Java, les objets : tout de suite !

Cnam Paris
jean-michel Douin, douin@cnam.fr
Version du 17 janvier 2003
http://jfod.cnam.fr/tp_cdi/ et http://jfod.cnam.fr/tp_cdi/douin/

Notes de cours associées a ces deux chapitres

http://www.bluej.org/objects-first/chapters/objects-first-ch1.pdf
http://www.bluej.org/objects-first/chapters/objects-first-ch2.pdf

tutorial BlueJ

http://www.bluej.org/doc/documentation.html

Ce support accompagne, référence le livre de David J. Barnes & Michael Kölling
Objects First with Java A Practical Introduction using BlueJ
Pearson Education, 2003 ISBN 0-13-044929-6.

Sommaire avec BlueJ, extrait Objects first

Chapitres Objects first

- **1) Objets et classes,**
- **2) définitions de classe**

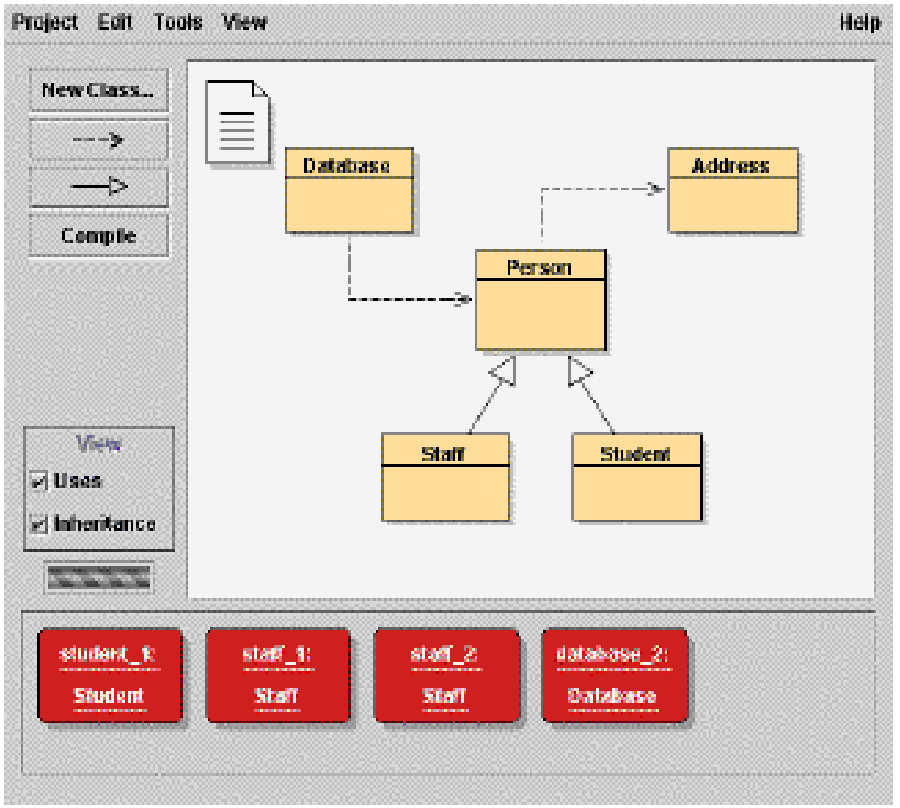
- 3) Interaction entre Objet
- 4) Grouper, réunir plusieurs objets (object)
- 5) Librairies
- 6) Tests, Maintenance,débogage
- 7) Conception de classes

- 8) Héritage
- 9) Polymorphisme
- 10) Couplage faible et ajouts de fonctionnalités
- 11) Gestion des erreurs, exceptions
- 12) Conception d'applications

- Note : BlueJ pour bluejay (ou la java bleue ?)

http://www.bluej.org/what/what.html

- fully integrated environment
- graphical class structure display
- graphical and textual editing
- built-in editor, compiler, virtual machine, debugger, etc.
- easy-to-use interface, ideal for beginners
- interactive object creation
- interactive object calls
- interactive testing (with the J2SE librairies)
- incremental application development



Sommaire

- **Les objectifs des concepteurs du langage Java**
- **Présentation des concepts de l'orienté Objet**

Approche traditionnelle

Impérative, voire syntaxique et ensuite Objet

Approche de la POO en utilisant BlueJ

Approche Objet, graphique et interactive

Java le langage : Bibliographie utilisée

Ce site contient toutes les références et l'environnement BlueJ à installer

- **http://www.bluej.org**

Le cours de M.Kölling

- **http://www.mip.sdu.dk/~mik/teaching/se01**

En français, un site de vulgarisation

- http://www.commentcamarche.net/java/javaintro.php3

Sur le langage Java

- The Java Handbook, Patrick Naughton. Osborne McGraw-Hill.1996.http://www.osborne.com
- http://java.sun.com/docs/books/jls/
- http://java.sun.com/docs/books/tutorial.html
- Program Development in Java,
  Abstraction, Specification, and Object-Oriented Design, B.Liskov avec J. Guttag
  voir http://www.awl.com/cseng/ Addison Wesley 2000. ISBN 0-201-65768-6

Java : les objectifs

- **« Simple »**
  syntaxe " C/C++ "
- **« sûr »**
  pas de pointeurs, vérification du code à l ’exécution et des accès réseau et/ou fichiers
- **Orienté Objet**
  (et seulement !), pas de variables ni de fonctions globales, types primitifs et objet
- **Robuste**
  ramasse miettes, fortement typé, gestion des exceptions
- **Indépendant d ’une architecture**
  Portabilité assurée par la présence d ’un interpréteur de bytecode sur chaque machine
- **Environnement riche**
  Classes pour l ’accès Internet
  classes standard complètes
  fonctions graphiques évoluées
- **Support d’une méthodologie de conception basée sur les « Patterns »**
  Conception Orientée Objet

Simple : syntaxe apparentée C,C++

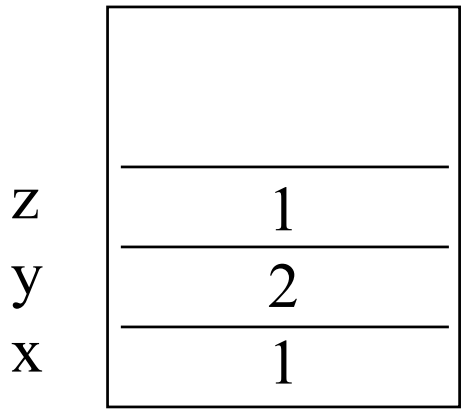
```
public class Num{  // le source : le fichier Num.java


   public int gcd( int n, int d){
      while( n != d)
        if (n > d)
          n = n - d;
        else
          d = d - n;
   }


}
```

Sûr par l 'absence de pointeurs accessibles au programmeur

- **Deux types : primitif ou Object** (et ses dérivés)

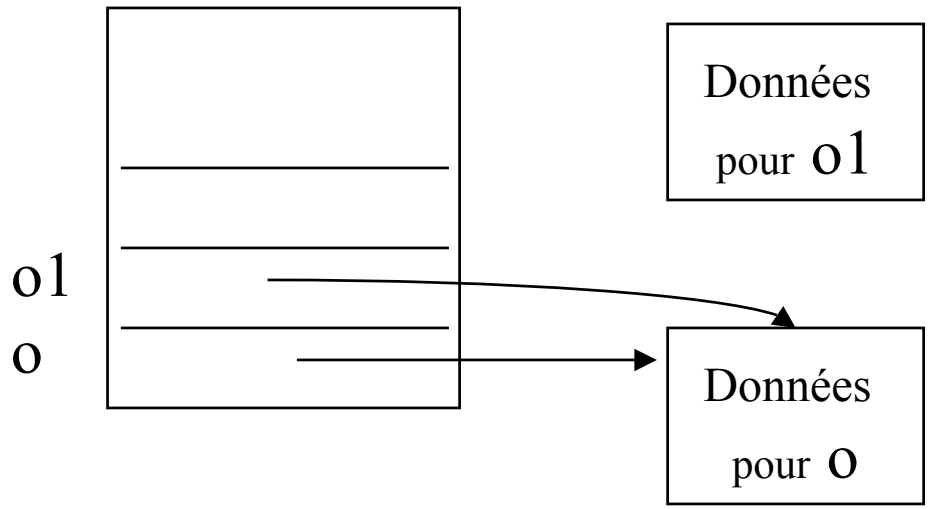
- **primitif :**

```
int   x = 1;
int   y = 2;
int   z = x;
```

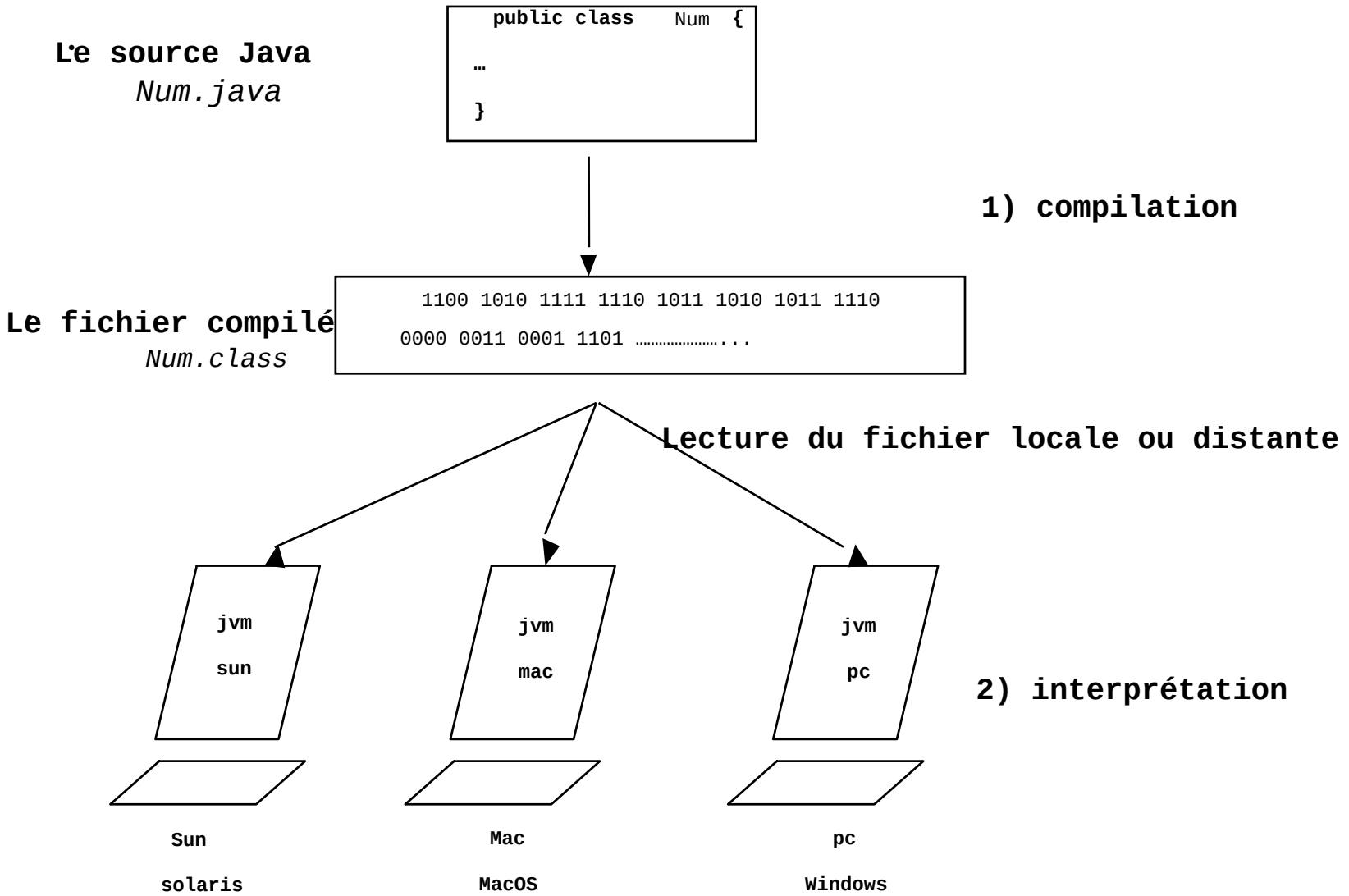


- **Object**

```
Object o = new Object();
Object o1 = new Object();

// seule opération possible
Object o1 = o;
```



Portable


Le source Java
Num.java
public class Num {
…
}
1) compilation
Le fichier compilé
Num.class
1100 1010 1111 1110 1011 1010 1011 1110
0000 0011 0001 1101 ……...
Lecture du fichier locale ou distante
jvm
sun
jvm
mac
jvm
pc
2) interprétation
Sun
solaris
Mac
MacOS
pc
Windows

Environnement (très) riche

- java.applet
- java.awt
- java.beans
- java.io
- java.lang
- java.math
- java.net
- java.rmi
- java.security
- java.sql
- java.text
- java.util
- javax.accessibility
- javax.swing
- org.omg.CORBA
- org.omg.CosNaming

- Liste des principaux paquetages de la plate-forme JDK 1.2

soit environ 1500 classes !!! Et bien d 'autres A.P.I. JSDK, JINI, ...

- le JDK1.3/1850 classes
- JDK 1.4 encore plus, déjà le JDK 1.5 ….

Pattern

- **Catalogue de modèles de conception réutilisables**
- **Assemblage de classes pour un discours plus clair**
- **Un modèle == plusieurs classes == Un nom de Pattern**
- **Les librairies standard utilisent ces Patterns**

  L'API AWT utilise le modèle composite ???

  Les évènements de Java sont dérivés du Pattern Observateur ???

  Etc…

Concepts de l'orienté objet

- **Le vocable Objet :**

- **Classe et objet (instance d ’une classe)**
- **Etat d'un objet et données d ’instance**
- **Comportement d'un objet et méthodes**
  liaison dynamique
- **Composition et interaction des objets**
  Est composé de, est constitué de
- **Héritage**
  Le verbe être
- **Polymorphisme**

Un historique

- **Algorithm + Data Structures = Program**

- **A + d = P** **langage de type pascal**

- **A + D = P** **langage modulaire, Ada, Modula-2**

- **a + D = P** **langage Orienté Objet**

A + D = **P**, un exemple

surface (Triangle t) =

surface (Carré c) =

surface (PolygoneRégulier p) =

....

périmètre(Triangle t) =

périmètre(Carré c) =

périmètre(PolygoneRégulier p) =

....

- *usage : import de la **librairie** de calcul puis*

PolygoneRégulier unPolygone; ***// une variable de type PolygoneRegulier***

...

- **y = périmètre (** unPolygone**)**

A + D = P

type PolygoneRégulier = structure
entier longueurDuCote;
entier nombreDeCotés;
fin_structure;
>>>----------<<<

surface (PolygoneRégulier p) =
périmètre (PolygoneRégulier p) =
…… (PolygoneRégulier p) =

- *usage : import du **module PolygoneRégulier** puis*

  PolygoneRégulier unPolygone; *// une variable de type PolygoneRegulier*
  …

- **y = périmètre (unPolygone)**

classe **PolygoneRégulier** =

entier longueurDuCôté

entier nombreDeCôtés;

surface ( ) =

périmètre ( ) =

……. () =

fin_classe;

- *usage : import de la **classe PolygoneRégulier** puis*

PolygoneRégulier unPolygone; *// une **instance** de la classe PolygoneRegulier*

...

- **y = unPolygone.**périmètre ( )

classe **Carré** hérite_de **PolygoneRégulier** =

surface ( ) = */* redéfinition de surface */*

fin_classe;

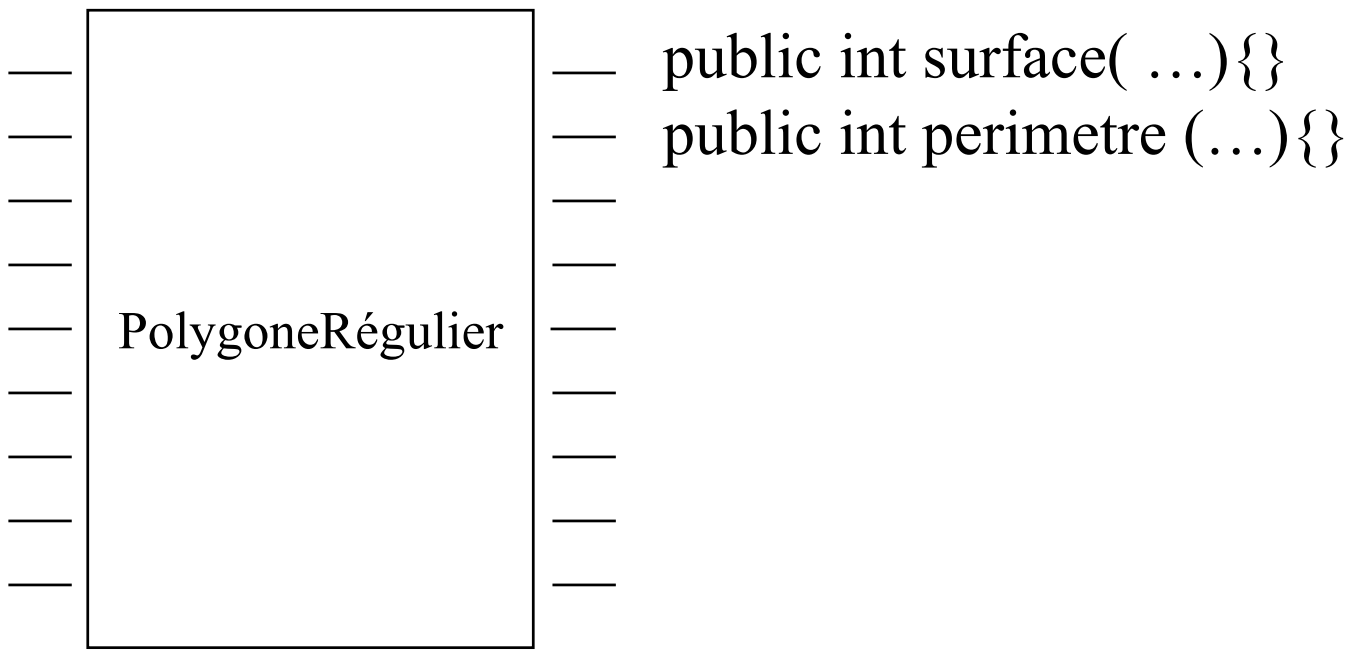
- *usage : import de la **classe carré** puis*

carré unCarré; ***// une instance de la classe Carré***

- **y = unCarré.surface ( )**
- **y = unCarré.perimetre ( )** ***// hérité de PolygoneRégulier***

Encapsulation

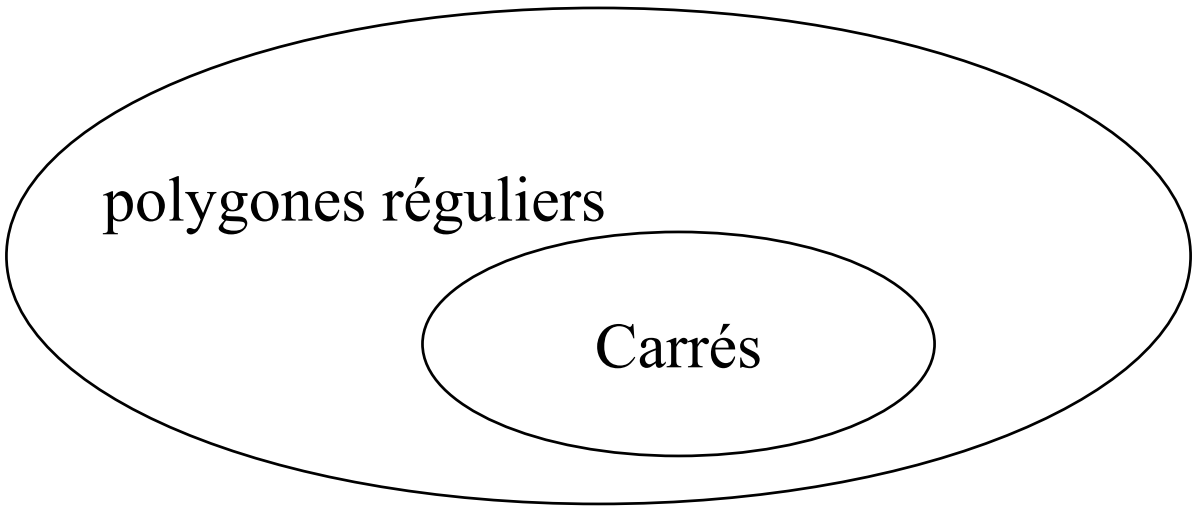
- **contrat avec le client**
  - **interface publique**
  - **implémentation privée, ce sont des choix d'implémenteurs**



Héritage et classification

- **définir une nouvelle classe en ajoutant de nouvelles fonctionnalités à une classe existante**
  - **ajout de nouvelles fonctions**
  - **ajout de nouvelles données**
  - **redéfinition de certaines propriétés héritées (masquage)**

- **Une approche de la classification en langage naturel**
- Les carrés **sont** des polygones réguliers (ce serait l 'idéal...)



Polymorphisme : définitions

- **Polymorphisme ad'hoc**

  Surcharge(overloading),

  plusieurs implémentations d'une méthode en fonction des types de paramètres souhaités, le choix de la méthode est résolu statiquement dès la compilation

- **Polymorphisme d'inclusion** (overridding),

  est fondé sur la relation d'ordre partiel entre les types, relation induite par l'héritage. si le type B est inférieur selon cette relation au type A alors on peut passer un objet de type B à une méthode qui attend un paramètre de type A, le choix de la méthode est résolu dynamiquement en fonction du type de l'objet receveur

- **Polymorphisme paramètrique**

  ou généricité,

  consiste à définir un modèle de procédure, ensuite incarné ou instancié avec différents types, ce choix est résolu statiquement

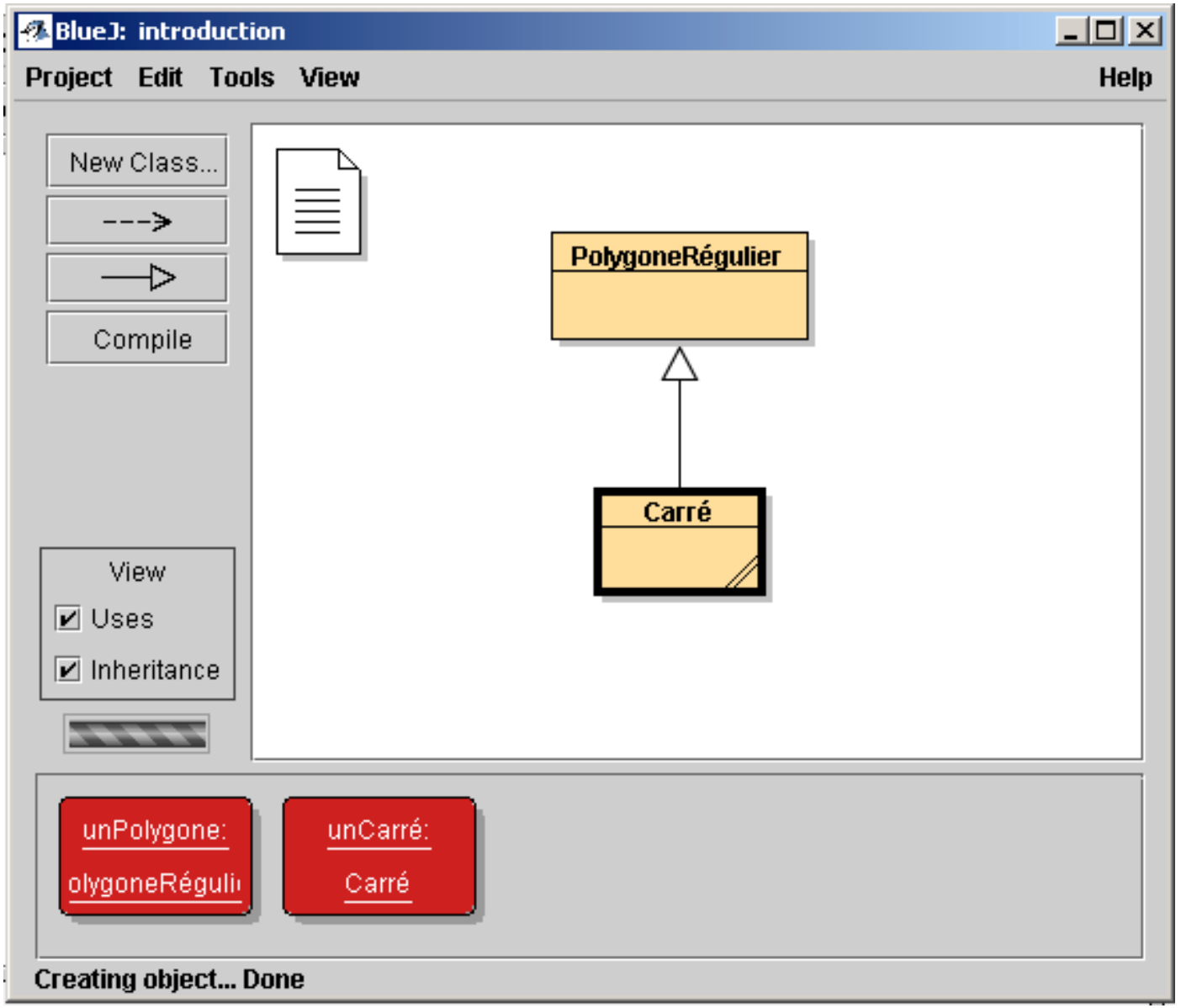
  extrait de M Baudouin-Lafon. La Programmation Orientée Objet. ed. Armand Colin

Le premier exemple en syntaxe Java

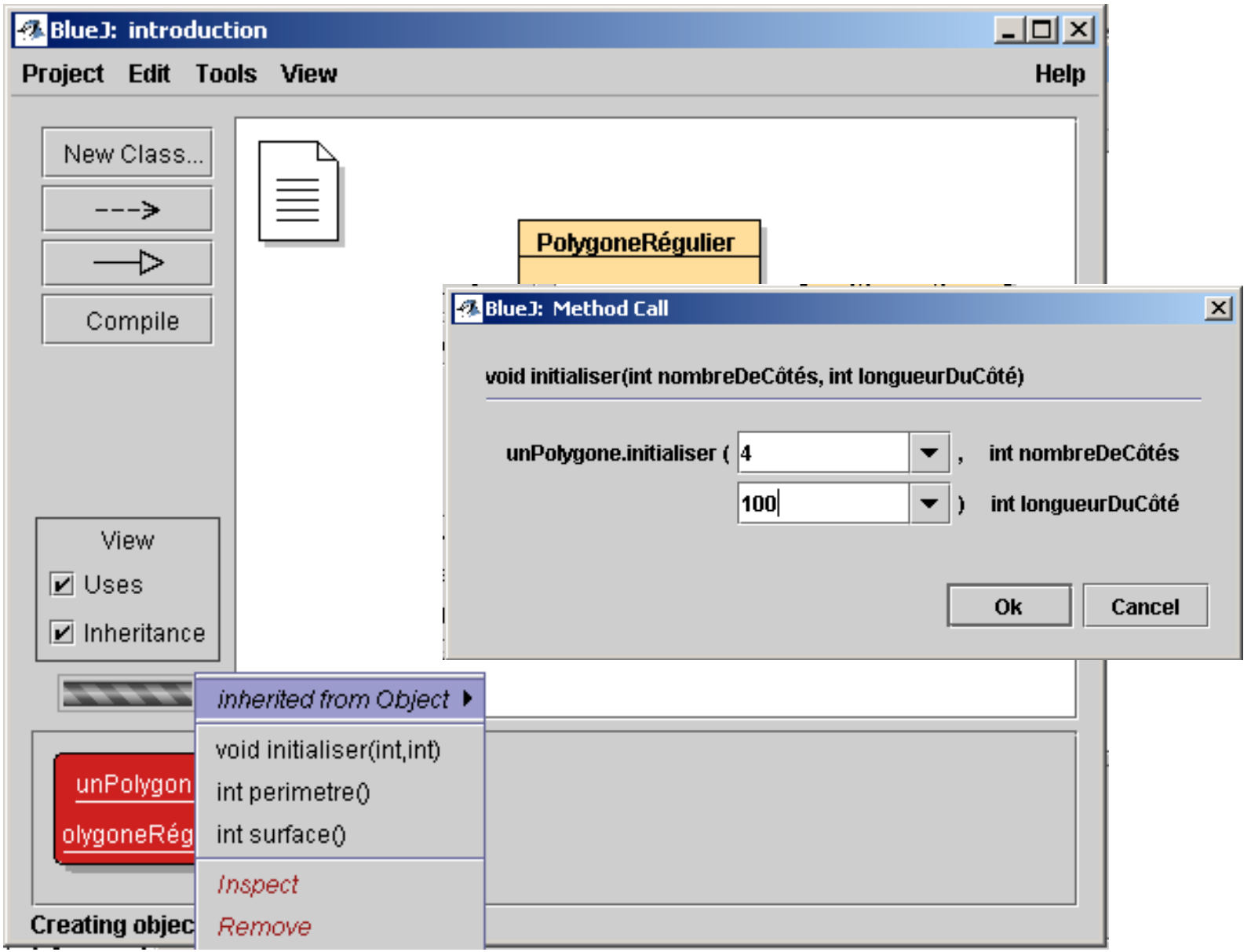
```
public class PolygoneRégulier{
  int longueurDuCôté;
   int nombreDeCôtés;


  void initialiser(int nombre, int longueur){
    longueurDuCôté = longueur;
     nombreDeCôtés = nombre;
 }

 int périmètre(){
   return longueurDuCôté * nombreDeCôtés ;
 }

 public int surface(){ …..}

}
```

```
// un usage de cette classe
PolygoneRégulier unPolygone = new PolygoneRégulier();
unPolygone.initialiser(4,100);
int y = unPolygone.périmètre();
```

Avec BlueJ, Classes et instances



Avec BlueJ, unPolygone.initialiser(4,100)



BlueJ

- **Démonstration, utilisation**

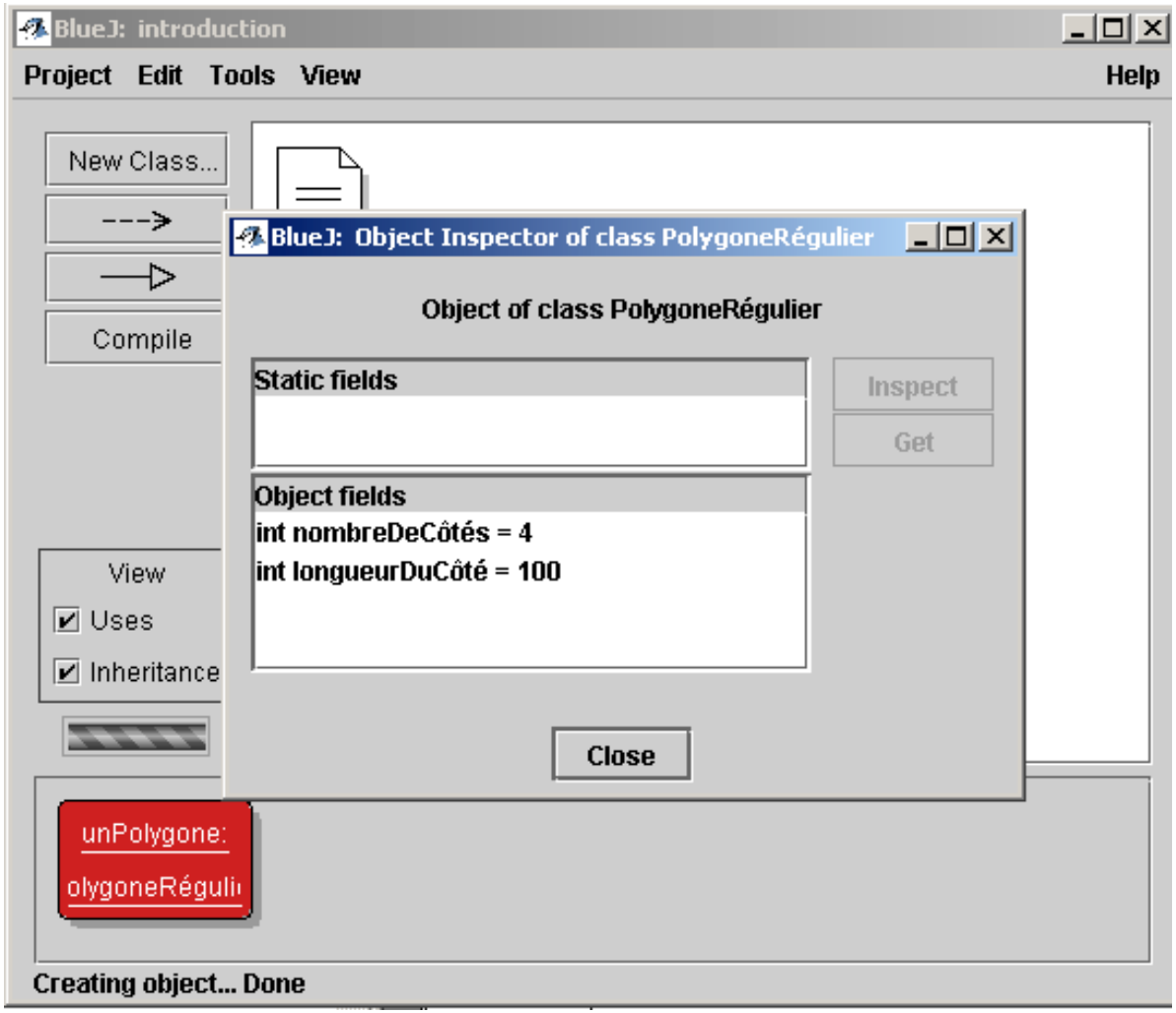
  tutorial page 8

- **Tutorial référencé**

  http://www.bluej.org/tutorial/tutorial.pdf

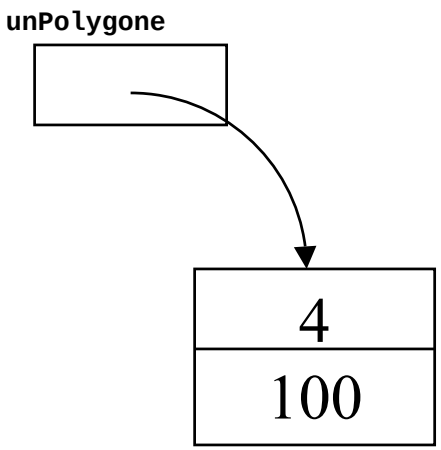
  existe en français http://www.bluej.org/doc/documentation.html

Objet, instance : unPolygone

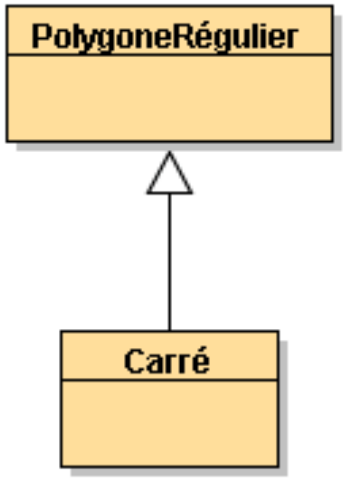


Objet et référence

- **`PolygoneRégulier unPolygone = new PolygoneRégulier();`**
- **`unPolygone.initialiser(4,100);`**



Classe (1)

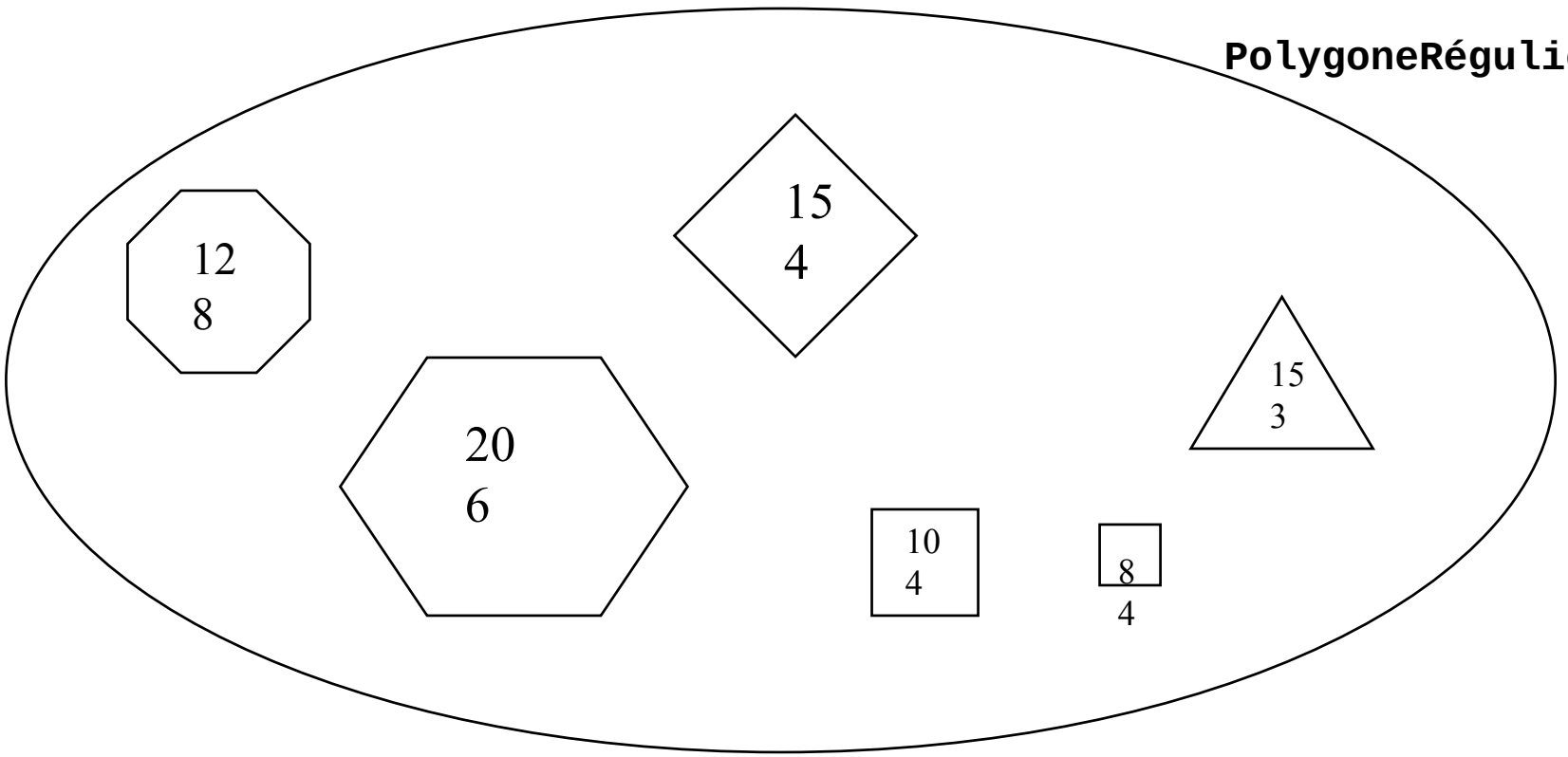



```
public class PolygoneRégulier{

}
public class Carré extends PolygoneRégulier{

}
```

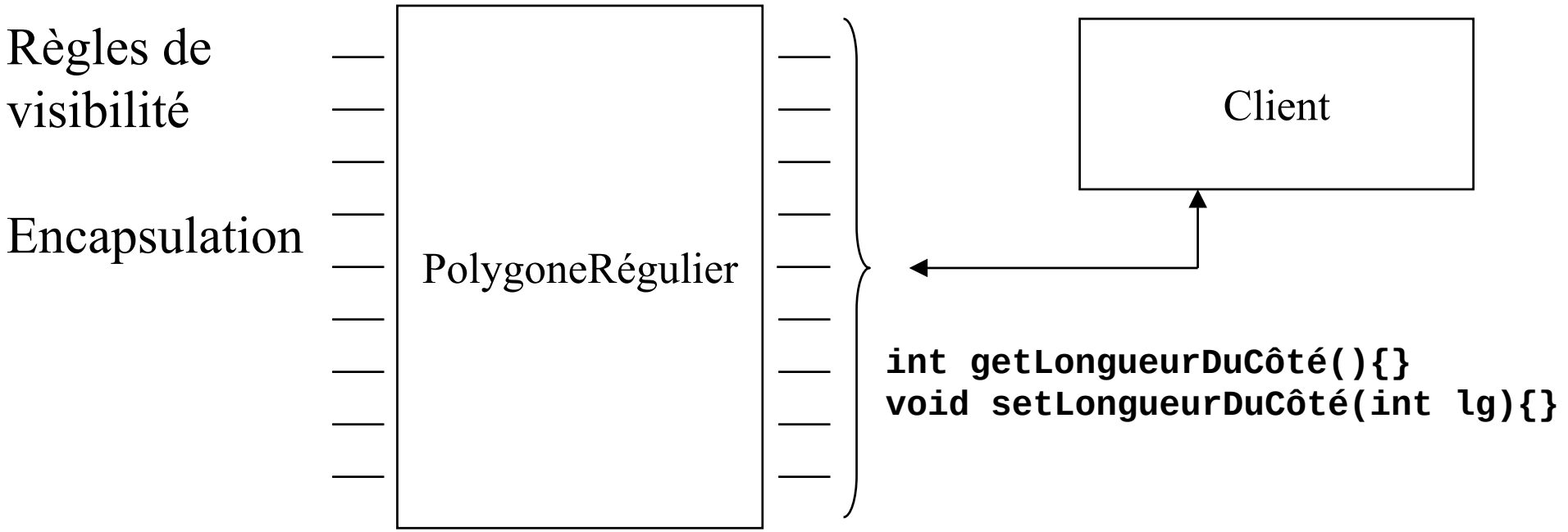
Champs d ’instance, état

```
public class PolygoneRégulier{
   int longueurDuCôté;
   int nombreDeCôtés;
```

- **Valeurs des champs de l ’instance : son état**



Champs d ’instances privées



Règle : Les champs sont privés et accessibles par des méthodes

```
public class PolygoneRégulier{
   private int nombreDeCôtés;
   private int longueurDuCôté;
```

Méthodes/signatures

```
public class PolygoneRégulier{
  private int longueurDuCôté;
  private int nombreDeCôtés;

  public void initialiser(int nombre, int longueur){
      longueurDuCôté = longueur;
      nombreDeCôtés = nombre;
  }


 public int périmètre(){
  return longueurDuCôté * nombreDeCôtés ;
  }

}
```

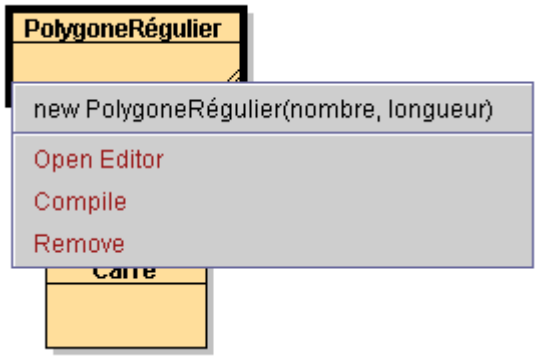
Constructeur

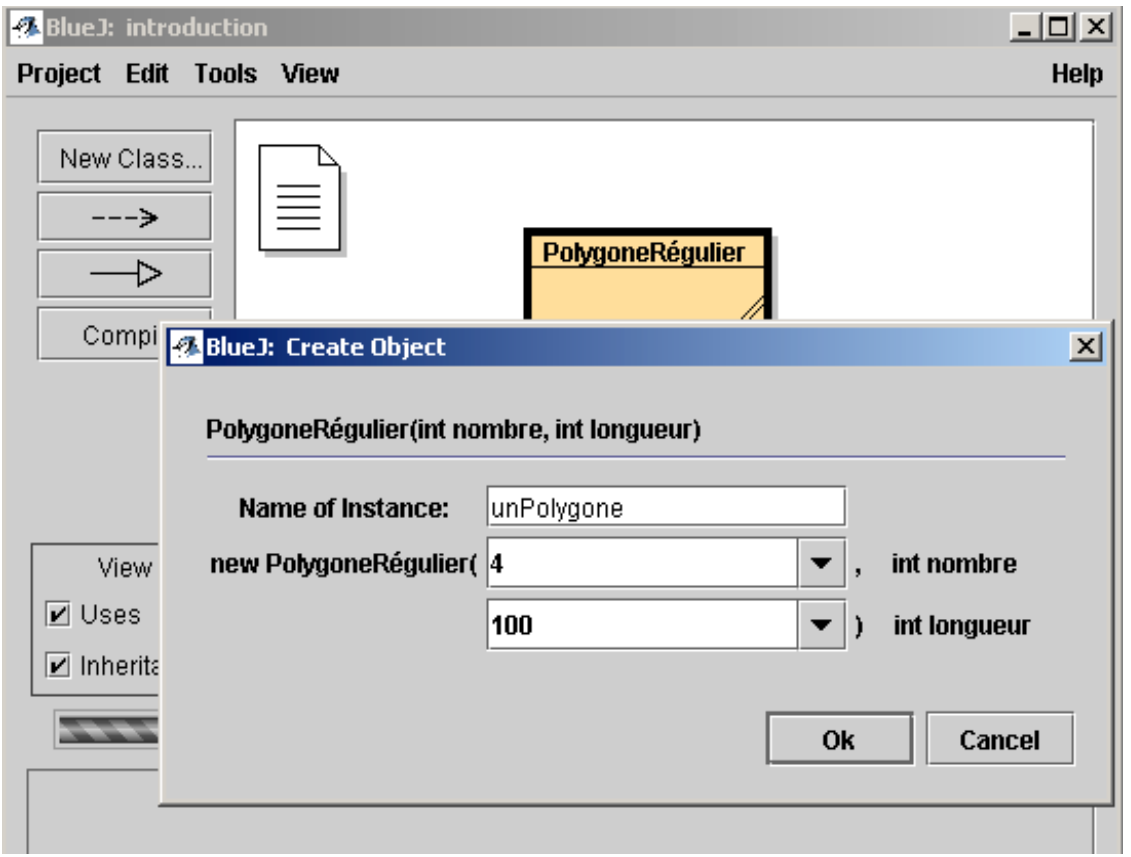
PolygoneRégulier unPolygone **= new PolygoneRégulier();**
unPolygone.initialiser(4,100)

- **en plus concis, plus naturel ...**
  PolygoneRégulier unPolygone **= new PolygoneRégulier(4,100);**

```
public class PolygoneRégulier{
   private int longueurDuCôté;
   private int nombreDeCôtés;

   public PolygoneRégulier(int nombre, int longueur){
       longueurDuCôté = longueur;
       nombreDeCôtés = nombre;
   }
```

Constructeur et BlueJ





Type de données

- **Types primitifs**

```
int x, y;
boolean b = false;
double d = 3.14159;
```

ou

- **« Object »**

```
PolygoneRégulier unPolygone = new PolygoneRégulier();
Carré unCarré = new Carré();

Object o = new Object();
```

Méthodes de modification et d'interrogation

- **Accesseurs : Méthodes de lecture des champs**
- **Mutateurs : Méthodes de modification des champs**
- **Constructeur**
- **Autres méthodes**

  lecture, ou modification

Méthodes « accesseur »

- Par convention public *type* **get***NomDuChamps*

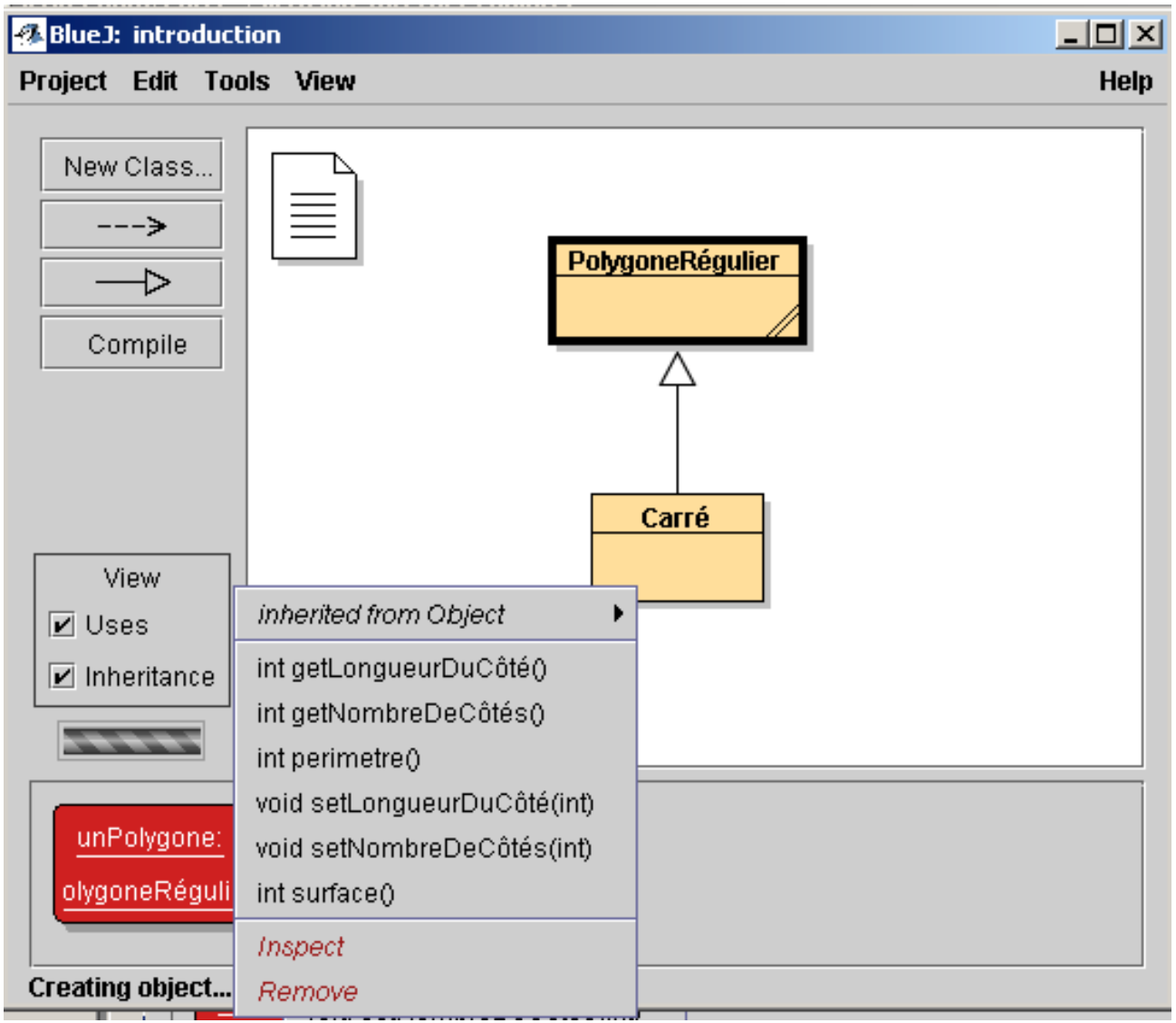
```
public class PolygoneRégulier{
   private int longueurDuCôté;
   private int nombreDeCôtés;

   public int getNombreDeCôtés(){
     return nombreDeCôtés;
   }

   public int getLongueurDuCôté(){
     return longueurDuCôté;
   }
```

Méthodes « mutateur »

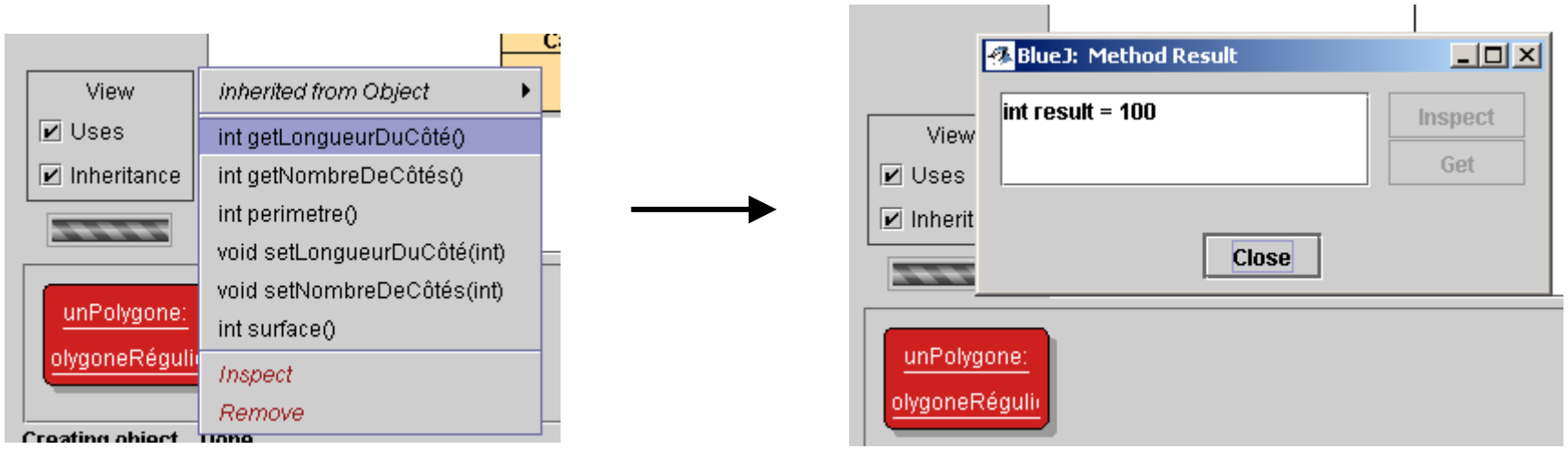
- Par convention public *void* **set***NomDuChamps(type id)*

```
public class PolygoneRégulier{
   private int longueurDuCôté;
   private int nombreDeCôtés;

   public void setLongueurDuCôté(int longueur){
      longueurDuCôté = longueur;
   }

   public void setNombreDeCôtés(int nombre){
      nombreDeCôtés = nombre;
   }
```

Les méthodes accessibles



Valeurs retournées

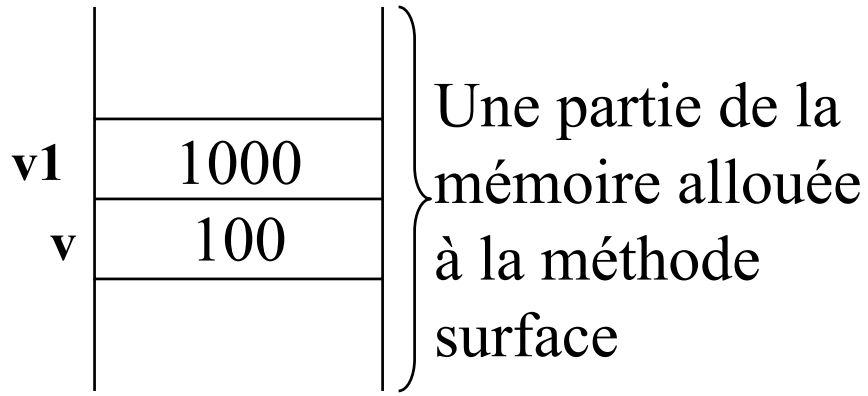



```
int getLongueurDuCôté(){
  return longueurDuCôté;    // valeur retournée
```

- `void setLongueurDuCôté(int longueur){`
  `// pas de valeur retournée`

Déclaration et Affectation

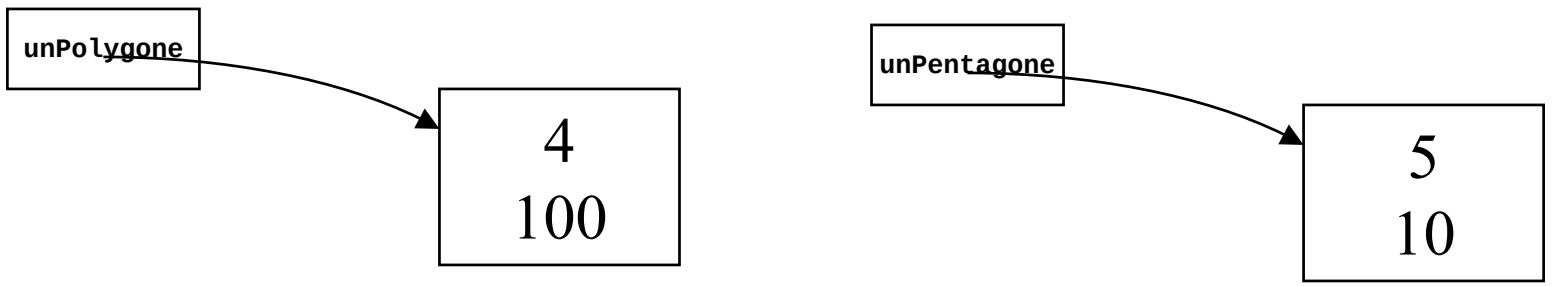
```
// déclaration de la méthode surface
public int  surface(){
    int v;  // une variable locale
    v = longueurDuCôté;
    int v1 = v * v;

    ….
```



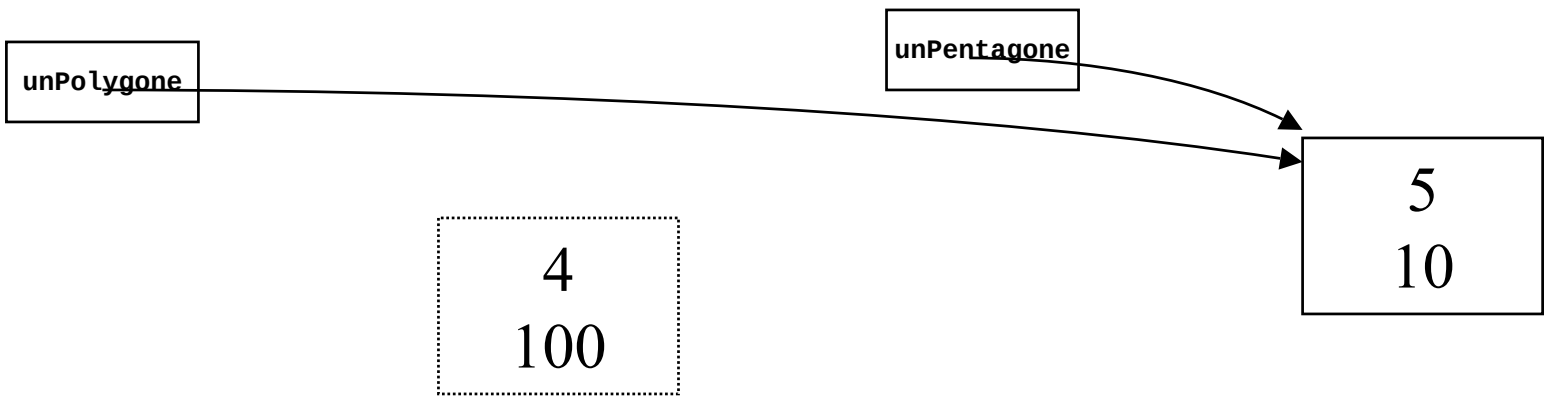

```
// appel de surface
int s = unPolygone.surface();
```

Affectation(2)

- `PolygoneRégulier unPolygone, unPentagone;`
- `unPolygone = new PolygoneRégulier (4,100);`
- `unPentagone = new PolygoneRégulier (5,10);`



- **`unPolygone = unPentagone;`**



Instruction conditionnelle

- ***if (expression-booleenne) instructions1; [else instructions2]***

public class Ticket{

```
public void insertMoney(int amount)    {
   if(amount > 0) {
      balance += amount;
   } else {
      System.out.println("Use a positive amount: " + amount);
   }
}
```

System.out.println : affichage sur la console

Quelques opérateurs

- **+ , - , * , /**

```
int x =0;
int y = x +1;
```

- **-=,+=**

```
int x=2;  x += 2;  // x = x  + 2;
```

- **++, --**

```
x++;  // x = x +1 ou x+=1
```

- **+**

```
String s = "bon"  + "jour";
int ammount = 1000;
System.out.println( "amount =  " + amount);
```

- **>= , > , <= , <**

```
boolean b = x <100;
if(amount <= 0){
  System.out.println(" erreur!");
}else{
  total = total + amount;
```

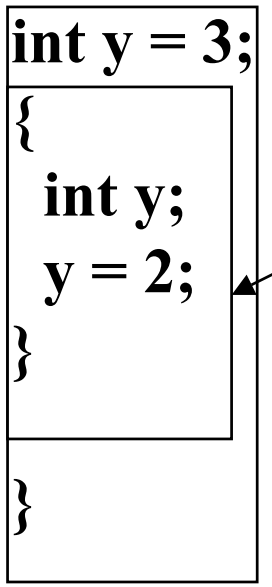
- **== , !=**

```
if(x == 1)
```

Visibilité(1)

```
x = 3; // erreur ici
int x;
x += 2;




}
```

Visibilité de x
après sa déclaration



Visibilité de y

```
int y = 3;
{
  int y;
  y = 2;
}

}
```

Visibilité de y
(un autre y)

Visiblité(2)

```
// Attention à la visibilité (réduite…)
public class PolygoneRégulier{
   private int longueurDuCôté;
   private int nombreDeCôtés;


   // erreur possible
   public PolygoneRégulier(int nombre, int longueur){
       int longueurDuCôté = longueur;
       nombreDeCôtés = nombre;
   }
```

Visibilité(3)

```
// Attention à la visibilité
public class PolygoneRégulier{
   private int longueurDuCôté;
   private int nombreDeCôtés;


   public PolygoneRégulier(int nombre, int longueurDuCôté ){
       longueurDuCôté = longueurDuCôté;
       nombreDeCôtés = nombre;
   }
```

- *Seule solution l 'emploi de* **this**

```
public PolygoneRégulier(int nombre, int longueurDuCôté ){
       this.longueurDuCôté = longueurDuCôté;
       nombreDeCôtés = nombre;
   }
```

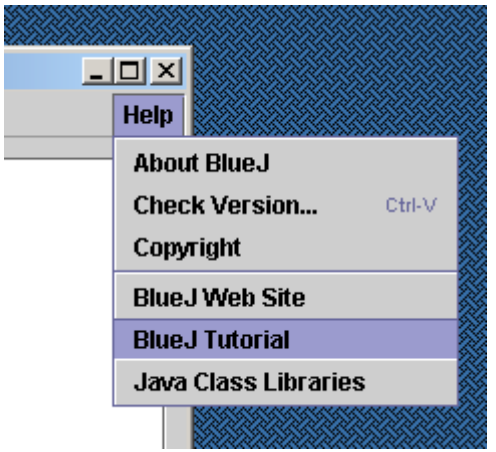
Commentaires du source

```
/** accesseur, lecture du NombreDeCôté de ce polygone.
 * @return le nombre de côtés
 */
public int getNombreDeCôtés(){
  return nombreDeCôtés;
}

/** mutateur, modification de la longueur du Côté.
 * @param la nouvelle longueur
 */
public void setLongueurDuCôté(int longueur){
  longueurDuCôté = longueur; // attention aux valeurs négatives
}
```

Outil javadoc, exécuté par BlueJ

- http://java.sun.com/j2se/1.4/docs/tooldocs/javadoc/index.html

précède ce que l 'on désire commenter, et à chaque nouvelle classe, BlueJ engendre ces lignes




```
/**
 * Write a description of class Test here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
 public class PolygoneRégulier{
  /**
   * An example of a method - replace this comment with your own
   *
   * @param  y   a sample parameter for a method
   * @return     the sum of x and y
   */
  public int sampleMethod(int y){
      // put your code here
      return x + y;
  }
```

Commentaires en pages HTML pour les utilisateurs

Package **Class** Tree Index Help

PREV CLASS NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

Class PolygoneRégulier

java.lang.Object
  |
  +--**PolygoneRégulier**

Direct Known Subclasses:
Carré

public class **PolygoneRégulier**
extends Object

Constructor Summary

	PolygoneRégulier(int nombre, int longueur)

Method Summary

int	**getLongueurDuCôté**()
int	**getNombreDeCôtés**() accesseur, lecture du NombreDeCôté de ce polygone.
int	**perimetre**()
void	**setLongueurDuCôté**(int longueur) mutateur, modification de la longueur du Côté.

getNombreDeCôtés

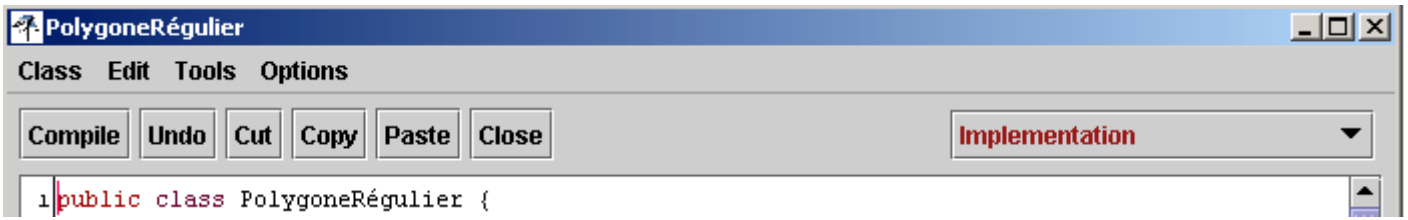
```
public int getNombreDeCôtés()
```

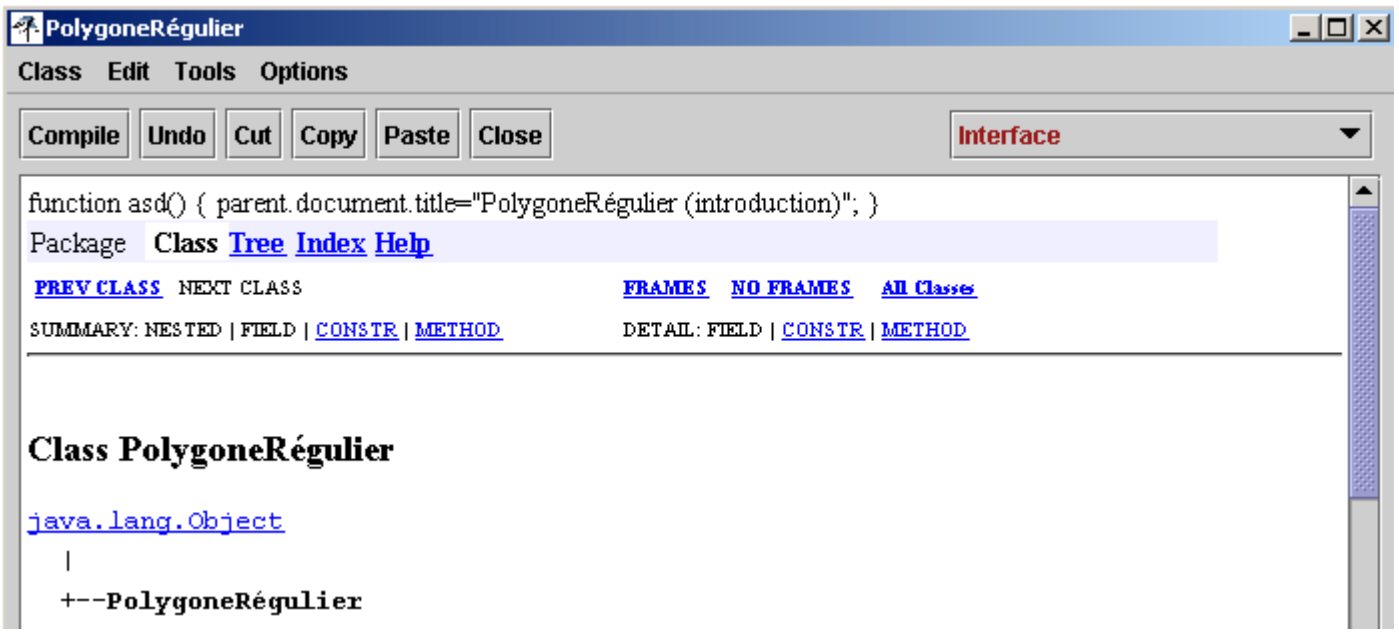
accesseur, lecture du NombreDeCôté de ce polygone.

Returns:
le nombre de côtés

Commentaires avec BlueJ

- **Génération simplifiée des commentaires (HTML)**
  du projet : tutorial page 29
- **génération unitaire**





blue jay