Conception objet en Java avec BlueJ
une approche interactive

5. Des comportements plus complexes

Utiliser des bibliothèques de classes pour implanter des fonctionnalités avancées

David J. Barnes, Michael Kölling
version française: Patrice Moreaux

Rédigé avec OpenOffice.org

Principaux concepts étudiés

- Utiliser des bibliothèques de classes
- Lire la documentation
- Écrire une documentation

Les bibliothèques de classes Java

- Des milliers de classes
- Des dizaines de milliers de méthodes
- Beaucoup de classes utiles qui rendent la vie plus facile
- Un programmeur Java compétent doit savoir employer les bibliothèques

Travailler avec les bibliothèques

Vous devriez:

- connaître quelques classes importantes par leur nom
- savoir comment trouver l'information sur les autres classes

Souvenez-vous:

- nous devons uniquement connaître l'interface, pas l'implantation

Un système de support technique

- Un système de dialogue en mode texte
- Inspiré du système «*Elisa*» de Joseph Weizenbaum (MIT, années 1960)

- *Expérimentez…*

Structure de la boucle principale

```
boolean finished = false;

while(!finished) {

    faire quelque chose

    if(conditon de sortie) {
        finished = true;
    }
    else {
        faire encore autre chose
    }
}
```

Corps de la boucle principale

```
String input = reader.getInput();
...
String response = responder.generateResponse();
System.out.println(response);
```

La condition de sortie

```
String input = reader.getInput();

if(input.startsWith("bye")) {
    finished = true;
}
```

- D'où provient "**`startsWith`**" ?
- Qu'est ce que c'est? Que fait-elle?
- Comment le savoir?

Lire la documentation des classes

- La documentation des bibliothèques Java est au format HTML
- Donc consultable avec tout navigateur Web
- Interface de programmation d'application (API: *Application Programmers' Interface)*
- Description des interfaces des classes de toutes les bibliothèques

Interface versus implantation (1)

La documentation comporte

- le nom de la classe
- une description générale de la classe
- Une liste des constructeurs et des méthodes
- Les valeurs de retour et les paramètres des constructeurs et des méthodes
- Une description du rôle de chaque constructeur et méthode

 c'est l'***interface*** **de la classe**

Interface versus implantation (2)

La documentation ***ne concerne pas:***

- Les champs privés (la plupart des champs sont privés)
- Les méthodes privées
- Les corps (code source) des méthodes

constituent l'***implantation*** **de la classe**

Utiliser les classes de bibliothèques

- Les classes d'une bibliothèque doivent être importées par une instruction **`import`** (sauf les classes de **`java.lang`**).
- Elles sont utilisables comme les classes du projet

Paquetages et importation

- Les classes sont regroupées en paquetages ("packages")
- On peut importer des classes une par une:

```
import java.util.ArrayList;
```

- Ou par paquetage entier:

```
import java.util.*;
```

Aparté: égalité de chaînes

```
if(input == "bye") {
    ...
}
```

teste l'égalité d'identité

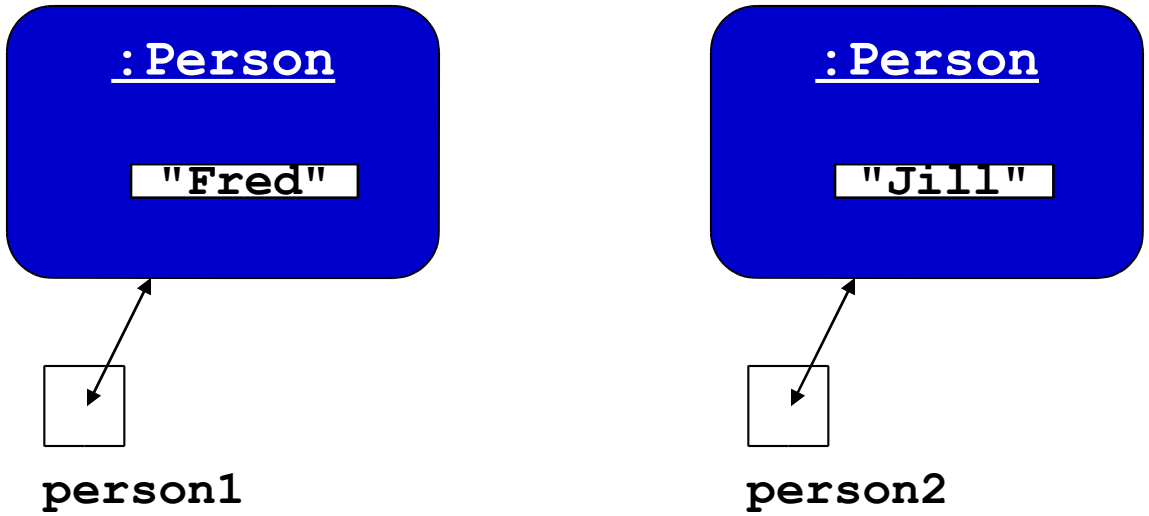
```
if(input.equals("bye")) {
    ...
}
```

teste l'égalité de valeur

- Comparer (presque toujours) les chaînes avec **`.equals`**

Identité versus égalité (1)

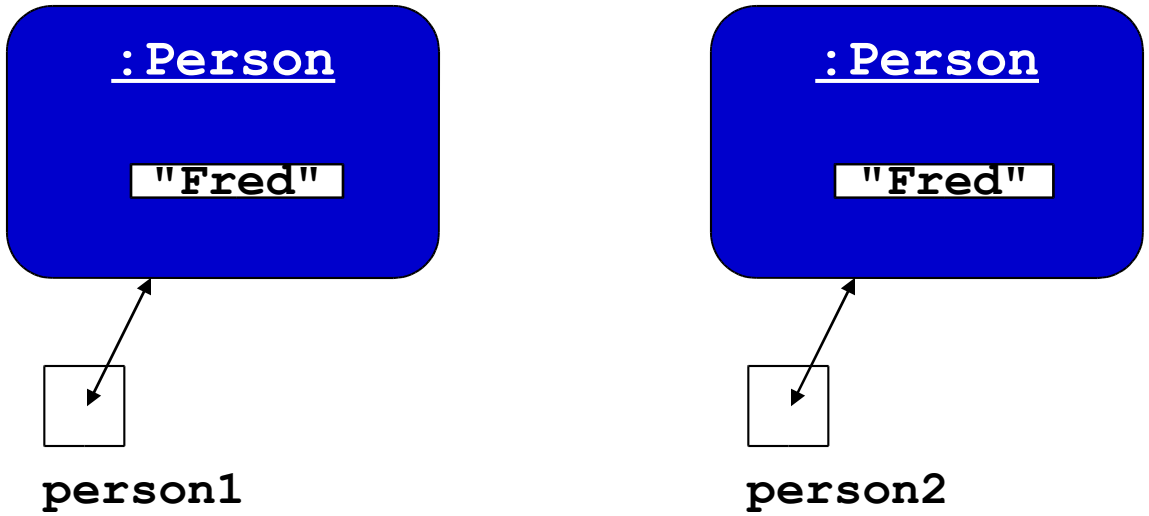
Autres objets (non chaînes):




```
person1 == person2 ?
```

Identité versus égalité (2)

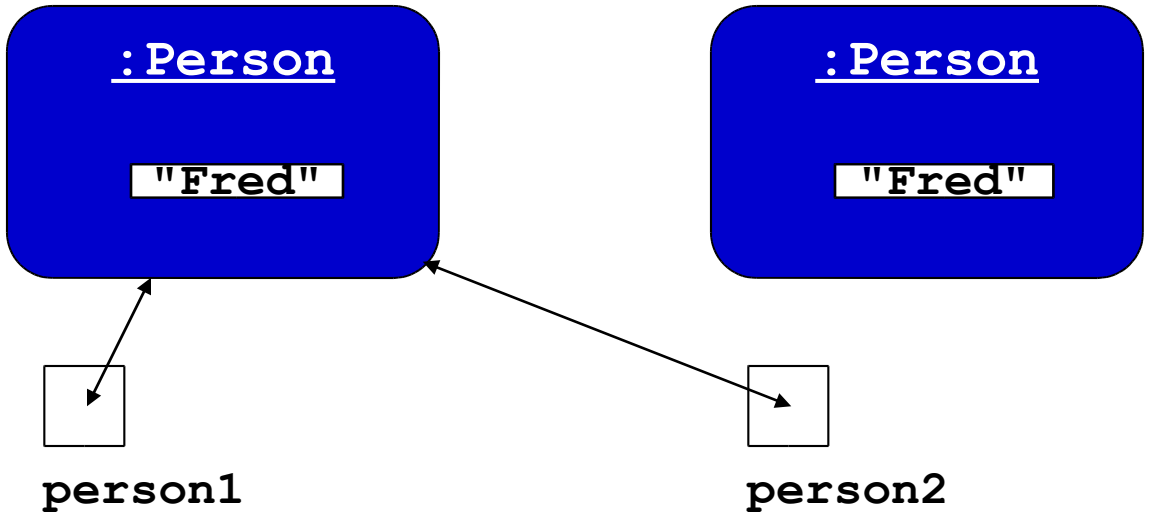
Autres objets (non chaînes):




```
person1 == person2 ?
```

Identité versus égalité (3)

Autres objets (non chaînes):

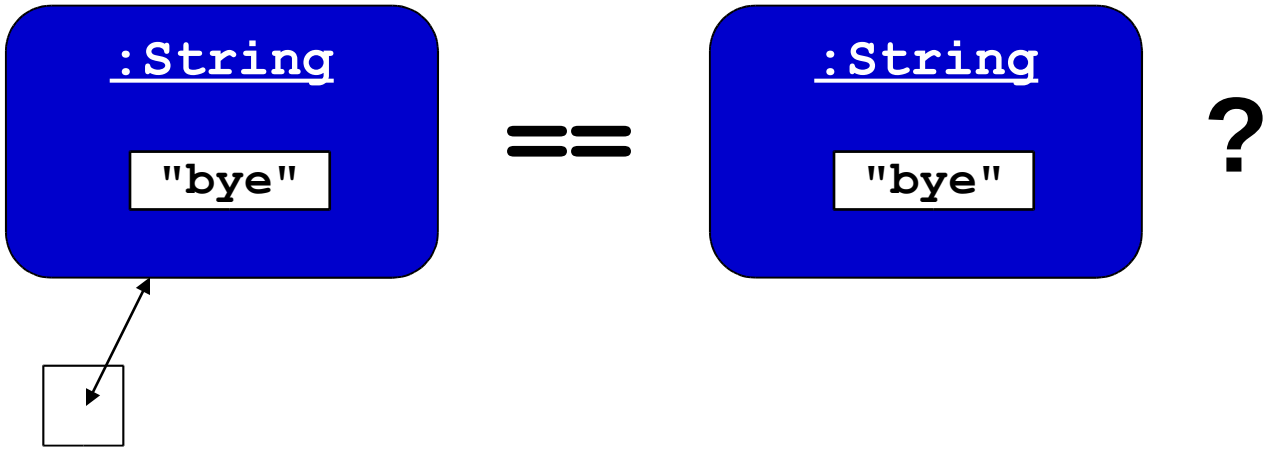



```
person1 == person2 ?
```

Identité versus égalité - chaînes - (1)

```
String input = reader.getInput();
if(input == "bye") {
    ...
}
```

== teste l'identité

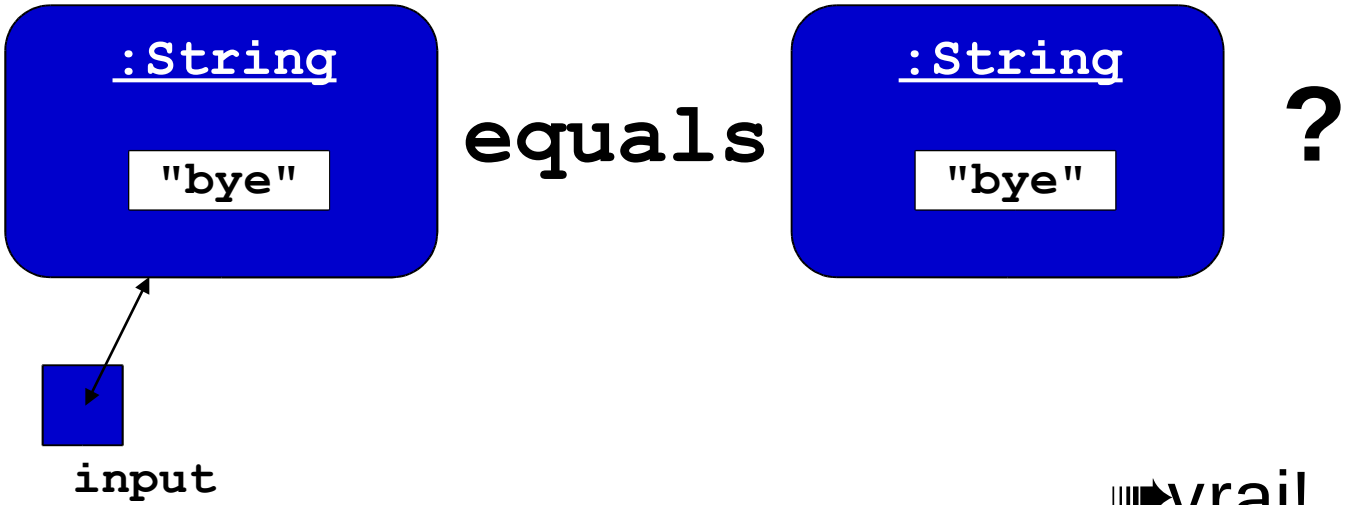


➠(peut être) faux!

Identité versus égalité - chaînes - (2)

```
String input = reader.getInput();
if(input.equals("bye")) {
    ...
}
```

`equals` teste l'égalité



➠vrai!

Utiliser **Random**

- La classe de bibliothèque **Random** génère des nombres aléatoires

```
import java.util.Random;
...
Random randomGenerator = new Random();
...
int index1 = randomGenerator.nextInt();
int index2 = randomGenerator.nextInt(100);
```

Générer des réponses aléatoires

```
public Responder()
{
    randomGenerator = new Random();
    responses = new ArrayList();
    fillResponses();
}

public String generateResponse()
{
    int index = randomGenerator.nextInt(responses.size());
    return (String) responses.get(index);
}

public void fillResponses()
...
```

Associations ("Maps“)

- Les associations sont des collections de paires d’éléments
- Une paire comporte une ***clé*** ("key") et une ***valeur***
- On fournit une clé et on récupère la valeur associée dans la paire
- Exemple: un annuaire ("*telephone book*").

Utiliser les associations (1)

- Une association avec des chaînes comme clés et comme valeurs

:HashMap

"Charles Nguyen"	"(531) 9392 4587"
"Lisa Jones"	"(402) 4536 4674"
"William H. Smith"	"(998) 5488 0123"

Utiliser les associations (2)

```
HashMap phoneBook = new HashMap();

phoneBook.put("Charles Nguyen", "(531) 9392 4587");
phoneBook.put("Lisa Jones", "(402) 4536 4674");
phoneBook.put("William H. Smith", "(998) 5488 0123");


String number = (String)phoneBook.get("Lisa Jones");
System.out.println(number);
```

Utiliser les ensembles

```
import java.util.HashSet;
import java.util.Iterator;
...
HashSet mySet = new HashSet();

mySet.add("one");
mySet.add("two");
mySet.add("three");

Iterator it = mySet.iterator();
while(it.hasNext()) {
    appeller it.next() pour obtenir le nouvel objet
    faire quelque chose avec cet objet
}
```

Comparez avec le code ArrayList

Scinder une chaîne en mots ("tokens")

```
public HashSet getInput()
{
    System.out.print("> ");
    String inputLine =
            readInputLine().trim().toLowerCase();

    StringTokenizer tokenizer =
            new StringTokenizer(inputLine);
    HashSet words = new HashSet();

    while(tokenizer.hasMoreTokens()) {
        words.add(tokenizer.nextToken());
    }
    return words;
}
```

Écrire la documentation d'une classe

- Vos propres classes doivent être documentées comme celles des bibliothèques
- Les autres programmeurs doivent pouvoir utiliser vos classes sans lire leurs implantations.
- Faites aussi «classe» que dans la bibliothèque standard!

Éléments de documentation (1)

La documentation d'une classe doit comporter:

- le nom de la classe
- un commentaire décrivant le but général et les caractéristiques de la classe
- un numéro de version
- le nom des auteurs
- la documentation de chaque constructeur et de chaque méthode

Éléments de documentation (2)

La documentation de chaque constructeur et de chaque méthode doit comporter:

- le nom de la méthode
- le type de retour
- les noms et types des paramètres
- une description du rôle et du comportement de la méthode
- une description de chaque paramètre
- une description de la valeur retournée

Javadoc (1)

Commentaires de classe:

```
/**
 * La classe Responder fournit un
 * générateur d'objets réponse.
 * Elle est utilisée pour générer
 * automatiquement une réponse.
 *
 * @author  Michael Kölling and David J. Barnes
 * @version 1.0  (1.Feb.2002)
 */
```

Javadoc (2)

Commentaire de méthode:

```
/**
 * Lit une ligne de texte sur l'entrée standard
 * (le terminal en mode texte)
 * et la retourne comme ensemble de mots.
 *
 * @param  prompt  Une invite à afficher.
 * @return  Un ensemble de chaînes, chacune est
 *          un des mots entrés par l'utilisateur
 */
public HashSet getInput(String prompt)
{
    ...
}
```

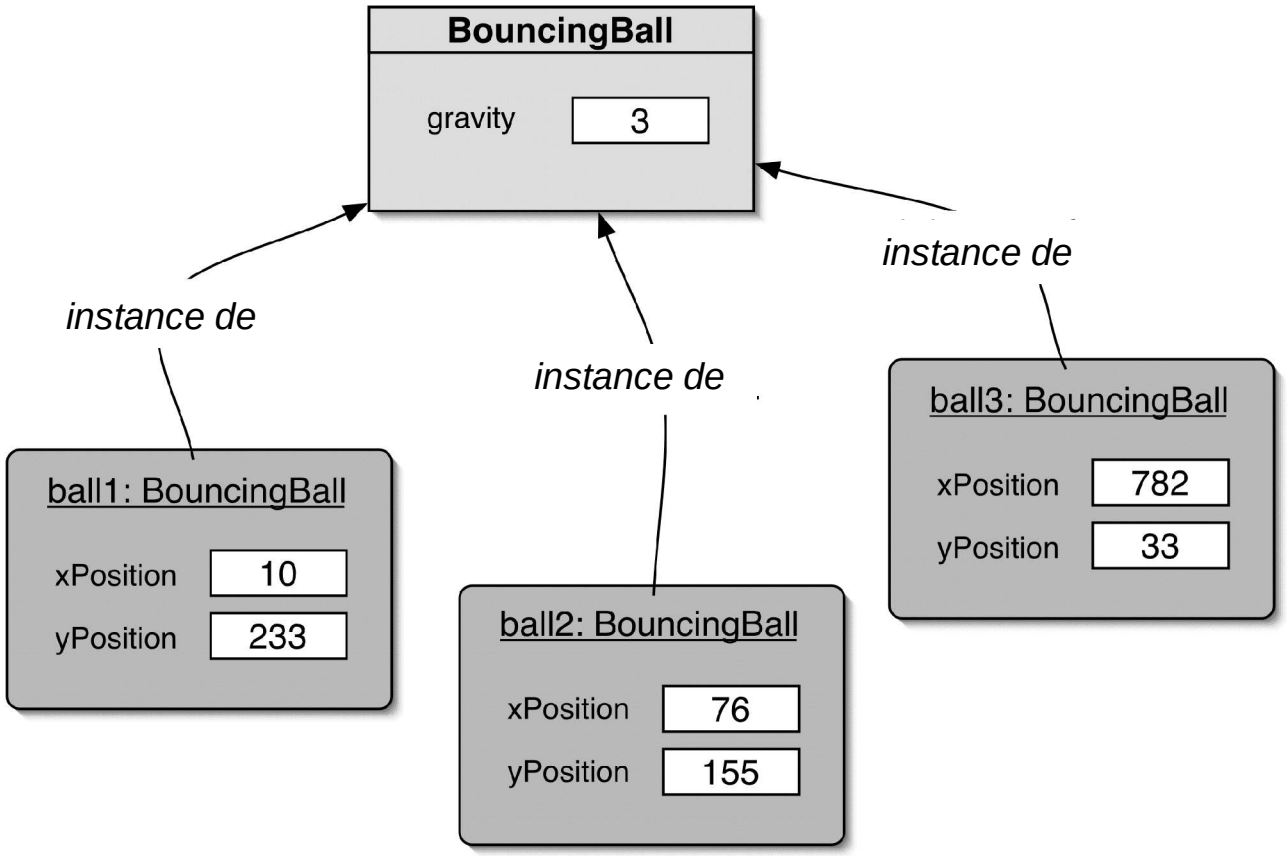
Public versus privé

- Les attributs publics (champs, constructeurs, méthodes) sont accessibles aux autres classes.
- Les champs ne devraient pas être publics.
- Les attributs privés ne sont accessibles que dans la même classe.
- Seules les méthodes prévues pour être utilisées par d'autres classes devraient être publiques.

Masquage d'information

- Les données d'un objet sont masquées aux autres objets.
- Savoir ***ce*** qu'un objet ***peut faire***, pas ***comment*** il le fait.
- Le masquage d'information élève le niveau d'indépendance.
- L'indépendance des modules est importante pour les grands systèmes et la maintenance.

Variables de classe



Constantes

```
private static final int gravity = 3;
```

- **`private`**: modificateur d'accès (comme d'habitude)
- **`static`**: variable classe
- **`final`**: constante

Résumé

- Java possède une bibliothèque de classes très étendue.
- Un bon programmeur doit connaître cette bibliothèque.
- La documentation nous indique ce que nous devons savoir pour utiliser une classe (son interface).
- L'implantation est masquée (masquage d'information).
- Nous documentons nos classes de sorte que l'interface puisse être lue seule (commentaires de classe et de méthodes).

Sommaire général