

Langages objets

Objets
(bloc 8)

M2 Pro CCI, Informatique
Emmanuel Waller, LRI, Orsay

résumé des épisodes précédents

- découverte Java, prise en main environnement, types primitifs
- opérateurs et expressions, instructions de contrôle, débogage
- tableaux
- fonctions

Objets

- Notion d'objet
- Définir la structure d'un objet
- Déclaration, création, utilisation
- représentation en mémoire
- Affectation d'objets
- Vocabulaire
- Répartition des classes en fichiers
- champs de classe et objets : rien de nouveau
- notion de constructeur
- Déroulement mémoire

remarque

- Cours à relire en parallèle avec cours tableaux : mêmes concepts, même structure

Notion d'objet

- Types primitifs (int, double, etc.)
- Tableaux : regroupement de valeurs de même type
- Objets :
 - But : Regroupement de valeurs de types différents
 - Construit par programmeur
 - Struct C + pointeur dessus
 - Zone mémoire + adresse de cette zone
 - format pour les bibliothèques Java (ex : String)

Exemple : manipuler des rationnels

- Créer un rationnel, l'afficher
- $r = n / d$
- Un rationnel possède deux attributs
 - Un numérateur, entier
 - Un dénominateur, entier
- Rem : dans tout programme : données, traitements

Exemple : avec tableaux

- Remarque : possible par tableau à deux éléments, mais peu lisible, nécessite convention (num., dén.)

```
class Parallele {
    public static void main(String[] args) {
        int [ ] r;
        r = new int [2];
        r[0] = 1;
        r[1] = 2;
        System.out.println(r[0] + « / » + r[1]);
    } // affiche 1 / 2
}
```

- peu lisible : pas noms : rationnel, num, den, etc.
- Démonstration (Ex1bis)

Objets : organisation du programme

- 2 parties : données, traitements
 - Déclarer la structure des rationnels :
 - Num : entier
 - Den : entier
 - Écrire un programme usuel les manipulant : il faut
 - créer l'objet
 - affecter ses champs
- Démonstration (Ex1)

Définir la structure d'un objet

- La classe Rationnel
 - est un morceau de programme
 - Contient déclaration structure objets : champs et types
- Num, den sont des champs (ou attributs, ou variables d'instance)
- Objet (ou instance) de la classe Rationnel =
 zone mémoire avec cette structure
 + adresse de cette zone

déclaration

- Déclaration d'une variable Rationnel :
Rationnel r;
- comme pour les tableaux (rappel) :
 - réserve zone mémoire (pour une adresse)
 - Pas de zone allouée pour les champs
- R ne pourra pas contenir un objet d'une autre classe
- La variable r contiendra l'adresse d'un objet Rationnel
- Types des champs : tous types Java

création

- new fonctionne comme pour les tableaux
- allocation zone mémoire
- Champs initialisés à la valeur par défaut du type
- pas d'initialisateur : solution plus riche : constructeur (vu ult.)

utilisation

- Accès aux champs : « . » (rappel : tableaux : par indice)
 - lecture, affectation
- les champs d'un objet se manipulent comme des variable ordinaires du même type (comme cases tableau)
- pas d'analogue du champ t.length des tableaux

Objets

- Notion d'objet
- Définir la structure d'un objet
- Déclaration, création, utilisation
- représentation en mémoire
- Affectation d'objets
- Vocabulaire
- Répartition des classes en fichiers
- champs de classe et objets : rien de nouveau
- notion de constructeur
- Déroulement mémoire

Objets : dans la mémoire

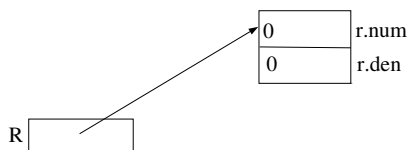
- Rappel : entier, tableaux : connu
- Rationnel r;
 - Réserve un emplacement mémoire pour une référence (une adresse)
 - Autrement dit : alloue des octets de la mémoire, associés à la variable r de la classe Rationnel
 - R inaccessible comme d'habitude
- new Rationnel()
 - Alloue l'emplacement nécessaire pour 2 entiers
- r = new Rationnel();
 - Place en plus adresse cet emplacement dans la case r

- Dit un peu autrement : `r = new Rationnel();`
 - `new` demande au système l'espace pour deux entiers
 - Le système l'alloue et en renvoie l'adresse
 - Elle est placée dans `r`
 - Comme pour les tableau
- autres formulations :
 - `r` reçoit la référence à l'objet créé par `new`
 - `r` est une variable de type classe

Rationnel r ;

101	102	r
102	0	r.num
103	0	r.den
104		

- L'autre représentation



exemple

- Adresse d'un objet


```
Rationnel r;
System.out.println(r); // compile pas, pas spécifique objets
r = new Rationnel();
System.out.println(r); // [Rationnel@46fa7002
// c'est le contenu de r : l'adresse mémoire, la vraie...
```
- Comme pour les tableaux
- Démonstration (Ex2bis)

vocabulaire

- Différents abus de langage :
 - Objet = référence
 - ex : « la variable r contient un objet », « r est un objet »
 - Objet = zone mémoire
 - ex : « r contient une référence à un objet »
- Aucune expression Java ne désigne un objet dans sa globalité :
 - Soit référence
 - Soit champs

exemple

- Ex2Memoire
 - Déroulement
 - démonstration

Objets

- Notion d'objet
- Définir la structure d'un objet
- Déclaration, création, utilisation
- représentation en mémoire
- Affectation d'objets
- Vocabulaire
- Répartition des classes en fichiers
- champs de classe et objets : rien de nouveau
- notion de constructeur
- Déroulement mémoire

Affectation d'objets

- Contenu (zone mémoire) : case par case obligatoirement (comme tableaux)

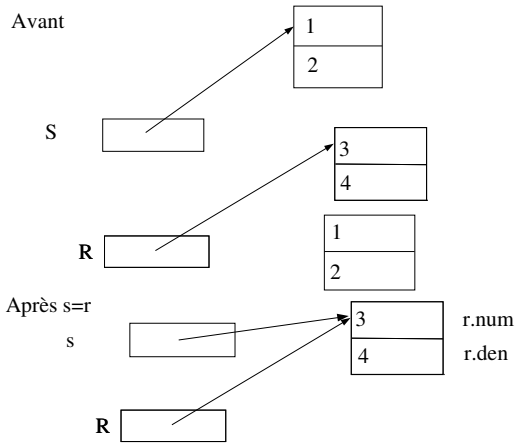
Recopie d'objet : erreur à éviter

- Comme les tableaux


```
Rationnel s = new Rationnel(); // ok
s.num = 1; s.den = 2;           // ok : 1 objet
Rationnel r = new Rationnel(); // ok
r.num = 3; r.den = 4;           // ok : 2 objets
s = r;                          // non : r n'a pas été dupliqué
```
- Pas eu de recopie des valeurs de l'objet
- Copie de référence
- objet référencé par s change pas, plus référencé par s

exemple

- Ex3 :
 - Déroulement
 - démonstration



cases

- En particulier :
Rationnel r = new Rationnel(), s;
r.num = 1; r.den = 2;
s = r;
s.num = 5;
System.out.println(r.num); // 5

Flèches

cases

exemple

- la bonne façon de recopier (comme tableaux) :
- Ex4 :
 - démonstration
 - Déroulement : travail personnel

Flèches

cases

<div data-bbox="92 78 651 129" data-label="Section-Header"> <h2>Déclaration d'objet : erreur à éviter</h2> </div> <div data-bbox="70 165 668 479" data-label="List-Group"> <ul style="list-style-type: none"> • Cas particulier de copie de référence inadéquate : <div data-bbox="105 212 414 365" data-label="Text"> <pre>Rationnel r = new Rationnel(); r = new Rationnel(); r.num = 1; r.den = 2;</pre> </div> • inutile : vous avez réservé deux tables au restaurant ; vous occupez la deuxième et la première est perdue </div>	<div data-bbox="869 49 956 82" data-label="Text"> <p>Flèches</p> </div>
<div data-bbox="70 627 132 660" data-label="Text"> <p>cases</p> </div>	<div data-bbox="869 669 1383 819" data-label="List-Group"> <ul style="list-style-type: none"> • le garbage collector (glaneur de cellules) <ul style="list-style-type: none"> – récupère la place définitivement perdue (et la mémoire ne s'encombre pas) – Mais ralentit l'exécution </div>
<div data-bbox="316 1153 419 1205" data-label="Section-Header"> <h2>Objets</h2> </div> <div data-bbox="59 1184 596 1648" data-label="List-Group"> <ul style="list-style-type: none"> • Notion d'objet • Définir la structure d'un objet • Déclaration, création, utilisation • représentation en mémoire • Affectation d'objets • Vocabulaire • Répartition des classes en fichiers • champs de classe et objets : rien de nouveau • notion de constructeur • Déroulement mémoire </div>	<div data-bbox="1101 1198 1236 1249" data-label="Section-Header"> <h2>exemple</h2> </div> <div data-bbox="869 1281 1417 1568" data-label="List-Group"> <ul style="list-style-type: none"> • Regroupement d'éléments de différents types <div data-bbox="903 1328 1074 1480" data-label="Text"> <pre>Class Personne { String nom; int age; }</pre> </div> • Voir aussi dernier exemple </div>
<div data-bbox="316 1713 419 1765" data-label="Section-Header"> <h2>Objets</h2> </div> <div data-bbox="59 1742 596 2206" data-label="List-Group"> <ul style="list-style-type: none"> • Notion d'objet • Définir la structure d'un objet • Déclaration, création, utilisation • représentation en mémoire • Affectation d'objets • Vocabulaire • Répartition des classes en fichiers • champs de classe et objets : rien de nouveau • notion de constructeur • Déroulement mémoire </div>	<div data-bbox="956 1758 1383 1809" data-label="Section-Header"> <h2>vocabulaire : type et classe</h2> </div> <div data-bbox="869 1841 1465 2121" data-label="List-Group"> <ul style="list-style-type: none"> • idée : le type d'une entité (objet, etc.) en décrit la structure, le format mémoire et les manipulations possibles • voici ce qu'on appelle type en Java : <ul style="list-style-type: none"> – les types primitifs – les classes – les types tableau : T [], pour tout type T </div>

vocabulaire : valeur

- les « valeurs » sont les éléments des types :
 - un entier, un booléen, etc.
 - un objet
 - un tableau
- type = ensemble de valeurs

Vocabulaire : tableau

- Tableau = zone + référence
- « un tableau est un objet »
- Sauf :
 - Éléments : de même type, de même nom, indicés
 - Différences subtiles seront découvertes lors utilisation avec autres concepts de la programmation orientée objets
- Fonctionnement en mémoire identique

Objets

- Notion d'objet
- Définir la structure d'un objet
- Déclaration, création, utilisation
- représentation en mémoire
- Affectation d'objets
- Vocabulaire
- Répartition des classes en fichiers
- champs de classe et objets : rien de nouveau
- notion de constructeur
- Déroulement mémoire

Répartition des classes en fichiers

- Les deux classes sont dans le même fichier (on peut faire autrement : hors module)
- javac crée les fichiers Rationnel.class et Application.class
- On doit appeler la classe qui contient la fonction main :
unix% java Ex
- démonstration

Objets

- Notion d'objet
- Définir la structure d'un objet
- Déclaration, création, utilisation
- représentation en mémoire
- Affectation d'objets
- Vocabulaire
- Répartition des classes en fichiers
- champs de classe et objets : rien de nouveau
- notion de constructeur
- Déroulement mémoire

objets et champs de classe : rien de nouveau

- rappel :
 - champs des objets : existent dans chaque objet
 - un champ de classe :
 - existe en un exemplaire unique pour la classe (qu'il existe ou non des objets)
- exactement comme sans objets

exemple

- gérer des personnes, avec en particulier le nombre de personnes créées au cours de l'exécution en cours du programme
- principe :
 - personnes : on sait faire
 - nombre de personnes :
 - unique pour toute la classe : champ statique
 - incrémenté par la fonction de création des personnes

```
class Personne {
    static int nbPersonnesCrees = 0;
    String nom;
    int age;
}
class Application {
    public static void main (String args) {
        Personne p = new Personne(); p.nom = "Jules"; p.age =25;
        nbPersonnesCrees++;
        System.out.println(Personne.nbPersonnesCrees);
        ... affichage p ...
    }
}
```

<div>démonstration</div> <ul style="list-style-type: none"> exemple 5 : on va faire la même chose sur les Rationnels 	<div>Méthodologie : champ de classe ou champ d'objet ?</div> <ul style="list-style-type: none"> – Valeur qui existe pour chaque objet : champ d'objet <ul style="list-style-type: none"> Ex : âge d'une personne chaque objet en a un différent – Valeur qui existe une seule fois pour tous les objets de la classe : champ de classe <ul style="list-style-type: none"> Ex : nombre total de personnes – Rem : possible répéter le nombre total de personnes dans chaque objet, mais inconvénients : <ul style="list-style-type: none"> Duplication inutile Lors modification, nécessite modifier tous les objets Donc ne jamais le faire
<div>notation</div> <ul style="list-style-type: none"> Si o objet de classe C et a champ de classe de C, possible : o.a, au lieu de C.a <ul style="list-style-type: none"> Mais moins clair : déconseillé... 	<div>remarque</div> <ul style="list-style-type: none"> Notion de propriété calculée : exemple : on ne stocke pas le prix d'un billet si on peut le calculer à partir du prix du kilomètre
<div>Objets</div> <ul style="list-style-type: none"> Notion d'objet Définir la structure d'un objet Déclaration, création, utilisation représentation en mémoire Affectation d'objets Vocabulaire Répartition des classes en fichiers champs de classe et objets : rien de nouveau notion de constructeur Déroulement mémoire 	<div>la notion de constructeur</div> <ul style="list-style-type: none"> un constructeur est une fonction (spéciale) de même nom que la classe il est appelé automatiquement après new et l'initialisation des champs il peut contenir des instructions quelconques
<pre>class Point { ... Point (int abs, int ord) { // pas void x = abs; y = ord; } } class Application/main Point a = new Point(7,3); ... affichage de a ... Point b = new Point(); // erreur compil : plus possible Point(4, 5); // erreur compilation : interdit appeler</pre>	<div>démonstration</div> <ul style="list-style-type: none"> on reprend l'exemple ci-dessus exemple 6 exemple 7 : la seule bonne manière

<div>Objets</div> <ul style="list-style-type: none"> • Notion d'objet • Définir la structure d'un objet • Déclaration, création, utilisation • représentation en mémoire • Affectation d'objets • Vocabulaire • Répartition des classes en fichiers • champs de classe et objets : rien de nouveau • notion de constructeur • Déroulement mémoire 	<div>exemple</div> <ul style="list-style-type: none"> • Programme avec 2 sortes d'objets • Ex8 : démonstration
<div>Déroulement mémoire</div> <ul style="list-style-type: none"> • exemple 	<div>quel rapport avec C ?</div> <ul style="list-style-type: none"> • toujours rien de nouveau : un objet n'est qu'un pointeur sur un struct • seule la présentation en plusieurs classes est un « habillage différent » des mêmes concepts
<div>Objets</div> <ul style="list-style-type: none"> • Notion d'objet • Définir la structure d'un objet • Déclaration, création, utilisation • représentation en mémoire • Affectation d'objets • Vocabulaire • Répartition des classes en fichiers • champs de classe et objets : rien de nouveau • notion de constructeur • Déroulement mémoire 	<div>récapitulatif : objets (Delannoy 6.1, 6.2, 6.4)</div>
<div>délégués ?</div>	