Plan du cours

1/ Rappels et compléments Java.

2/ Tableaux, boucles et invariants.

3/ Notions élémentaires de complexité.

4/ Récursion.

5/ Structures de données et introduction aux types abstraits de données. **ICI**

6/ Quelques compléments Java.

Important

La **représentation des données** nécessaires à la résolution d'un problème est une étape très **importante**.

Important

La **représentation des données** nécessaires à la résolution d'un problème est une étape très **importante**.

Définition d'un Type de Données Abstrait

C'est la spécification des opérations permettant de manipuler les objets du type de données.

Cette spécification correspond à un **cahier de charges** qu'on doit ensuite **implémenter** (par exemple en JAVA).

Type de Données Abstrait et Structures de Données

Important

La **représentation des données** nécessaires à la résolution d'un problème est une étape très **importante**.

Définition d'un Type de Données Abstrait

C'est la spécification des opérations permettant de manipuler les objets du type de données.

Cette spécification correspond à un **cahier de charges** qu'on doit ensuite **implémenter** (par exemple en JAVA).

Définition détaillée

Un **type abstrait** est défini par :

- un **nom** ;
- un ensemble de **constructeurs constantes** ou **fonctions** qui permettent de produire des données du type abstrait à partir d'autres données ;
- des **opérateurs** qui permettent de manipuler les données du type abstrait ;
- parfois des **invariants** et **propriétés** des opérateurs qui caractérisent les données du type abstrait.

Un exemple

Type booléen
Opérations
vrai : -> booléen
faux : -> booléen
Non : booléen -> booléen
Et : booléen x booléen -> booléen
Ou : booléen x booléen -> booléen
OuExclusif : booléen x booléen -> booléen
Propriétés
pour tout a, b booléens, nous avons
NON(vrai) = faux
vrai ET a = a
faux ET a = faux
a OU b = NON(NON(a) ET NON(b))

Un exemple

Type booléen
Opérations
vrai : -> booléen
faux : -> booléen
Non : booléen -> booléen
Et : booléen x booléen -> booléen
Ou : booléen x booléen -> booléen
OuExclusif : booléen x booléen -> booléen
Propriétés
pour tout a, b booléens, nous avons
NON(vrai) = faux
vrai ET a = a
faux ET a = faux
a OU b = NON(NON(a) ET NON(b))

Mise en œuvre

Lors de l'implantation, il nous faut décrire l'organisation des données et écrire les algorithmes réalisant les opérations du type de données abstrait. (ex: tableau, liste chaînée, tableau de booléens, ...)

Le type de données abstrait PILE

Définition du type Pile de X (ici X est un type quelconque)

- **`Nom du type`** : Pile[X]
- **`Opérateurs`** (signature des méthodes)
  - Création. Pile :-> Pile[X]
  - Test. estVide: Pile[X] -> Booléen
  - empiler: XxPile[X] -> Pile[X]
    dépiler: Pile[X] -> XxPile[X]
    sommetPile: Pile[X] -> X
- **`Propriétés des opérateurs:`**
  - préconditions:
    dépiler(X): X ne doit pas être vide (utiliser le test au préalable!)
  - axiomes:
    - Pour tout x de X pour tout s de Pile[X]
    - estVide(Pile()) (une pile créée est initialement vide)
    - non estVide(empiler(x,s))
    - dépiler(empiler(x,s))=(x,s)

Le type de données abstrait PILE

Propriété

L'élément renvoyé par sommetPile[X] est le dernier élément empilé. La Pile est une structure **LIFO** ("Last In First Out").

Dessin

Décrire schématiquement les instructions suivantes (pile d'entiers):
P = Pile()
empiler(P,1)
empiler(P,2)
depiler(P)
renvoyer(sommetPile(P))

Implémentation des Piles par les tableaux en JAVA

Interface Java

Une interface en Java est composée de déclarations de méthodes mais **`sans définition de méthodes`** ni **`présence de variables d'instance`**.

Interface Java pour une Pile

On va utiliser la **`généricité de Java`** et donc le type Object (qui peut contenir n'importe quel objet du type référence). On va construire un type Pile de Object.

Implémentation des Piles par les tableaux en JAVA

Interface Java

Une interface en Java est composée de déclarations de méthodes mais **sans définition de méthodes** ni **présence de variables d'instance**.

Interface Java pour une Pile

On va utiliser la **généricité de Java** et donc le type Object (qui peut contenir n'importe quel objet du type référence). On va construire un type Pile de Object.

On a donc pour notre pile **générique** :

Interface Pile simple

```
interface Pile {
        public Object depiler();
        public void empiler(Object o);
        public boolean estVide();
}
```

Remarques sur les interfaces et la POO (1/2)

Les interfaces

- une interface I ne contient que des déclarations de méthodes (abstraites)

Remarques sur les interfaces et la POO (1/2)

Les interfaces

- une interface I ne contient que des déclarations de méthodes (abstraites)
- une classe C (concrète) qui implémente une interface doit définir les méthodes déclarées dans l'interface: la signature et la valeur retournée de ces méthodes doivent être identiques pour l'interface et la classe qui l'implémente

Remarques sur les interfaces et la POO (1/2)

Les interfaces

- une interface I ne contient que des déclarations de méthodes (abstraites)
- une classe C (concrète) qui implémente une interface doit définir les méthodes déclarées dans l'interface: la signature et la valeur retournée de ces méthodes doivent être identiques pour l'interface et la classe qui l'implémente
- on peut déclarer des variables de type I, mais on ne peut pas créer d'objet pour une interface I

Remarques sur les interfaces et la POO (1/2)

Les interfaces

- une interface I ne contient que des déclarations de méthodes (abstraites)
- une classe C (concrète) qui implémente une interface doit définir les méthodes déclarées dans l'interface: la signature et la valeur retournée de ces méthodes doivent être identiques pour l'interface et la classe qui l'implémente
- on peut déclarer des variables de type I, mais on ne peut pas créer d'objet pour une interface I
- tout objet de C qui implémente I peut être considéré comme étant de type I

Remarques sur les interfaces et la POO (1/2)

Les interfaces

- une interface I ne contient que des déclarations de méthodes (abstraites)
- une classe C (concrète) qui implémente une interface doit définir les méthodes déclarées dans l'interface: la signature et la valeur retournée de ces méthodes doivent être identiques pour l'interface et la classe qui l'implémente
- on peut déclarer des variables de type I, mais on ne peut pas créer d'objet pour une interface I
- tout objet de C qui implémente I peut être considéré comme étant de type I
- une variable déclarée de type I contenant un objet de type C qui implémente I peut être affectée à une variable de type C à condition d'explicitement faire un "cast" ((C)).

Remarques sur les interfaces et la POO (1/2)

Les interfaces

- une interface I ne contient que des déclarations de méthodes (abstraites)
- une classe C (concrète) qui implémente une interface doit définir les méthodes déclarées dans l'interface: la signature et la valeur retournée de ces méthodes doivent être identiques pour l'interface et la classe qui l'implémente
- on peut déclarer des variables de type I, mais on ne peut pas créer d'objet pour une interface I
- tout objet de C qui implémente I peut être considéré comme étant de type I
- une variable déclarée de type I contenant un objet de type C qui implémente I peut être affectée à une variable de type C à condition d'explicitement faire un "cast" ((C)).
- l'association du nom d'une méthode de l'interface avec la méthode concrète qui l'implémente se fait à l'exécution suivant le type de l'objet.

Remarques sur les interfaces et la POO (1/2)

Les interfaces

- une interface I ne contient que des déclarations de méthodes (abstraites)
- une classe C (concrète) qui implémente une interface doit définir les méthodes déclarées dans l'interface: la signature et la valeur retournée de ces méthodes doivent être identiques pour l'interface et la classe qui l'implémente
- on peut déclarer des variables de type I, mais on ne peut pas créer d'objet pour une interface I
- tout objet de C qui implémente I peut être considéré comme étant de type I
- une variable déclarée de type I contenant un objet de type C qui implémente I peut être affectée à une variable de type C à condition d'explicitement faire un "cast" ((C)).
- l'association du nom d'une méthode de l'interface avec la méthode concrète qui l'implémente se fait à l'exécution suivant le type de l'objet.

Remarques sur les interfaces et la POO (2/2)

Mécanisme de la POO

Ces mécanismes sont à la base de la programmation orientée objets.
Ils se généralisent en Java à d'autres constructions que les interfaces.
Pour l'essentiel on a les deux notions liées:

Remarques sur les interfaces et la POO (2/2)

Mécanisme de la POO

Ces mécanismes sont à la base de la programmation orientée objets. Ils se généralisent en Java à d'autres constructions que les interfaces. Pour l'essentiel on a les deux notions liées:

- **`héritage`** : un objet de type A peut être considéré comme étant de type B (similaire à: un objet de type C implémentant une interface I peut être considéré comme un I). De cette manière une variable déclarée d'un type B peut en fait contenir un objet de type A.
- **`liaison dynamique`** : l'association d'un nom à une méthode se fait à l'exécution suivant le type de l'objet référencé et non suivant le type de la déclaration.

Implémentation d'une Pile avec des tableaux

Pile et tableaux

```
class PileTable implements Pile {
        private Object[] v;
        private int taillemax=1000;
        private int sommet;
       public PileTable() {
              v=new Object[taillemax];
               sommet=0;
       }
       public PileTable(int n){
              taillemax=n;
              v=new Object[taillemax];
               sommet=0;
       }
       public boolean estVide(){return sommet==0; }
       public Object depiler(){return v[–sommet];}
       public void empiler(Object o){v[sommet++]=o;}
}
```

Exemples d'utilisation d'une pile (1/2)

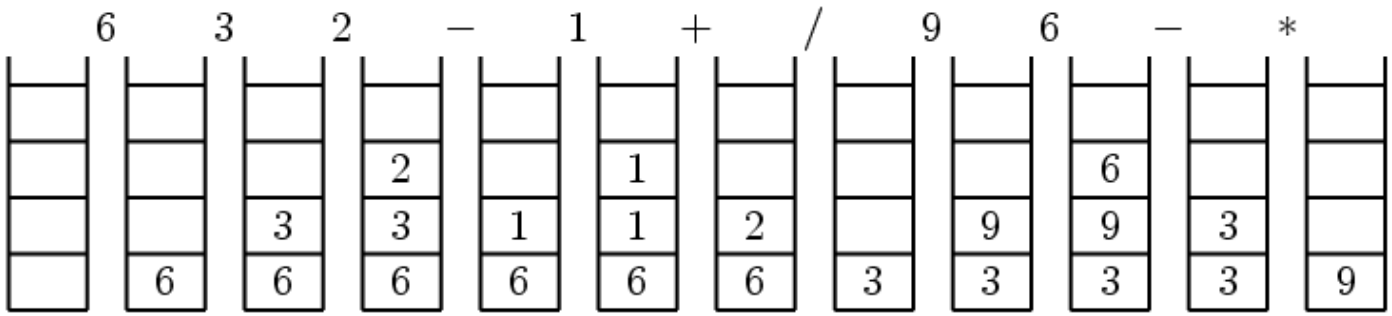
On veut calculer l'expression **postfixe** suivante:

$$6\ 3\ 2\ -\ 1\ +\ /\ 9\ 6\ -\ \star$$

Exemples d'utilisation d'une pile (1/2)

On veut calculer l'expression **postfixe** suivante:

$$6\ 3\ 2\ -\ 1\ +\ /\ 9\ 6\ -\ \star$$



Exemples d'utilisation d'une pile (2/2)

Vérification de parentèses, crochets, etc ...