Mémoire

La classe "encapsule"

- Stockage unique, passage "par pointeur"
- Objets partagés

Avantages

- Permet de stocker un grand nombre de variables d'instance
- Peut se manipuler comme une structure C
- Avec les accesseurs, on peut protéger les données

Tableau

- String args [] = String [] args
- args[n] retourne ne nième elément de args
- args[n] = valeur stocke valeur au rang n de args

Objet?

- args = new String [size] pour la création
- args.length retourne la longueur du tableau

Limites

- Pas redimentionnable
- Pas d'affectation dynamique

La classe Array

- java.lang.Array
- “encapsule les []”
- autorise des conversions entre les différents types de scalaires

La classe java.util.Vector

- import java.util.Vector;
- “Array” dynamique
- ne stocke que des objets

Avantages

- taille dynamique
- objet protégé en mémoire
- sérialisable (voir plus tard)

Utilisation

- Vector v = new Vector();
- v.add(objet)
- v.get(index)
- v.size();

Caveat

- ordonné
- stockage comme “Objects”, attention aux casts!
- peut contenir plusieurs fois le même élement

Un exemple

```
Vector fib = new Vector();
int suiteIndex = 0;
while(suiteIndex < 100) {
  fib.add(Fibonacci.calcul(new Entier
(suiteIndex++));
}

System.out.println(fib.toString());
```

Matrices

- Matrices a taille fixe:
  String[][] mat = new String [10][10];
  mat[2][9] = “hop”;

- Matrices d’objets:
  Vector qui contient des Vector

The Usual Suspects

- Piles (classe Stack)
- Files (Collection et Enumerator)

Dictionnaires

- Association clef <-> valeur
- la clef peut être n'importe quel objet (testée avec isEqual())
- la valeur peut être n'importe quel objet (testée avec isEqual())

La classe Hashtable

- java.util.Hashtable
- size()
- put(key, value)
- get(key)

Caveat

- stockage comme “Objects”, attention aux casts!
- les clefs et les valeurs sont testées avec isEqual() : si vous voulez un test particulier, vous devez surcharger
- peut contenir autant de fois que vous voulez la même valeur, mais toutes les clefs doivent être différentes

Mécanisme

- put(key, value) pour une clef existante **remplace** l'ancienne valeur associée
- Si vous voulez des valeurs multiples pour une clef mettez un Vector
- Les types de valeurs peuvent être différents

Un exemple

```
Hashtable dic = new HashTable();
int suiteIndex = 0;
while(suiteIndex < 100) {
  Entier n = new Entier(suiteIndex++);
  dic.put(n, Fibonacci.calcul(n));
}

System.out.println(dic.toString());
```

Pourquoi?

- On peut tester si un objet est déja présent dedans : contains() (et containsKey/ containsValue)
- Taille adaptative à vos besoins
- Objets