<code>int</code> width <code>int</code> height
<code>int</code> surface() <code>int</code> perimeter()

```
public class Rectangle{  
    int width;  
    int height;  
  
    int surface(){  
        return width * height;  
    }  
  
    int perimeter(){  
        return 2*width + 2*height;  
    }  
}
```

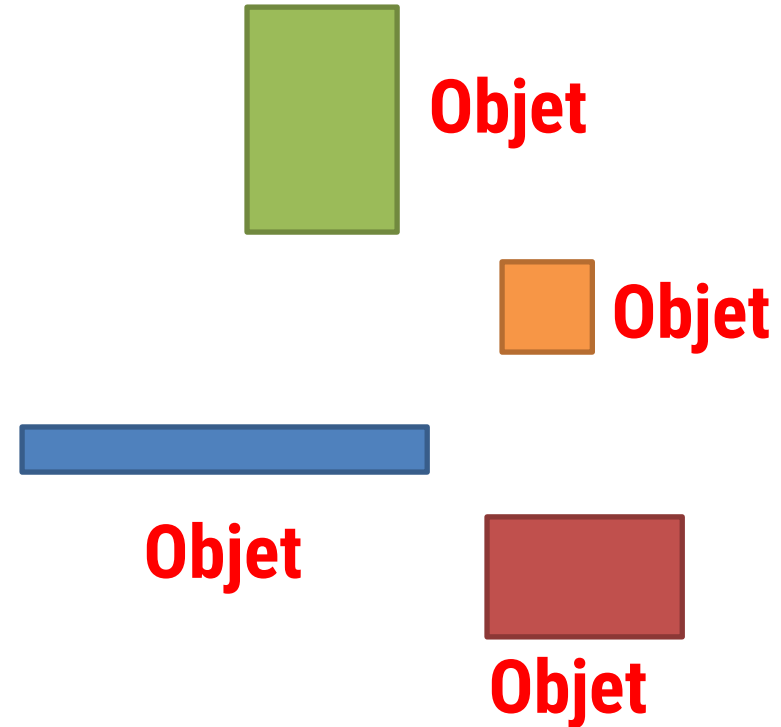
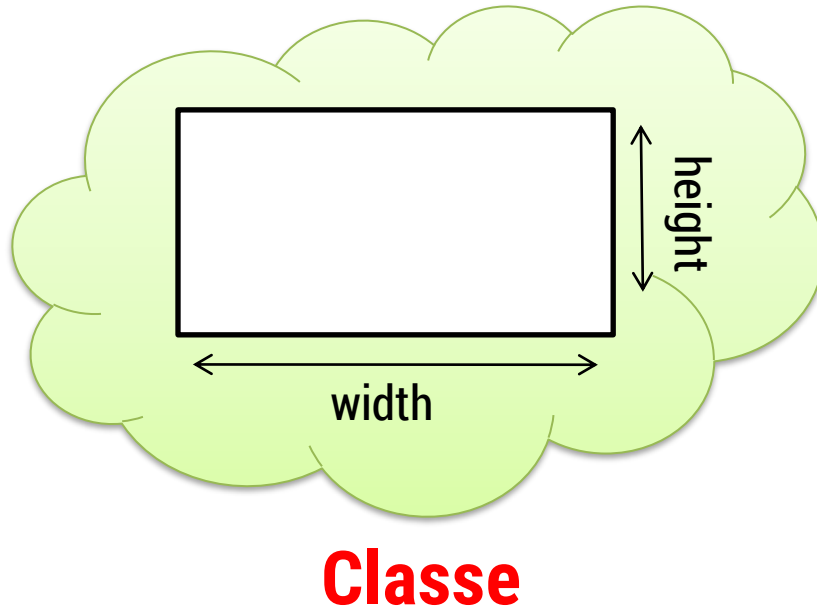
# Un peu de vocabulaire...

- Classe = moule
  - ▶ Une fois moulé (on dit *instancié*), l'objet possède attributs et méthodes du moule.
  - ▶ L'instance de la classe s'appelle un *objet*.
  - ▶ Si change le moule, instances différentes.

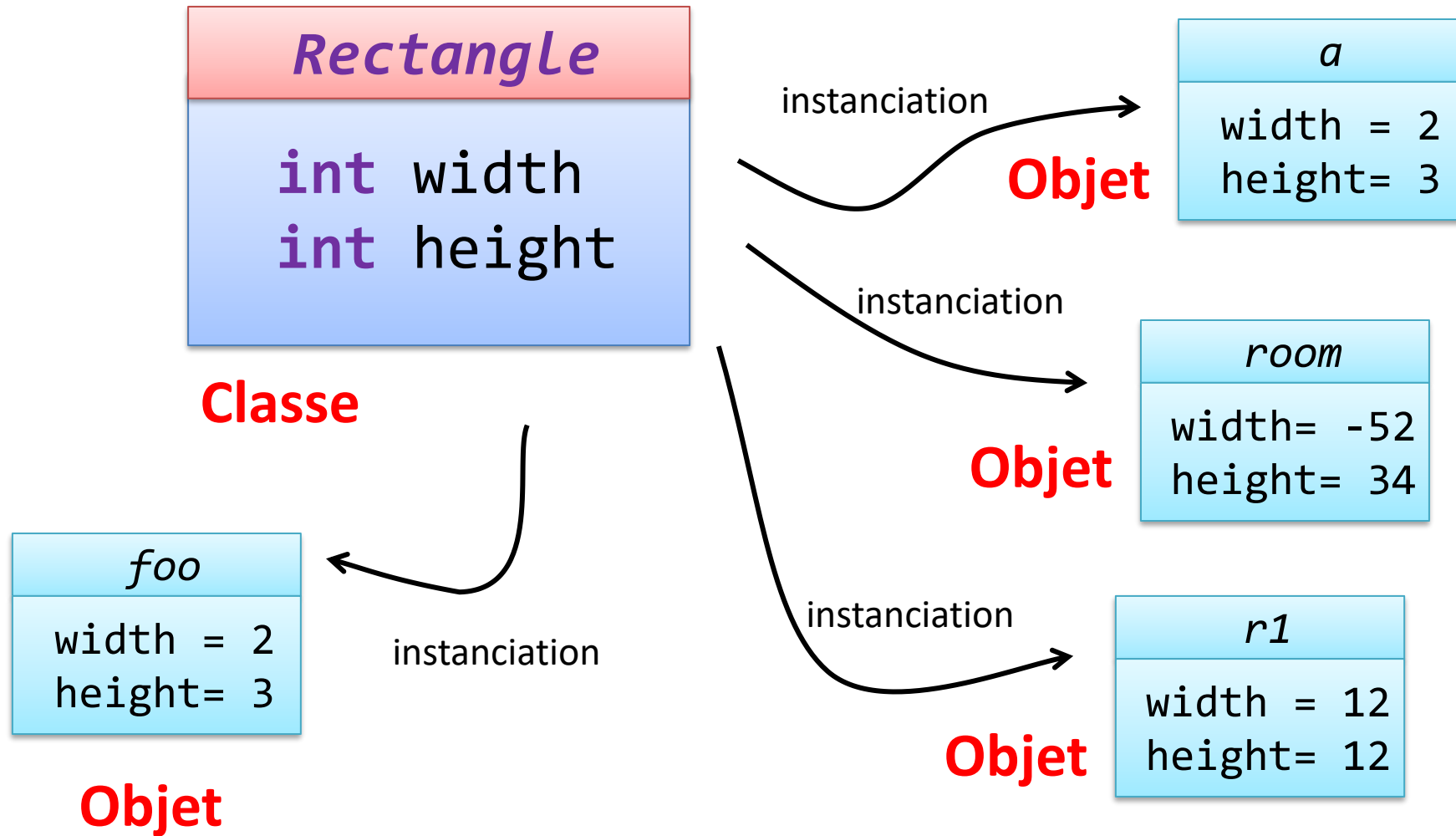


# Classes et objets (1)

- **Classe** Rectangle = le modèle de rectangle
- **Objets** = des rectangles particuliers



# Classes et objets (2)



## Instanciación (I)

- Classe = type abstrait → création objets comme variables.
- Syntaxe similaire

*className identifier*

*Exemple :*

```
Rectangle aRectangle;  
Rectangle bathRoom;
```



## Instanciación (II) – Operateur **new**

- Une fois le type de l'objet prévu, il faut encore le créer.
- La création de l'objet lui-même se nomme l'*instanciation*.
- Pour réserver la place mémoire, on utilise l'opérateur **new**

```
Rectangle room;  
room = new Rectangle();
```

← déclaration

← instanciación

# Accès aux membres

- Pour accéder aux membres → **opérateur '.'**

<i>Rectangle</i>	
<code>int</code>	<code>width</code>
<code>int</code>	<code>height</code>
<code>int</code>	<code>surface()</code>
<code>int</code>	<code>perimeter()</code>

```
Rectangle room;  
room = new Rectangle();  
  
room.witdh = 10;  
room.height = 4;  
int s = room.surface();
```

# Les classes



- Variable = tiroir avec un nom et taille.
- Une classe = une armoire modulaire
  - Peut contenir des *variables*
    - Nombre pas limité
    - Type quelconque (également d'autres classes)
  - Peut contenir des *méthodes*

# Usage des classes

## Abstraction des objets réels

Rectangle

Personne

Auto

...

## Regroupement de méthodes

Math

FunGraphics

...

## Un «mélange» des deux

String

Input

...

- Origine: le langage, d'autres codeurs, nous

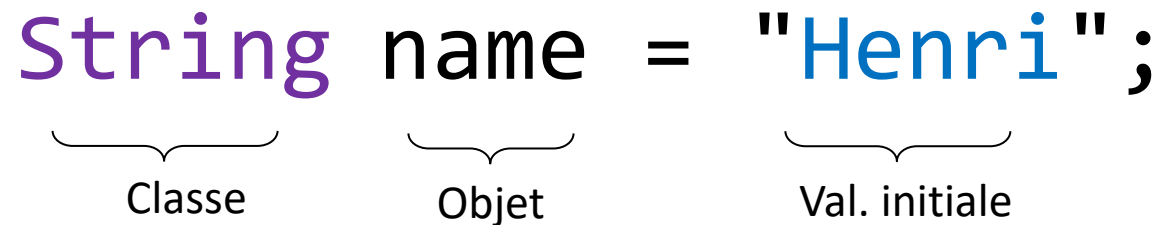
# Rappel (cours 3)

## Pour l'instant

- Une classe = un type
- Un objet = une variable de ce type

- *Exemple* : la classe **String** sert à contenir des chaînes de caractères

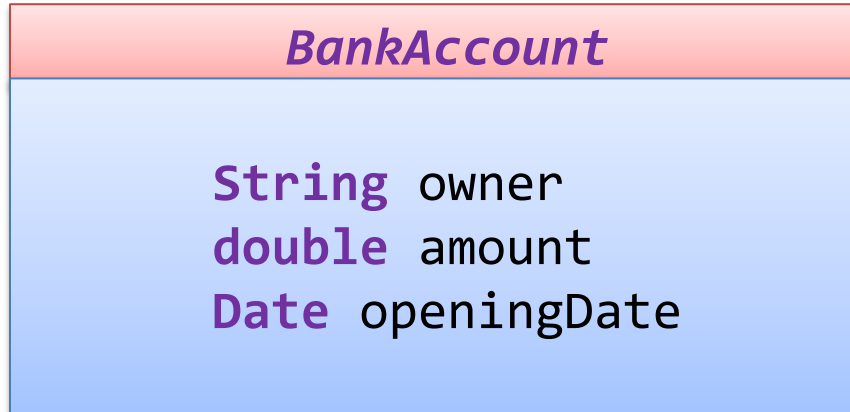
```
String name = "Henri";
```



Classe      Objet      Val. initiale

# Des classes multi-usages

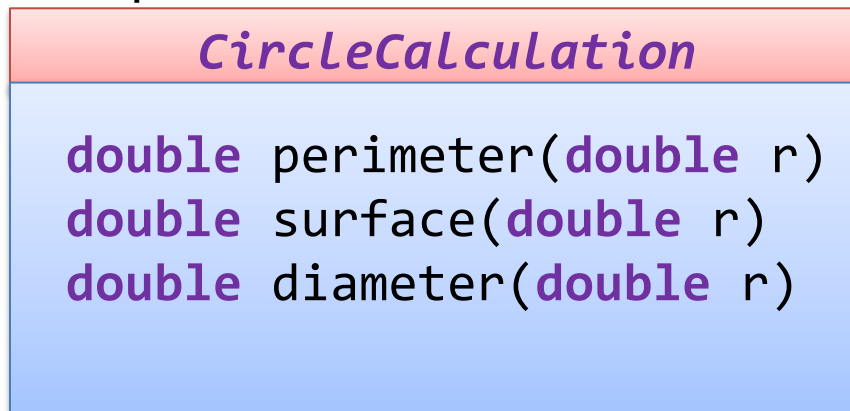
Exemple 1 : variables seules



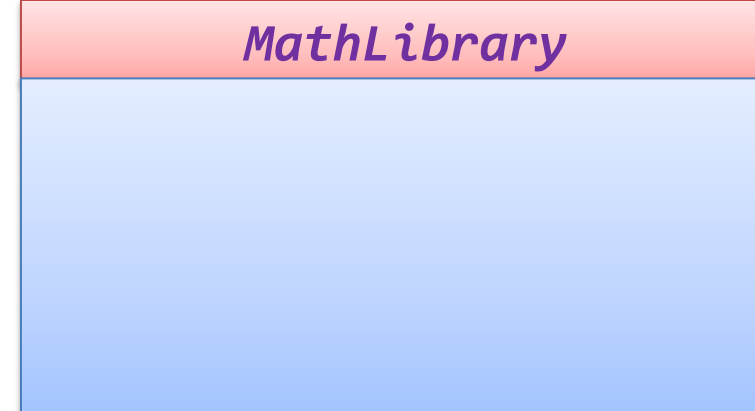
Exemple 3 :



Exemple 2 : méthodes seules



Exemple 4 :



# Deux conventions importantes

- Nom des classes : doit toujours commencer par des majuscules
- Valeurs par défaut : si font du sens
  - Attributs avec un type primitif (`int`, `char`, etc...) → initialisés automatiquement à 0
  - Attributs qui sont des objets → initialisés à `null`