Java, les objets : tout de suite !
Interaction entre objets

Cnam Paris
jean-michel Douin, douin@cnam.fr
Version du 22 Janvier 2003

Notes de cours associées au chapitre 3
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

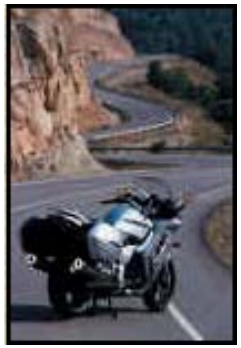
- abstraction et modularité
- diagramme de classes
- diagramme d'objets
- références d'objet
- types primitifs
- types d'objet
- création d'objet
- surcharge (polymorphisme ad'hoc)
- appel de méthodes internes ou externes
- débogueur et points d'arrêt

Abstraction et Modularité

- **Abstraction**

- **Modularité**

Exemple : les Véhicules



- **Véhicule,**

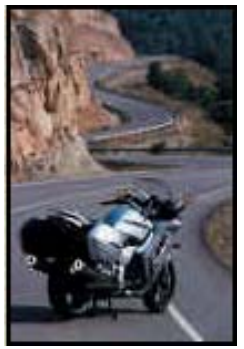
  obtenir la distance parcourue en fonction de la consommation

- **Quels sont les objets ?**

  Compteur

  Véhicule

Classe Véhicule



```
public class Véhicule{
 /** le compteur de ce véhicule */
  private Compteur compteur;
 /** le réservoir, initialisé à 0.0. */
  private double   jauge;
  /** le nom donné à ce véhicule */
  private String nom;
  /** La consommation de ce véhicule. */
  private double   consommation;

  public Vehicule(String nom, double consommation){
    this.compteur = new Compteur();
    this.consommation = consommation;
    this.nom = nom;
  }
  ...
}
```

Classe Compteur

```
public class Compteur{
  /** le totalisateur, ne peut être remis à zéro */
  private double totalisateur;
  /** le partiel */
  private double partiel;

  /** Création d'un compteur. */
  public Compteur(){
    totalisateur = partiel = 0.0;
  }
...
```

Diagramme de classes

- **Véhicule est composée de Compteur**

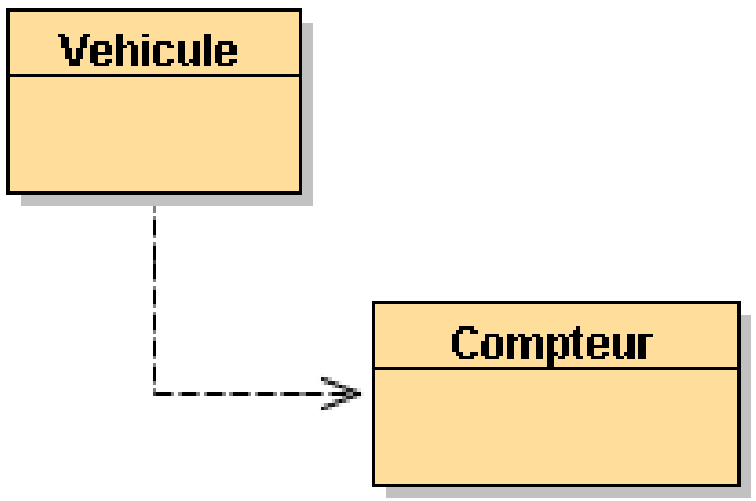
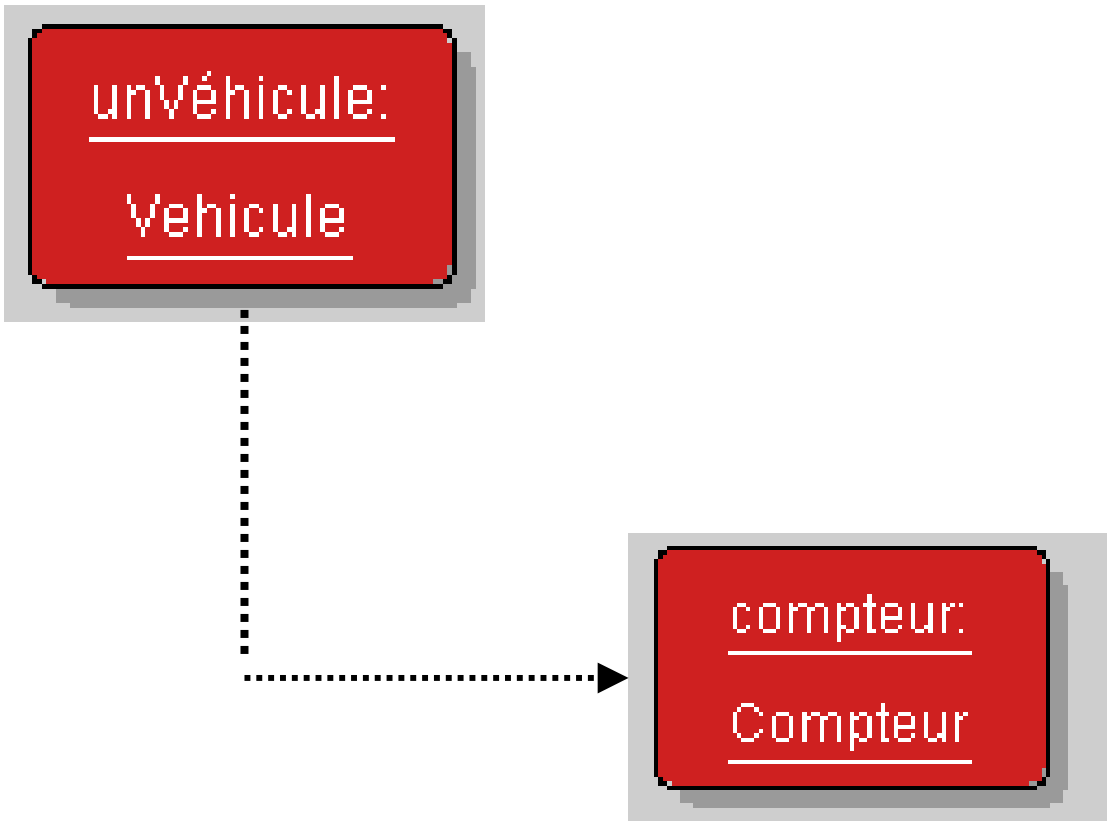


Diagramme d'instances

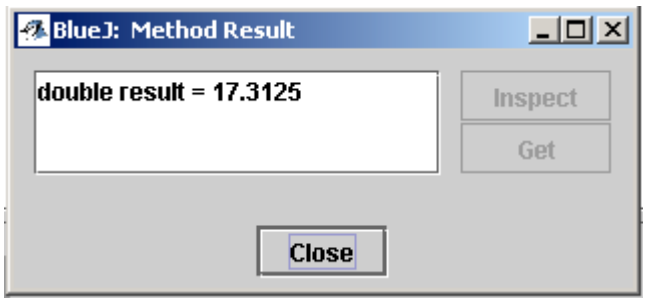
L 'instance « unVéhicule » est composée d 'une instance « compteur »



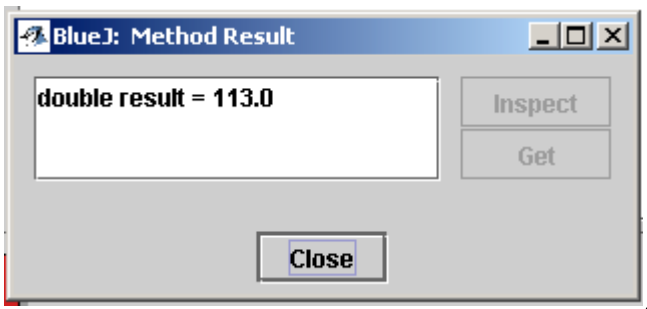
Source Java : exemple

- **Un extrait possible :**

```
Véhicule v = new Véhicule("FJR",0.075);

v.faireLePlein();
v.rouler(77.50);
v.rouler(25.00);
double jauge = v.getJauge();
```



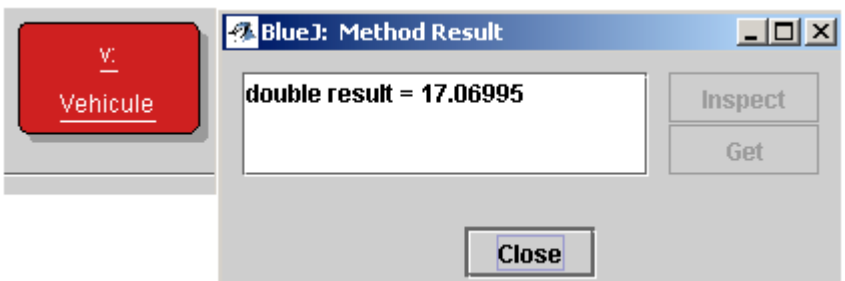

```
v.rouler(10,50);
double distance = v.distanceParcourue();
```



Source Java : exemple (suite)

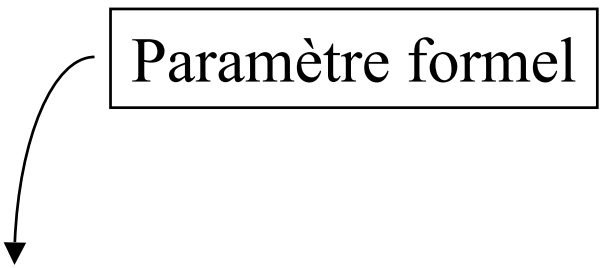
- **Suite de l ’extrait possible : Obtention du compteur**

```
//Véhicule v = new Véhicule("FJR",0.075);

Compteur c = v.getCompteur();
double dist = c.getTotalisateur();

// ou encore
double dist2 = v.getCompteur().getTotalisateur();

v.rouler(3.234);
v.getJauge();
```



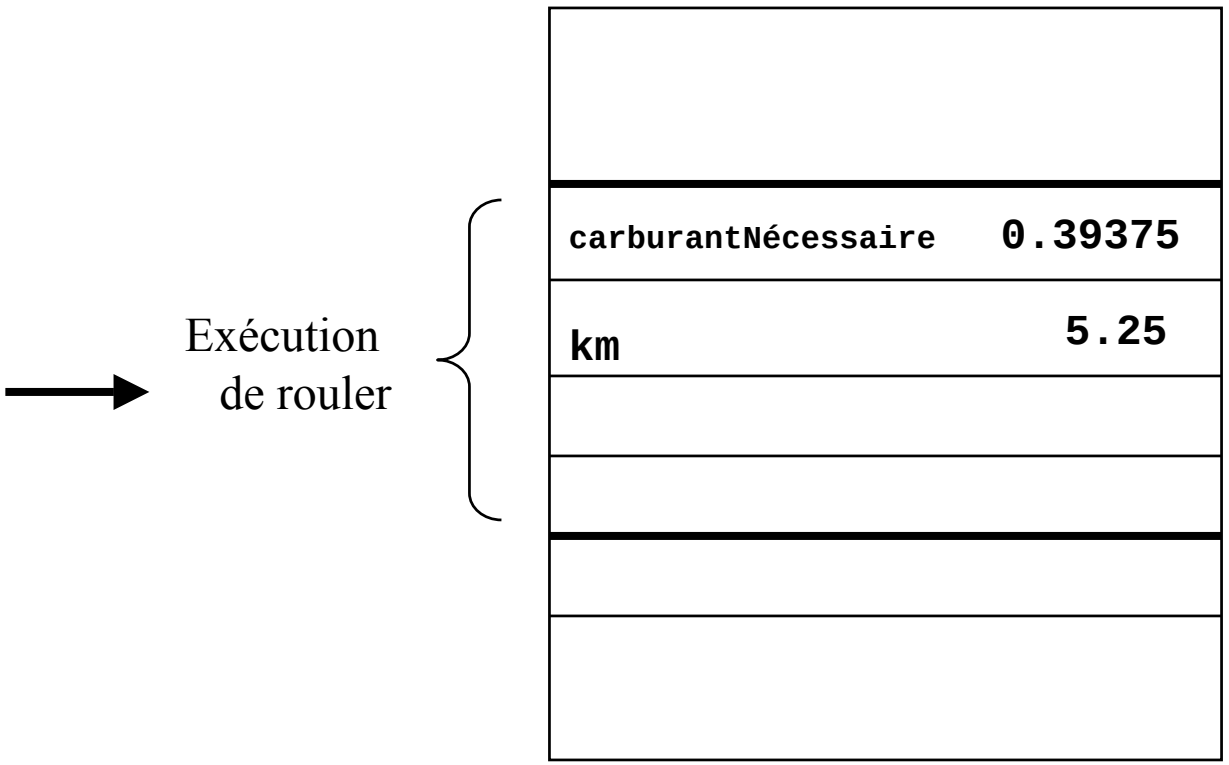
Paramètres formels et effectifs, par recopie

- **`v.rouler(5.25);`**




```
public void rouler(double km){
  double carburantNécessaire = (consommation*km);
  if(jauge-carburantNécessaire >= 0){
    compteur.add(km);
    jauge = jauge - carburantNécessaire;
  }else{
    compteur.add(jauge/consommation);
    jauge = 0.0;
  }
}
```

Appel de la méthode rouler, état de la pile



Références et paramètres

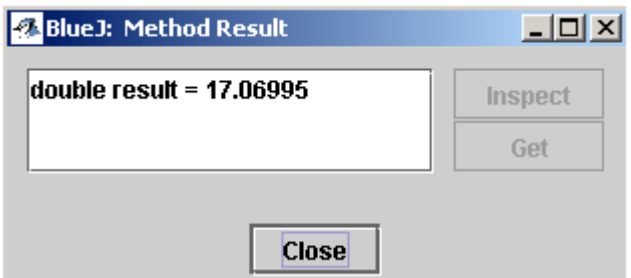
- **`maCollectionDeVéhicules.ajouter( v);`**

Paramètre effectif

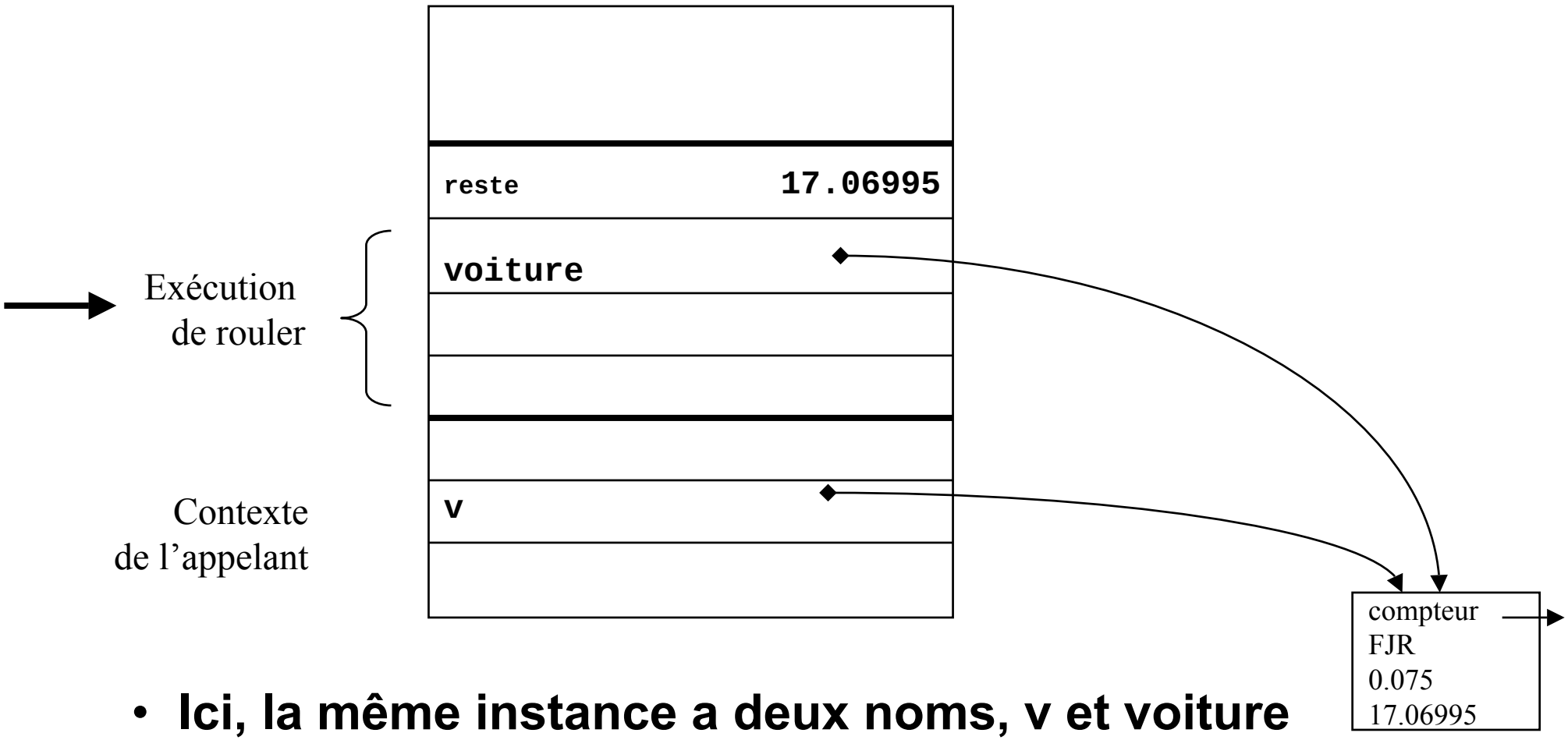
Paramètre formel

…

```
public void ajouter(Véhicule voiture){
  …….
  double reste = voiture.getJauge();
}
```



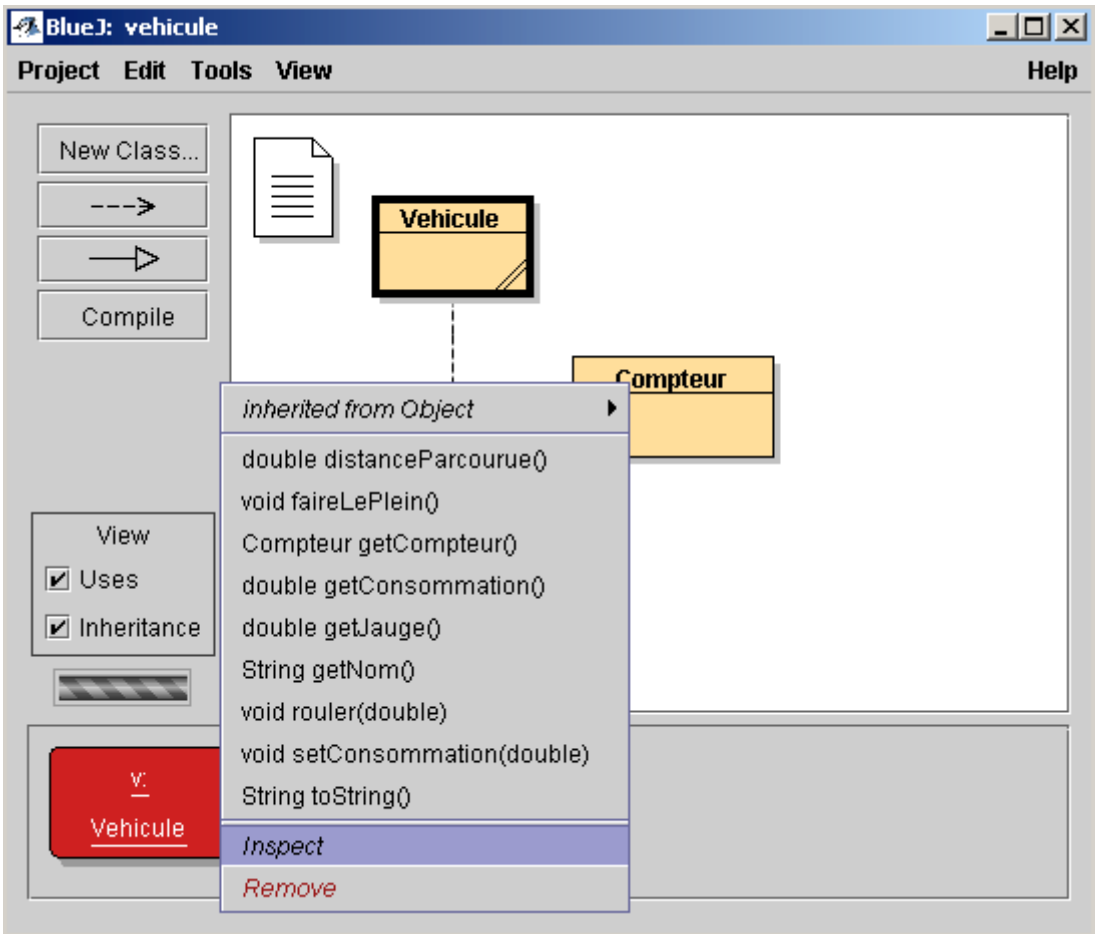
Appel de la méthode ajouter, état de la pile



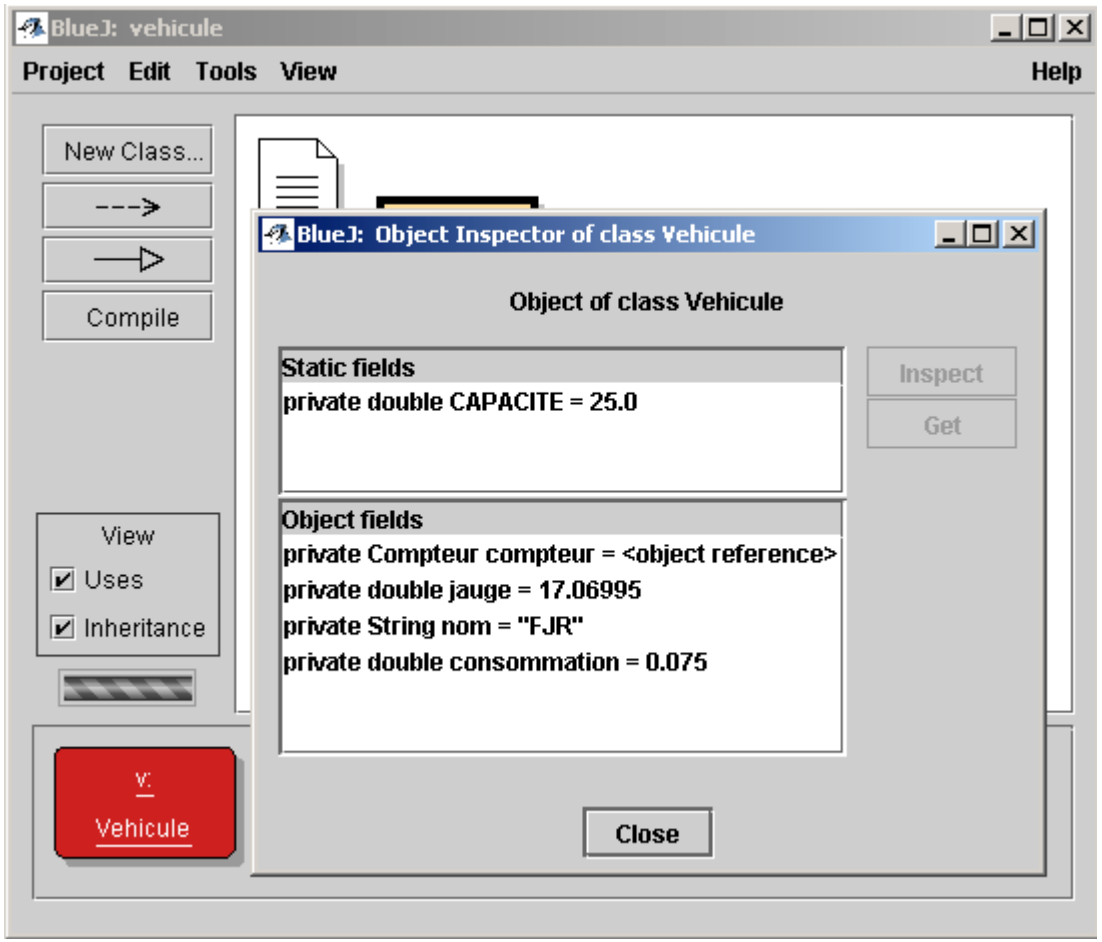
- **Ici, la même instance a deux noms, v et voiture**

BlueJ : Véhicule v

- **`Véhicule v = new Véhicule("FJR",0.075);`**

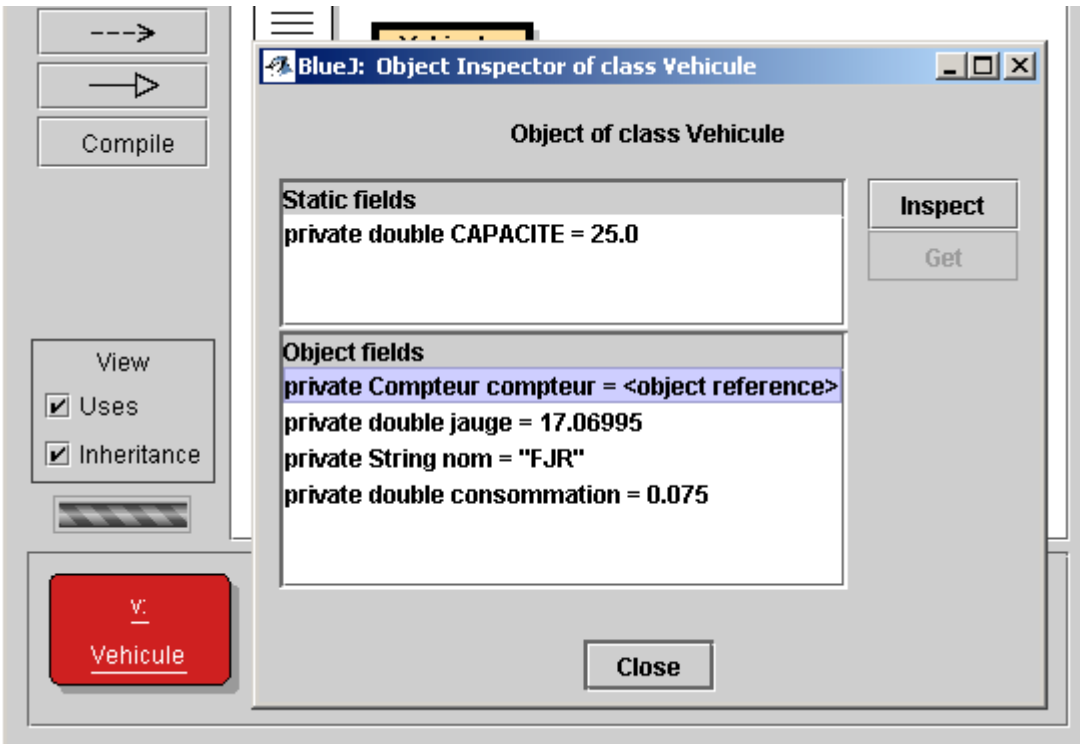


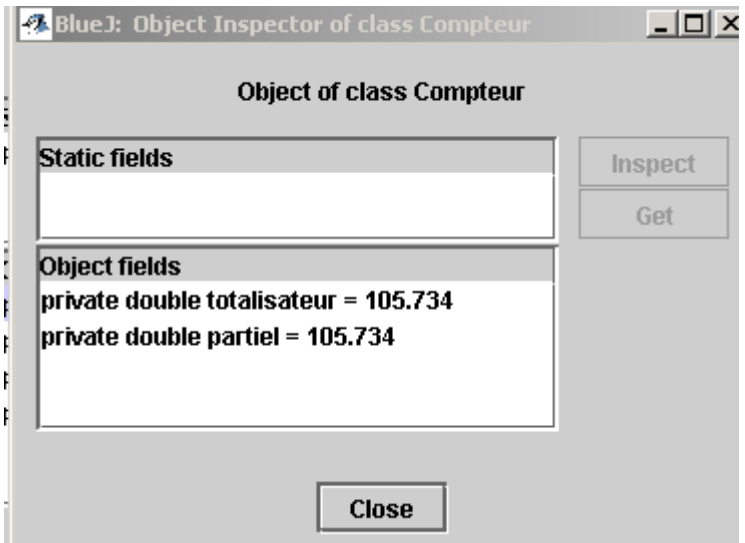
BlueJ : Inspection



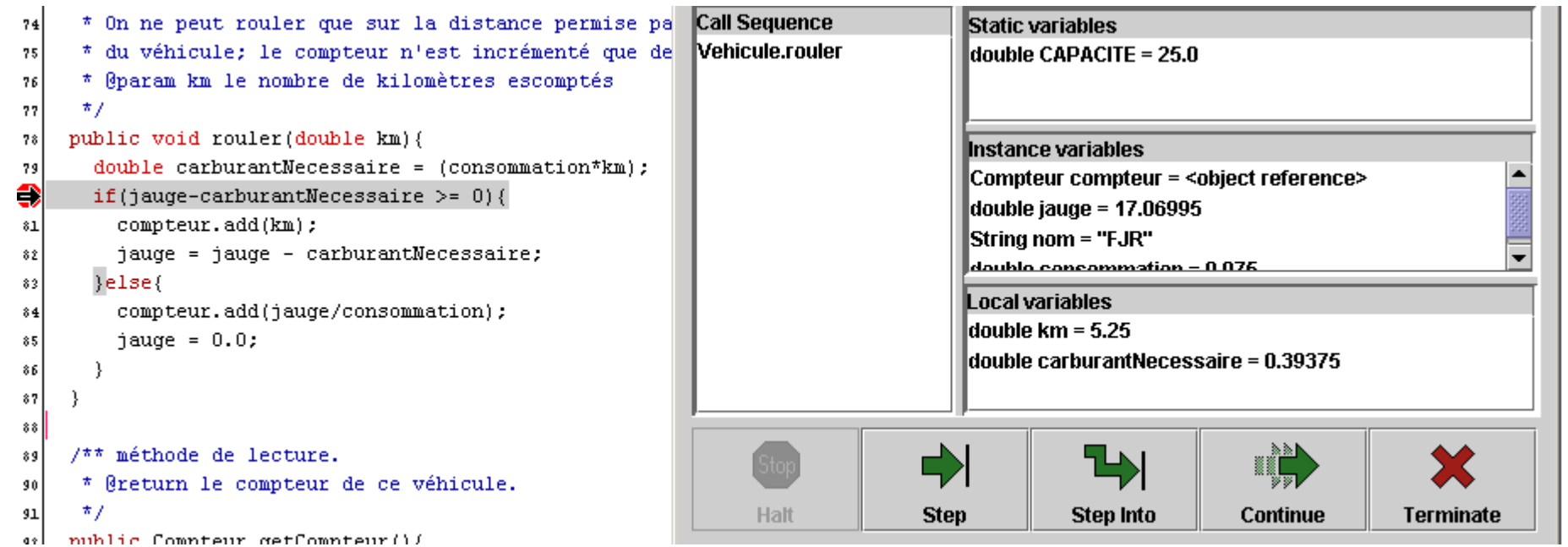
BlueJ : Obtenir le compteur de ce véhicule

- **Inspection de l ’inspection**





BlueJ : point d'arrêt



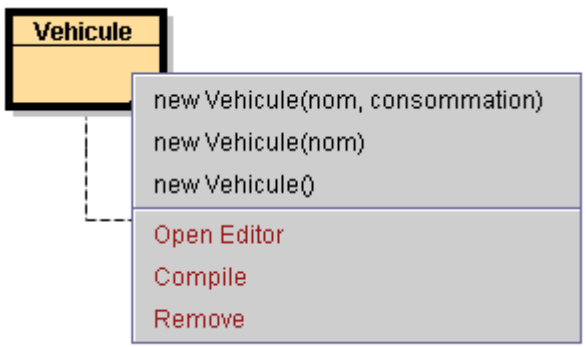
Surcharge, du constructeur

```
public Vehicule(String nom, double consommation){
  this.nom = nom;
  this.consommation = consommation;
  this.compteur = new Compteur();
}

public Vehicule(String nom){
  this.nom = nom;
  this.consommation = 0.1;
  this.compteur = new Compteur();
}

public Vehicule(){
  this.nom = "anonyme";
  this.consommation = 0.1;
  this.compteur = new Compteur();
}
```

this désigne l ’instance en cours

Surcharge du constructeur : 3 écritures possibles

- `Véhicule v = new Véhicule("FJR",0.075);`
- `Véhicule v1 = new Véhicule("majestic");`
- `Véhicule v2 = new Véhicule();`



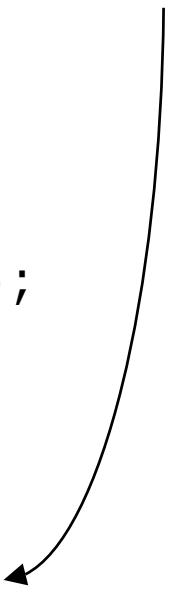
Un autre usage de this

```
public Vehicule(String nom, double consommation){
   this.nom = nom;
   this.consommation = consommation;
   this.compteur = new Compteur();
 }

 public Vehicule(String nom){
   this(nom, 0.1);
 }

 public Vehicule(){
   this("anonyme", 0.1);
 }
```

- **this(,)** : appel du constructeur a deux paramètres, avec cet entête

Appel de méthodes internes

```
public void rouler(double km){
   double carburant = carburantNécessaire(km);
   if(jauge-carburant >= 0){
     compteur.add(km);
     jauge = jauge - carburant;
   }else{
     compteur.add(jauge/consommation);
     jauge = 0.0;
   }
 }

 private double carburantNécessaire(double km){
   return consommation*km;
 }
```



note : préfixation éventuelle par this.

```
      double carburant = this.carburantNécessaire(km);
```

Appel de méthodes externes

```
 public class Véhicule{
  /** méthode de lecture.
   * @return la distance parcourue par ce véhicule.
   */
  public double distanceParcourue(){
    return compteur.getTotalisateur();
  }
...
```

```
 public class Compteur{
  private double totalisateur;

  public double getTotalisateur(){
    return totalisateur;
  }
...
```

Java : Quels sont les types primitifs ?

- **entier**

  signés seulement

  type **byte** (8 bits), **short** (16 bits), **int** (32 bits), **long** (64 bits)

- **flottant**

  standard IEEE

  type **float**(32 bits), **double** (64bits)

- **booléen**

  type **boolean** (true,false)

- **caractère**

  unicode,

  type **char** (16 bits) **http://www.unicode.org**

Type primitif et changement de type

- **Automatique**

  si la taille du type destinataire est supérieure

```
byte a,b,c;
int d = a+b/c;
```

- **Explicite**

```
byte b = (byte)50;
b = (byte) b * 2;            // b * 2 promue en int
```

par défaut la constante numérique est de type int,
suffixe L pour obtenir une constante de type long **40L**

par défaut la constante flottante est de type double,
suffixe F pour obtenir une constante de type float **40.0F**

Conversions implicites

- **Automatique**

  si la taille du type destinataire est supérieure

byte	**->**	**short,int,long,float,double**
short	**->**	**int, long, float, double**
char	**->**	**int, long, float, double**
int	**->**	**long, float, double**
long	**->**	**float, double**
float	**->**	**double**

Exemple : La classe Conversions

```
public class Conversions{

public  void exempleSyntaxique( ){
   byte b;short s;char c;int i;long l;float f;double d;

   b=(byte) 0; s = b; i = b; l = b; f = b; d = b;

   i = s; l = s; f = s; d = s;

   i = c; l = c; f = c; d = c;

   l = i; f = i; d = i;

   f = l; d = l;

   d = f;
   }
}
```

Déclarations de constantes

- `private final static double CAPACITE = 50.0;`
- `private final static String hi = "hello";`
- `public final int MAXIMUM = 100;`
- `public final long TAILLE_MAX = 100L;`
- `public final static byte MAX = (byte)0xFF;`

Type caractère

- **Java utilise le codage Unicode**

- **représenté par 16 bits**
- **\u0020 à \u007E code ASCII, Latin-1**

  \u00AE ©

  \u00BD / la barre de fraction ...
- **\u0000 à \u1FFF zone alphabets**

  ....

  \u0370 à \u03FFF alphabet grec

  .....

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

Opérateurs

- **Arithmétiques**

  +, -, *, /, %, ++ +=, -=, /=, %=, --,

- **Binaires**

  ~, &, |, ^, &=, |=, ^=

  >>, >>>, <<, >>=, >>>=, <<=

- **Relationnels**

  ==, !=, >, <, >=, <=

  Syntaxe C

- **Booléens**

  &, |, ^, &=, |=, ^=, ==, !=, !, ?:

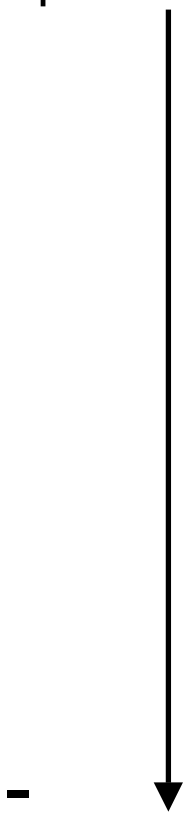
  ||, &&

Opérateurs booléens et court-circuits, exemple

```
1   public class Div0{
2     public void exempleDeCourtCircuits(
3       int den = 0, num = 1;
4       boolean b;
5
6       System.out.println("den == " + den);
7
8        b = (den != 0  &&  num / den > 10);
9
10       b = (den != 0  & num / den > 10);
11  }
12}
```

Exception in thread "main" java.lang.ArithmeticException :
/ by zero at Div0.main(Div0.java:10)

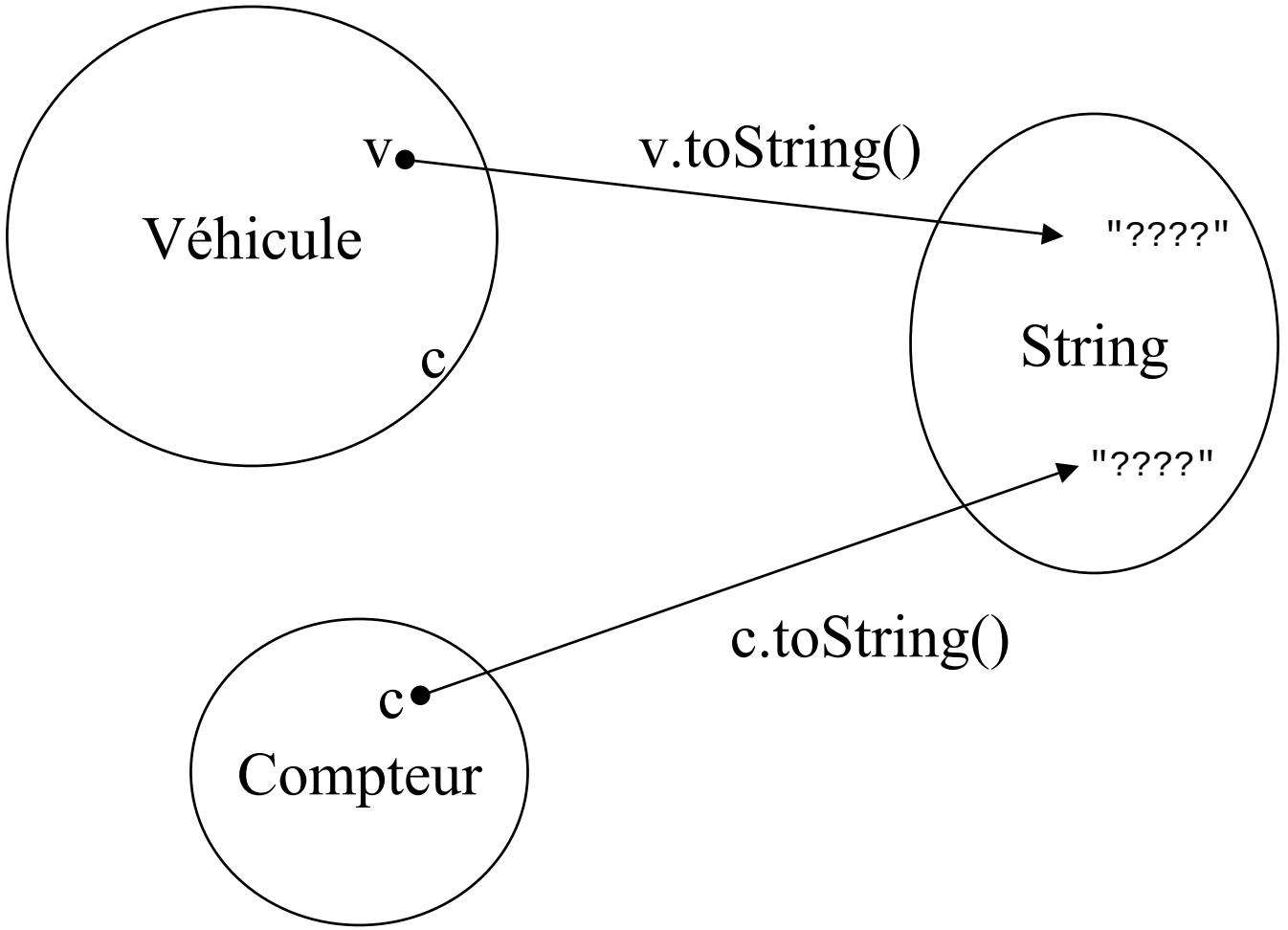
Précédence des opérateurs



()	[]	.		
++	--	~		!
*	/	%		
+	-			
>>	>>>	<<		
>	>=	<	<=	
==	!=			
&				
^				
|				
&&				
||				
?:				
=	*op*=			

- **int a = 1, b = 1, c=2;**
- **int x = a | 4 + c >> b & 7 | b >> a % 3;** // ??? Que vaut x ???

Un retour sur la classe Véhicule



Méthode toString()

- `Véhicule v = new Véhicule("FJR",0.075);`
- `v.faireLePlein();`
- `v.rouler(23.28);`
- `v.rouler(21.06);`
- `Compteur c = v.getCompteur();`

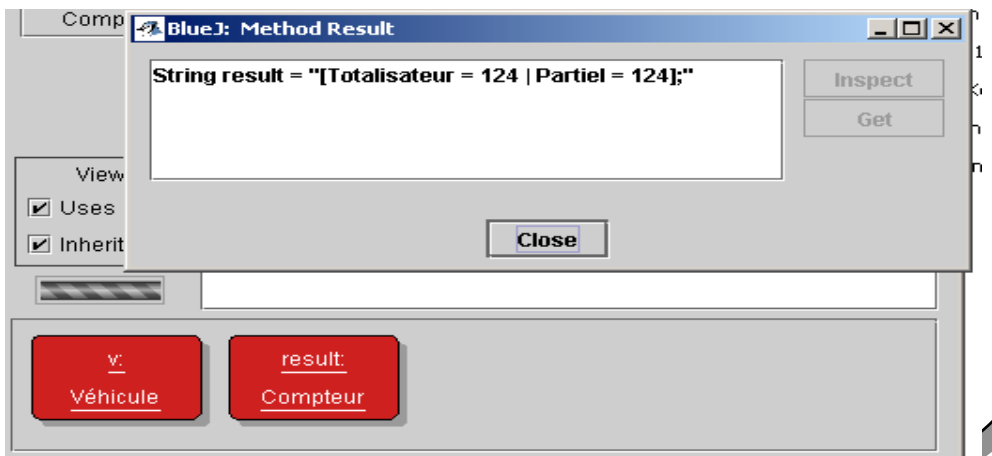
- `String vs = v.toString();`

- `String vc = v.getCompteur().toString();`

- `System.out.println("pour le FJR" + vs);`

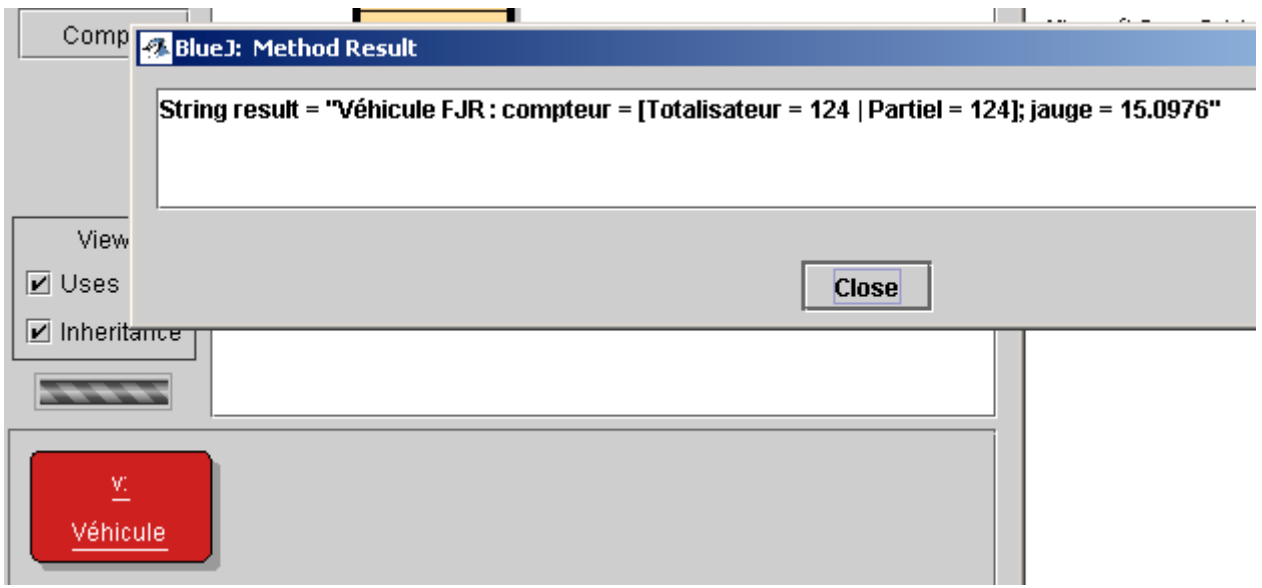
méthode toString(), classe Compteur

```
/** conversion en String.
 * affichage sous la forme d'un nombre entier
 *    entre 0 et 99 999 pour le totalisateur
 * et entre 0 et 999 pour le partiel
 * @return la chaîne selon ce format :
 *            "[Totalisateur = totalisateur | Partiel = partiel];"
 */

public String toString(){
  int total = (int)Math.round(this.totalisateur)%100000;
  int part  = (int)Math.round(this.partiel)%1000;
  return "[Totalisateur = " +  total +  " | Partiel = " + part + "];";
}
```



méthode toString(), classe Véhicule

```
/** conversion en String.
  * @return une chaîne selon ce format :
  * "Véhicule FJR  nom : compteur.toString() jauge = valeur
  */
 public String toString(){
   return "Véhicule " + getNom() + " : compteur = " +
          compteur.toString() + " jauge = " + getJauge();
 }
```



Synthèse : résumé, concepts