A3P/IPO 2024/2025

Cours 1

Sommaire

1. BlueJ **/!\ dense !**
2. Diagramme des classes
3. Interaction avec les objets
4. Classe / Objet / avantages
5. Attributs (type/valeur, 2 sortes de types)
6. Méthodes (3 sortes) + `public` + `private`
7. **Conventions de nommage**
8. Variables (3 sortes) + `final`
9. Appel de méthode + `this`
10. **A propos du projet**

BlueJ (1/2)

- Environnement interactif,
  mais on peut s'en passer :
  on peut tout faire en ligne de commandes.

- Dans les deux cas :

  - Cercle.java : instructions java

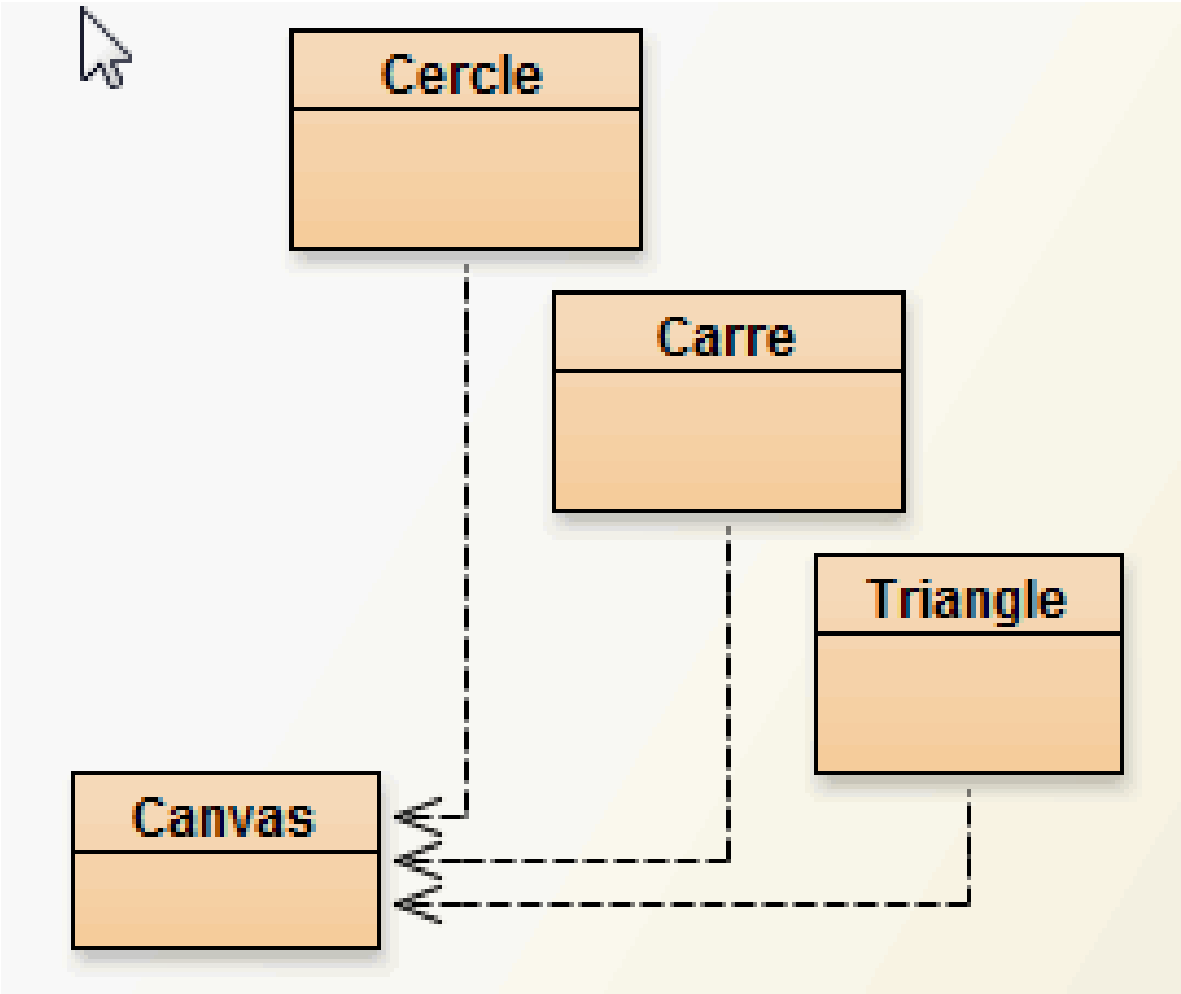
  - Cercle.class : instructions JVM

  Entre les deux : le compilateur java
  (BlueJ utilise le JDK standard)

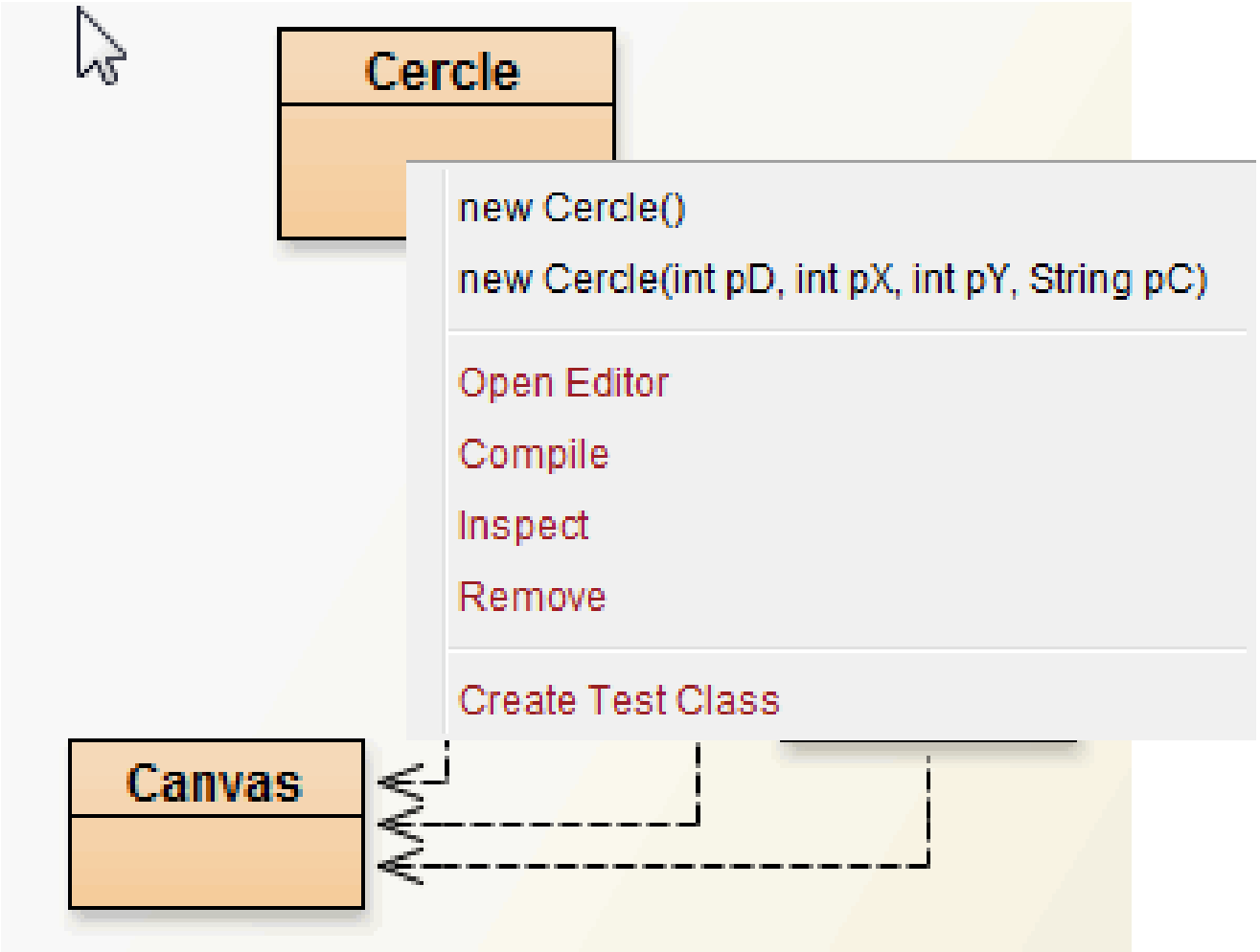
BlueJ (2/2)

- Fournit diagramme de classes, représentation graphique des objets, déclenchement de méthodes sur les objets, affichage du résultat des fonctions.
- Signale les erreurs de compilation, permet l'exécution pas à pas.
- Gère un projet avec plusieurs classes comme un tout (une archive `.jar`).

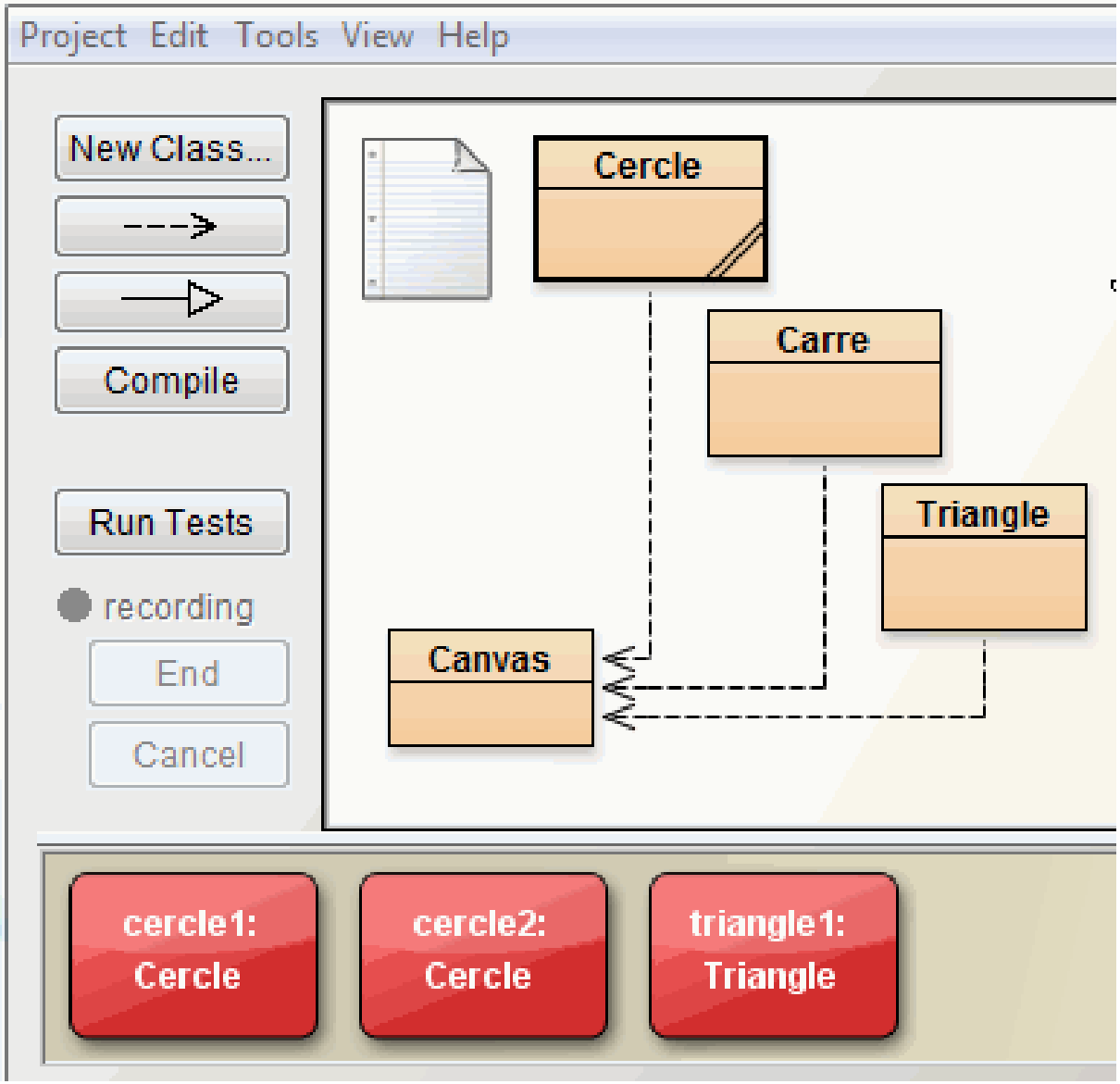
BlueJ : diagramme de classes



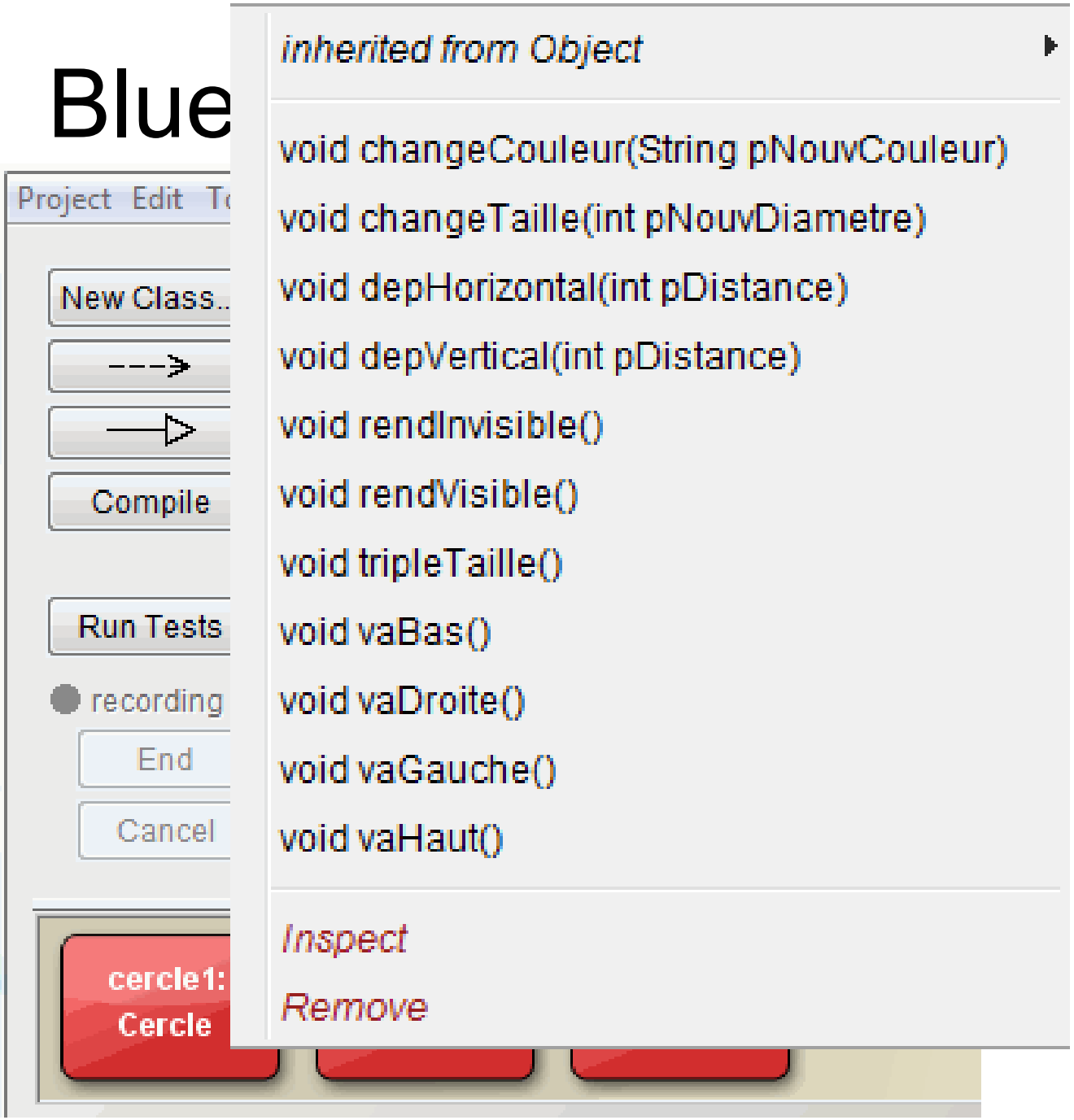
BlueJ : diagramme de classes



BlueJ : objets créés



Blue


Project Edit T
New Class..
Compile
Run Tests
recording
End
Cancel
cercle1:
Cercle
inherited from Object
void changeCouleur(String pNouvCouleur)
void changeTaille(int pNouvDiametre)
void depHorizontal(int pDistance)
void depVertical(int pDistance)
void rendInvisible()
void rendVisible()
void tripleTaille()
void vaBas()
void vaDroite()
void vaGauche()
void vaHaut()
Inspect
Remove

Les classes

- Modèle pour construire des objets
- - attributs (privés)
  caractéristiques de chaque objet
- - méthodes (publiques) [sauf exception**]
  services que peut rendre chaque objet
- Déclaration dans **`NomClasse.java`** :

```
public class NomClasse
{
    attributs et méthodes
} // NomClasse
```

```
* dessine
* efface
```

Exemple de classe

- Modèle pour construire des objets
- - attributs (privés)
  caractéristiques de chaque objet
- - méthodes (publiques) [sauf exception]
  services que peut rendre chaque objet

- Déclaration dans **`Cercle.java`** :

```
public class Cercle
{
    attributs et méthodes
} // Cercle
```

Avantages de l'approche Objet

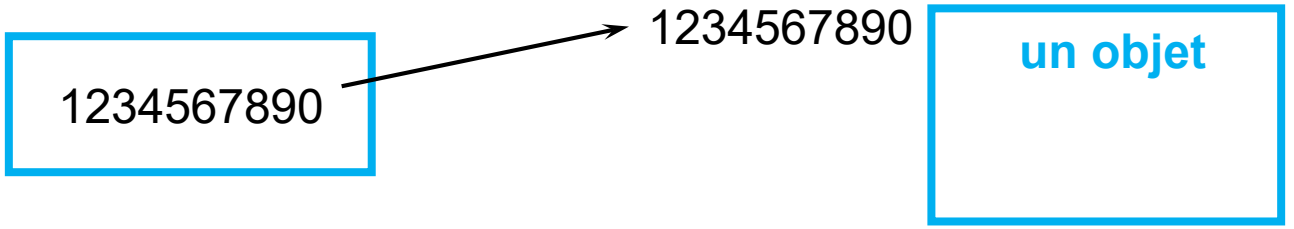
- Tout n'est pas au même niveau, structuration
- Même nom d'attribut/méthode dans +s classes
  private aCouleur, public changeCouleur()
- Si la couleur est incorrecte, laquelle des 10000 lignes incriminer ?
  Ajouter un affichage dans changeCouleur :
- ```
  if ( pNouvCouleur != this.aCouleur )
      affiche( "/!\ chgt de couleur !" );
  pNouvCouleur = this.aCouleur;
  ```
  ```

Les attributs

- Nom et type (+ valeur dans chaque objet)

- - Types primitifs (*contient valeur intéressante*)
  Exemples : `int` et `boolean`
  variable1

- - Types objets (*contient référence vers objet*)
  Exemples : `String` ou `Cercle` ou …
  variable2



Les méthodes : 3 sortes

- **Procédure** *(appel explicite)*
  ne retourne rien => **`void`**
- **Fonction** *(appel explicite)*
  retourne une et une seule valeur
  => préciser son type (*return type*)
- **Constructeur** *(appel automatique)*
  même nom que la classe,
  ni `void` ni type,
  rôle : initialiser tous les attributs

Exemples de méthodes

- **Procédure** *(appel explicite)*
  `void vaBas()`
- **Fonction** *(appel explicite)*
  `int getDiametre()`
  => préciser son type (*return type*)
- **Constructeur** *(appel automatique)*
  `Cercle()`
  ni `void` **ni** type,
  rôle : initialiser les attributs

Attention ! Faux ami

- Le mot « **constructeur** » est trompeur :
  il ne sert pas à construire l'objet,
  mais est appelé une fois celui-ci construit
  pour initialiser tous les attributs.

- Il aurait mieux valu l'appeler « initialiseur »,
  mais c'est ainsi.

- Petit secret : dans l'assembleur de la JVM,
  cette méthode s'appelle **`<init>`**

Conventions de nommage

- classe = commence par une Majuscule
  tout le reste = commence par une minuscule
  une contradiction ?

- mot java = entièrement en minuscules

- autre mot = initialeDeChaqueMotEnMajusculeSaufLaPremière

- variables = commencent par une lettre précise …

Exemples de nommages

- classe = **`Cercle`**, **`Carre`**, **`Triangle`**
  tout le reste = commence par une minuscule
  une contradiction ?

- mot java = **`public`**, **`class`**, **`private`**, **`return`**

- autre mot = **`maVariable`**, **`maMéthode`**

- variables = commenceront par une lettre précise

Les variables : 3 sortes

- Case mémoire : nom, type + valeur
- - attribut :
  caractéristique de tout objet de cette classe
  toute la classe, valeur par défaut
- - paramètre :
  info supplémentaire reçue par une méthode
  dans la méthode, forcément initialisé
- - variable locale :
  case mémoire auxiliaire dans un bloc
  uniquement entre { et }
  éventuellement non initialisée !

Les variables : déclaration

- attribut : *au début de la classe*

```
private type aNom;
```

- paramètre : *dans les parenthèses*

```
final type pNom,
```

`final` optionnel MAIS **prof => obligatoire**

- variable locale : *dans une méthode*

```
type vNom;
```
=> non initialisée !

```
type vNom = valeur;
```
=> initialisée

Exemples de déclarations

- attribut :

```
private int aDiametre;
```

- paramètre :

```
( final int pTaille, … )
```

`final` optionnel MAIS **prof => obligatoire**

- variable locale :

```
int vTmp;
```
=> non initialisée !

```
int vTmp = 1;
```
=> initialisée

Appel de méthode

- `obj.méth( params )`
  appelle la méthode `méth` sur l'objet `obj`
  avec les paramètres `params`

- Si `méth` est une procédure, ajouter un `;` à la fin
  Si `méth` est une fonction,
  1. stocker le résultat dans une variable
  OU 2. utiliser le résultat dans un calcul
  OU 3. afficher le résultat

- Et **comment** appeler une méthode sur le même objet que celui sur lequel elle a été appelée ?

`this`

- désigne l'objet courant, c-à-d l'objet sur lequel a été appelée la méthode qu'on écrit
  (forcément toujours disponible)

- **`this.`** devant attributs et appels de méthodes :
  optionnel MAIS prof => obligatoire

- accès attribut = *objet.aNom*
  accès méthode = *objet.méthode(paramètres)*
  objet = objet courant ou autre objet

A propos du projet

- Vous devez suivre les consignes indiquées dans la séance **Résa 1** et avoir validé la **Charte de l'unité /!**.
- **Réfléchissez au thème de votre jeu :** inventez une histoire et soyez capable de la résumer en une phrase selon le modèle indiqué dans les archives.
- Date limite de saisie : **la semaine prochaine**

Apprendre par cœur
tous les encadrés rouges