Conception objet en Java avec BlueJ

une approche interactive

2. Comprendre les définitions de classes

Analyser le contenu des classes

David J. Barnes, Michael Kölling

version française: Patrice Moreaux

Rédigé avec OpenOffice.org

Principaux concepts étudiés

- champs
- constructeurs
- méthodes
- paramètres
- instructions d’affectation
- instructions conditionnelles

Billetterie automatique
une vue externe

- Étude du comportement d'une billetterie automatique élémentaire.
  - utilisez le projet *naive-ticket-machine* .
  - la machine distribue des billets à prix fixe.
    - comment ce prix est-il déterminé?
  - comment l'«argent» est-il entré dans la machine?
  - comment la machine conserve-t-elle l'argent entré?

Billetterie automatique
une vue interne

- Interagir avec un objet donne des informations sur son comportement.
- L'étudier de l'intérieur permet de déterminer comment ce comportement est réalisé ou implanté.
- Toutes les classes Java possèdent une vue interne de même structure.

Structure de base d'une classe

```
public classe TicketMachine
{
    //*** partie de la classe
    //    non reproduite.
}
```

l'enveloppe de TicketMachine

```
public classe ClassName
{
    champs
    constructeurs
    méthodes
}
```

le contenu de la classe

Champs

- les champs stockent les valeurs pour un objet.
- aussi appelés variables d'instance.
- utilisez l'option *Inspecter* pour voir les champs d'un objet.
- les champs définissent l'état d'un objet.

```
public classe TicketMachine
{
    private int price;
    private int balance;
    private int total;

    //*** Constructeur et
    //    méthodes non
    //    reproduits.
}
```

modificateur de visibilité

type

nom du champ

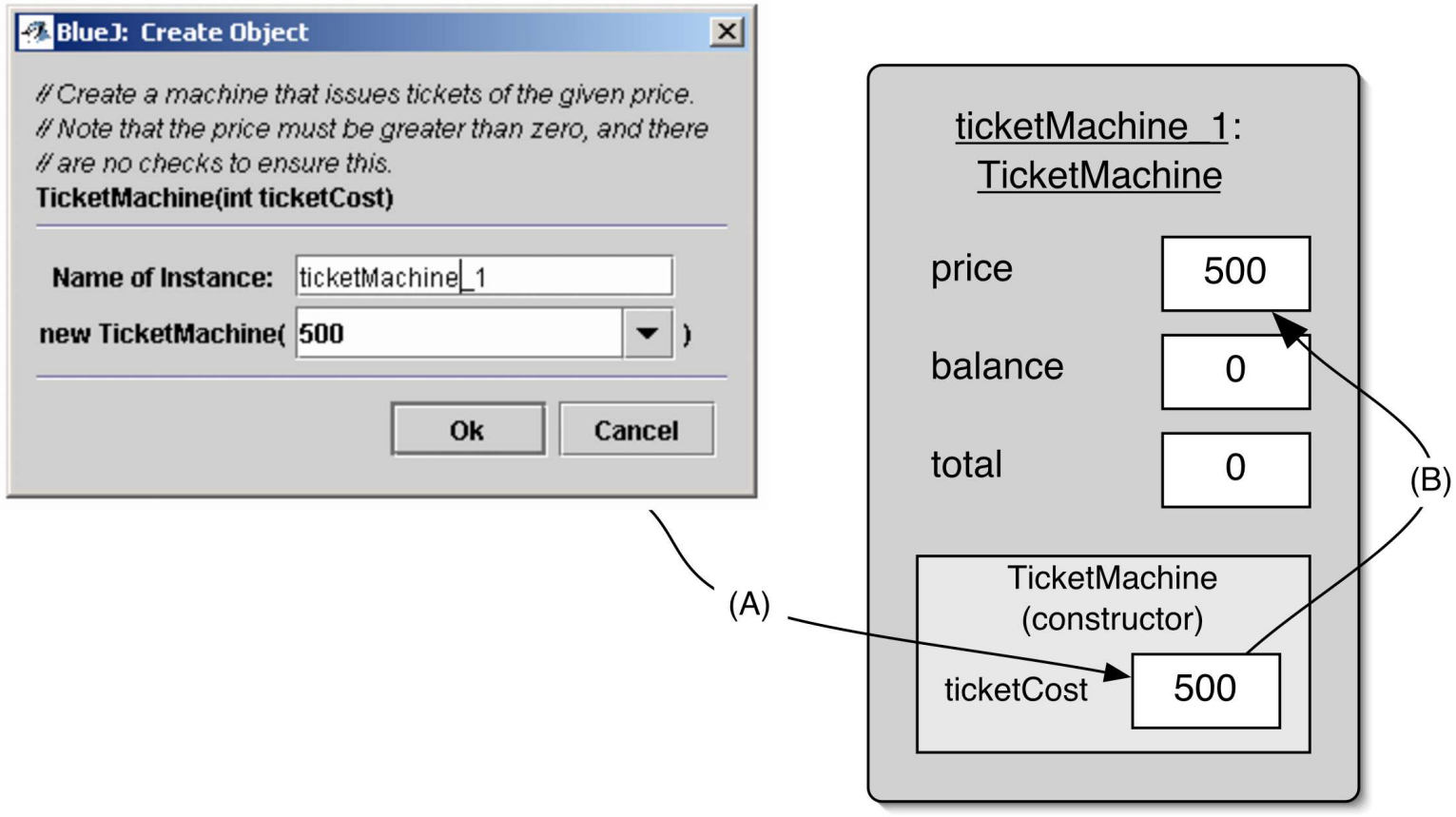
```
private int price;
```

Constructeurs

- Les constructeurs initialisent un objet.
- Ils ont le même nom que leur classe.
- Ils stockent les valeurs initiales dans les champs.
- Ils reçoivent souvent des valeurs externes en paramètre dans ce but.

```
public TicketMachine(int ticketCost)
{
    price = ticketCost;
    balance = 0;
    total = 0;
}
```

Fournir des données *via* les paramètres



Affectation

- Les valeurs sont stockées dans les champs (et dans les autres variables) avec des instructions d'affectation:
  - *variable = expression;*

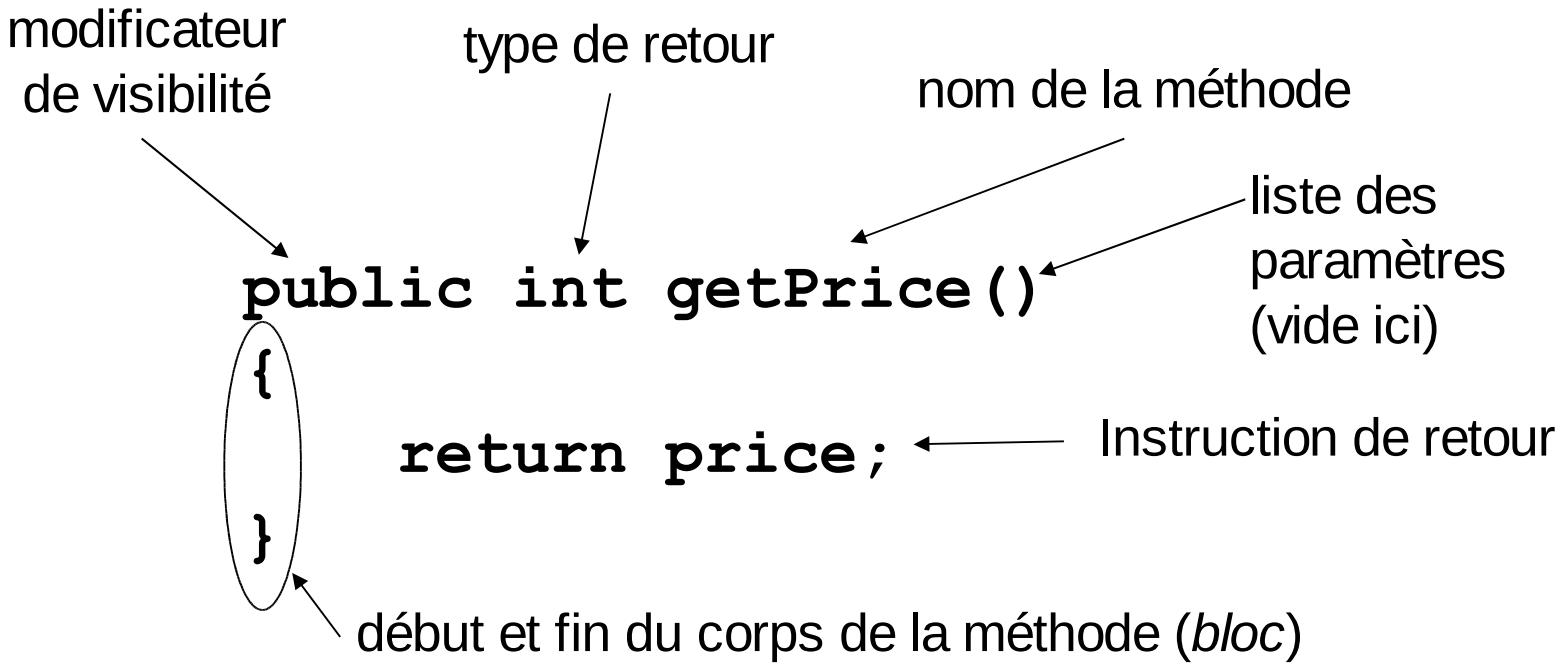
```
price = ticketCost;
```

- Une variable stocke une unique valeur, donc toute valeur précédente est perdue.

Méthodes d'accès (1)

- Les méthodes implantent le comportement des objets.
- Les méthodes d'accès (ou accesseurs) fournissent des informations sur un objet.
- Une méthode comporte un en-tête et un corps.
- L'en-tête définit la *signature* de la méthode: **`public int getPrice()`**
- Le corps contient les instructions de la méthode.

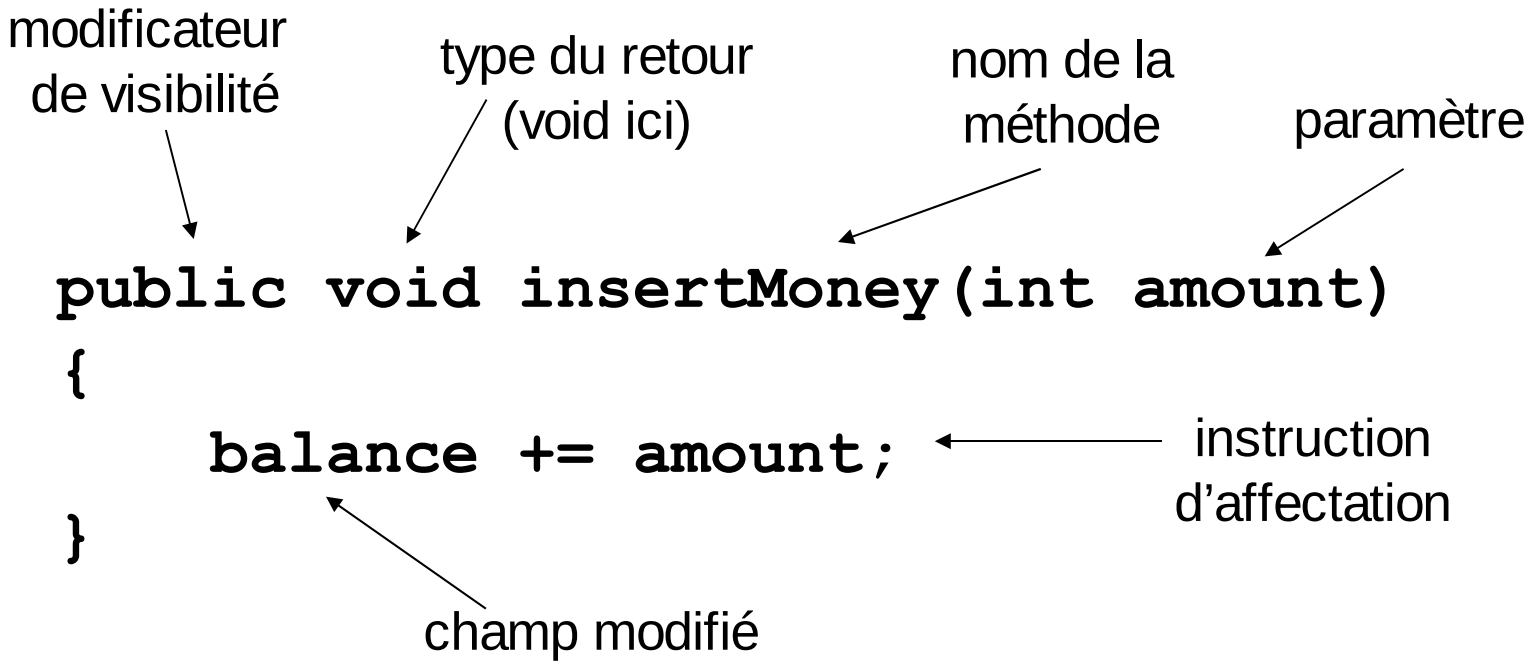
Méthodes d'accès (2)



Méthodes de modification (1) (modificateurs)

- Même structure: en-tête et corps.
- Utilisées pour *modifier* l'état d'un objet.
- Consistent à changer la valeur d'un ou plusieurs champs.
  - comportent typiquement des instructions d'affectation.
  - reçoivent en général des valeurs en paramètres.

Méthodes de modification (2)



Afficher depuis une méthode

```
public void printTicket()
{
    // Simule l'impression d'un ticket.
    System.out.println("####################");
    System.out.println("# La ligne BlueJ");
    System.out.println("# Ticket");
    System.out.println("# " + price + " centimes.");
    System.out.println("####################");
    System.out.println();

    // Mettre à jour le total collecté: ajouter le prix.
    total += balance;
    // Mettre à zéro la somme entrée.
    balance = 0;
}
```

Analyse des billetteries

- Leur comportement n'est pas satisfaisant pour plusieurs raisons:
  - pas de vérification des montants entrés.
  - pas de rendu de monnaie.
  - pas de vérification de l'initialisation.
- Comment faire mieux?
  - il nous faut des comportements plus sophistiqués.

Faire des choix (1)

```
public void insertMoney(int amount)
{
    if(amount > 0) {
        balance += amount;
    }
    else {
        System.out.println("Entrez un montant positif: " +
                                amount);
    }
}
```

Faire des choix (2)

Condition booléenne à tester
produit un résultat vrai (**`true`**) ou faux (**`false`**)

mot clé "**`if`**"

actions si la condition est vraie

```
if(perform some test) {
      Exécuter ces instructions si le test donne
      un résultat vrai
}
else {
      Exécuter ces instructions si le test donne
      un résultat faux
}
```

mot clé "**`else`**"

actions si la condition est fausse

Variables locales (1)

- Les champs sont une forme de variable.
  - ils stockent des valeurs pendant la vie d'un objet.
  - ils sont accessibles dans la classe.
- Les méthodes peuvent comporter des variables à durée de vie courte.
  - elles existent pendant l'exécution de la méthode.
  - elles ne sont accessibles que depuis la méthode.

Variables locales (2)

une variable locale

pas de modificateur de visibilité

```
public int refundBalance()
{
    int amountToRefund;
    amountToRefund = balance;
    balance = 0;
    return amountToRefund;
}
```

Résumé (1)

- les corps des classes contiennent des champs, des constructeurs et des méthodes.
- les champs stockent des valeurs qui définissent l'état d'un objet.
- les constructeurs initialisent les objets.
- les méthodes implantent le comportement des objets.

Résumé (2)

- champs, paramètres et variables locales sont tous des variables.
- les champs existent durant toute la vie d'un objet.
- les paramètres sont utilisés pour transmettre des valeurs aux constructeurs ou aux méthodes.
- les variables locales sont utilisées pour le stockage à courte durée de vie.

Résumé (3)

- Les objets peuvent prendre des décisions avec des instructions conditionnelles (`if`).
- Un test vrai/faux permet de choisir une suite d'exécution parmi deux.

Sommaire général