ITI 1521. Introduction à l'informatique II

Pile : concept

by

Marcel Turcotte

Version du 2 février 2020

Préambule

Préambule

Aperçu

Aperçu

Pile : concept

Nous nous intéressons à tous les aspects des **piles** en programmation. Une pile est un **type abstrait de données** semblable aux piles physiques. C'est une structure de données **linéaire** telle que l'**accès aux données ne se fait que d'une extrémité**, appelée le dessus de pile, et **un élément à la fois**.

Objectif général :

- Cette semaine, vous serez en mesure de décrire une pile.

Préambule

Objectifs d'apprentissage

Objectifs d'apprentissage

- **Décrire** le concept de pile en informatique.
- **Justifier** le rôle d'une pile dans la résolution d'un problème informatique.
- **Implémenter** une pile à l'aide d'un tableau.

Lectures :

- Pages 147-150 de E. Koffman et P. Wolfgang.

Préambule

Plan du module

Plan

Théorie

Théorie

Discussion

Discussion

- Discutez l'implémentation de mécanismes pour annuler (en anglais « *undo* ») et rétablir (« *redo* ») les opérations dans un éditeur de texte ?
  - Quelles informations devez-vous sauvegarder ?
  - Dans quel ordre ces informations sont sauvegardées et accédées ?
- Qu'ont en communs ces mécanismes avec ceux d'un logiciel de navigation (en anglais « *browser* ») permettant le retour à une adresse URL antérieure ou déjà visitée ?

Théorie

Définition

Théorie

Une **pile** (en anglais « *stack* ») est un **type abstrait de données** semblable aux piles physiques.

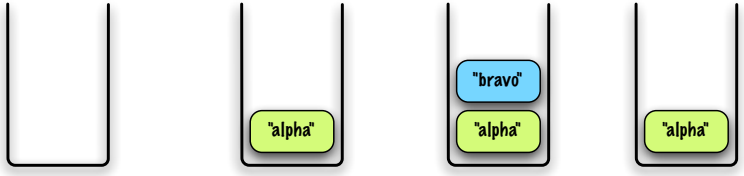
- Livres
- Assiettes
- Cabarets
- PEZ

L'analogie avec le distributeur d'assiettes que l'on retrouve dans une cafétéria est particulièrement intéressante, parce que **l'accès est limité à l'élément du dessus** et **il faut enlever une à une les assiettes du dessus afin d'accéder à celles du dessous.**

Définition

Une **pile** est une structure de données **linéaire** telle que l'accès aux données ne se fait que d'une extrémité, appelé le **dessus** de pile (en anglais « *top* »), et un élément à la fois.

On appelle souvent cette structure **LIFO**, de l'anglais « *last-in first-out* ».



Théorie

Applications

Applications

Les piles sont les structures de données derrière les applications suivantes :

- le développement de **compilateurs**, et l'analyse de langages formels en général ;
- l'implémentation d'**algorithmes de retour-arrière** (*backtracking*) ; utilisés par les systèmes de preuve automatique de théorèmes (**Prolog**), les algorithmes de jeux et l'intelligence artificielle ;
- la gestion de la mémoire lors de l'**exécution de programmes** ;
- supporter les opérations « **undo** » dans les programmes de traitement de textes ou encore le bouton « **back** » de votre navigateur Internet (**fureteur**).

Théorie

Opérations

Opérations

Les **opérations de base** sont :

push : **ajouter** un élément sur la pile

pop : **enlever** et **retourner** l'élément du dessus

isEmpty : vérifier que la pile **est vide**

Type abstrait de données (TAD)

```
public interface Stack<E> {
    E push(E elem);
    E pop();
    E peek();
    boolean isEmpty();
}
```

Comme on l'a vue au dernier cours, l'interface peut recevoir un paramètre de type !

```
class Mystery {
    public static void main(String[] args) {

        Stack<String> stack;
        stack = new StackImplementation<String>();

        for (int i=0; i<args.length(); i++) {
            stack.push(args[ i ]);
        }

        while (! stack.empty()) {
            System.out.print(stack.pop());
        }
    }
}
```

Que produit «`java Mystery a b c d e`» ?

Remarques

- Les éléments retirés apparaissent dans l'ordre inverse
- La construction suivante est fréquente dans l'utilisation des piles :

```
while (! stack.empty()) {
    element = stack.pop();
    ...
}
```

- Il faut faire attention de ne pas boucler à l'infini, par exemple en omettant le **pop()**.

Opérations (suite)

peek : **retourne** la valeur de l'élément du dessus **sans le retirer** ;

Implémentation à l'aide d'un tableau

Implémentation d'une pile à l'aide d'un tableau

Implémentation à l'aide d'un tableau

Objectifs d'apprentissage :

- **Implémenter** une pile à l'aide d'un tableau.

Lectures :

- Pages 70–75 de E. Koffman et P. Wolfgang.

Implémentations

Pensez à une implémentation possible ?

Il y a **deux** grandes familles d'implémentations :

- à base de **tableaux**
- à base d'**éléments chaînés**

```
Stack<Integer> s;

s = new ArrayStack<Integer>();
s = new DynamicArrayStack<Integer>();
s = new LinkedStack<Integer>();
```

Discussion

- L'une des implémentations proposées utilise un tableau, **pourquoi n'utilise-t-on pas tout simplement un tableau** pour la conception d'algorithmes ? **Quels sont les avantages** ?

Implémentation à l'aide d'un tableau

Variables d'instance

Implémentation à l'aide d'un tableau : ArrayStack

- Quelles sont les variables d'instance ?
- Quel est le type des éléments de ce tableau ?

Implémentation à l'aide d'un tableau

Stratégie

Implémentation à l'aide d'un tableau : stratégie

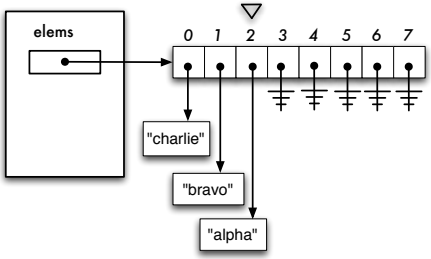
Quelle sera la stratégie adoptée afin de sauvegarder des éléments dans la pile ?

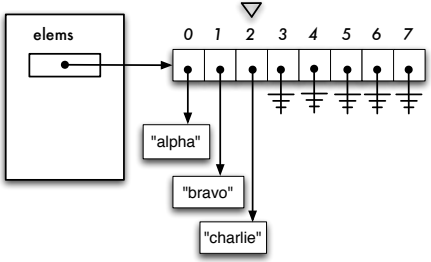
- Dans quel ordre sauvegarderez-vous les éléments ?
- Où ajouter un nouvel élément (**push**) et quel élément retiré (**pop**) ?

Prenez quelques minutes afin d'élaborer une stratégie.

Implémentation à l'aide d'un tableau : ordre des éléments

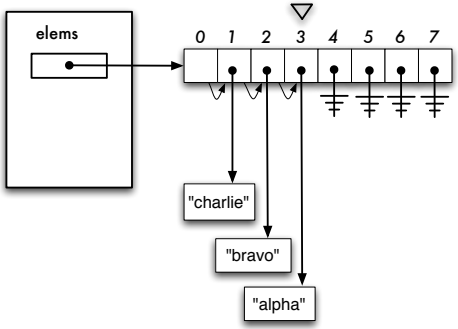
- Dans quel **ordre** ajouter les éléments ?
- Est-ce **important** ?

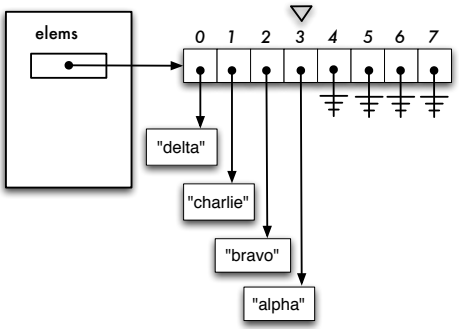




Implémentation à l'aide d'un tableau : dessous à droite

Qu'en pensez-vous ?





Implémentation à l'aide d'un tableau : trouver une cellule vide

Proposition. Qu'en pensez-vous ?

- La méthode **push** visite chacune des cellules du tableau, de gauche à droite, et insère le nouvel élément dans la première cellule dont la valeur est **null**.
- La méthode **pop** visite chacune des cellules du tableau, de droite à gauche, et retire l'élément dans la première cellule dont la valeur **n'est pas null**.

Implémentation à l'aide d'un tableau : ArrayStack

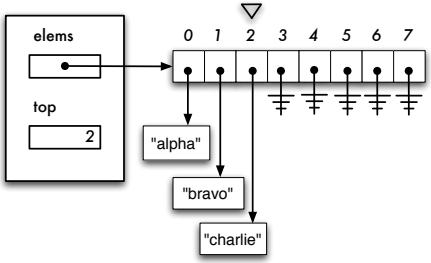
- Idéalement, la pile devrait **mémoriser** l'index de la cellule occupée la plus à droite.
- En programmation orientée objet, comment un objet, ici une pile, **mémorise**-t-il une valeur ?

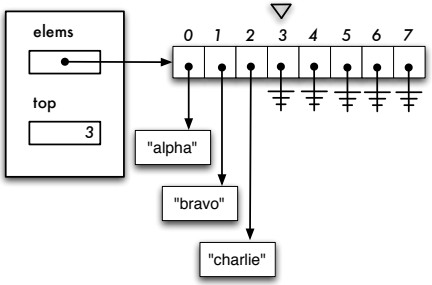
Résumé

- Nous utilisons un **tableau** afin de sauvegarder les éléments de la pile
- L'**ordre** des éléments est **important**
  - L'**élément du dessus** sera celui le **plus à droite**
- Il nous faut **deux** variables d'instance
  - L'une d'elles est une **référence** vers un tableau
  - L'autre variable nous permet de localiser l'**élément du dessus**

Implémentation à l'aide d'un tableau : top

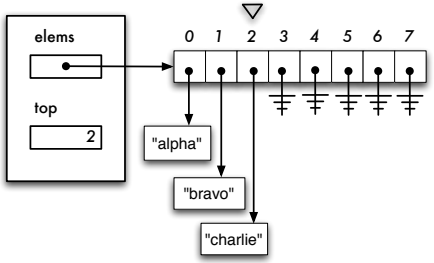
- Est-ce que la variable **top** désigne l'élément du dessus ou la cellule vide qui suit l'élément du dessus ?
- Quel est le **type** de cette variable ?





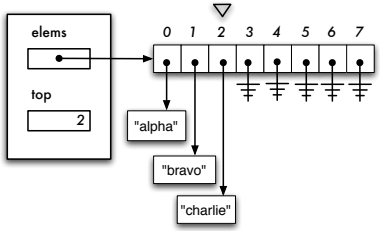
Ajout d'un élément lorsque top désigne l'élément du dessus

Quelles sont les **étapes** pour l'**ajout** d'un élément ?



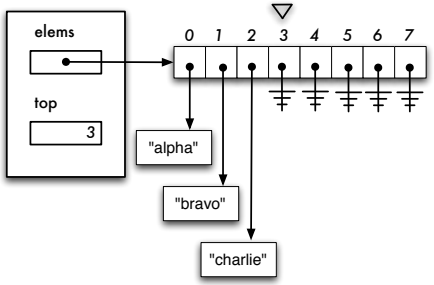
La variable top contient l'index de l'élément du dessus

Quelle est la valeur de **top** si la pile est **vide** ?



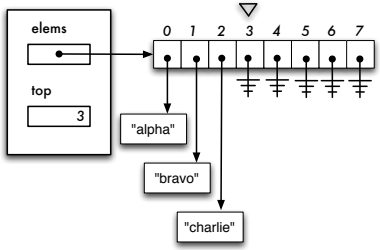
Ajout d'un élément lorsque top désigne la cellule vide qui suit l'élément du dessus

Quelles sont les **étapes** pour l'**ajout** d'un élément ?



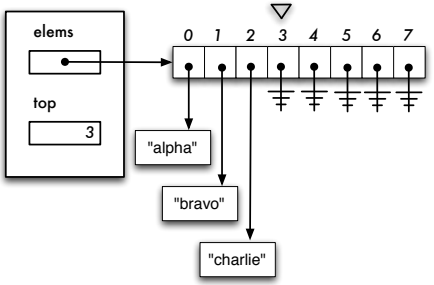
La variable top désigne la cellule vide qui suit l'élément du dessus

Quelle est la valeur de **top** si la pile est **vide** ?

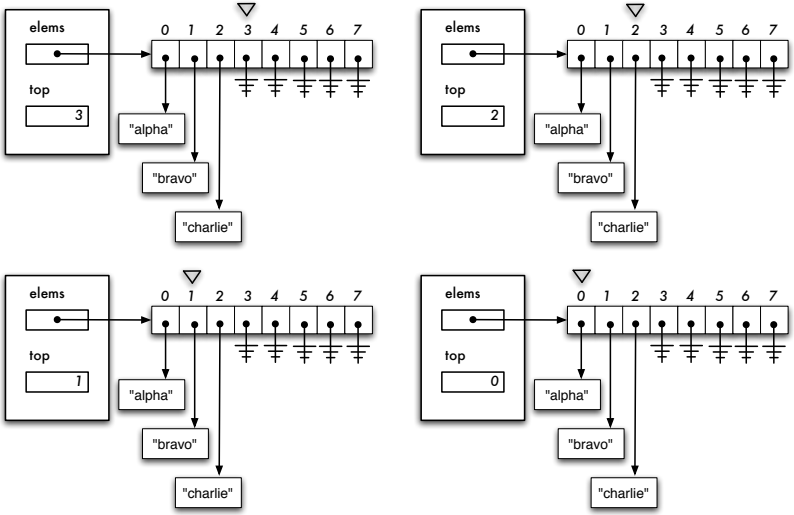


Retirer un élément lorsque top désigne le nombre d'éléments

Quelles sont les **étapes** pour **retirer** un élément ?



Qu'en pensez-vous ?



Implémentation à l'aide d'un tableau

Fuites de mémoire

Fuite de mémoire (en anglais « memory leak »)

Étapes pour retirer un élément :

- Décrémenter la valeur de **top**
- Sauvegarder dans une variable locale (temporaire) l'élément du tableau se trouvant à la **position** désignée par **top**
- Retourner la valeur sauvegardée

L'approche proposée donne lieu à des **fuites de mémoire** !

En effet, le **ramasse-miettes** (an anglais « *garbage collector* ») ne peut récupérer l'espace mémoire associé aux objets **accessibles**.

Quelle solution proposez-vous ?

Implémentation à l'aide d'un tableau

Implémentation en Java

Concevoir un type générique : ArrayStack<E>

Exemples d'**utilisation** de la pile :

```
Stack<String> s1;
name = new ArrayStack<String>(100);

Stack<Time> s2;
name = new ArrayStack<Time>(1024);

s1.push("alpha");
s2.push(new Time( 23,0,0 ));

String a;
a = s1.pop();
```

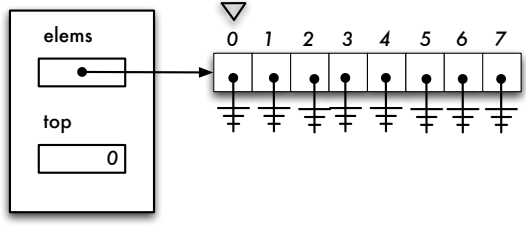
Implémentation à l'aide d'un tableau : ArrayStack

- Implémentation en **Java** d'une pile à l'aide d'un tableau.
- Ici, **top** désigne la **cellule vide** qui suit l'élément du dessus.
- L'implémentation où **top** désigne l'élément du dessus est laissée en exercice.

Implémentation à l'aide d'un tableau : ArrayStack

Implémentation à l'aide d'un tableau : ArrayStack

Donnez l'implémentation du **constructeur** qui produira le diagramme de mémoire ci-dessous :



Donnez le **diagramme de mémoire** correspondant à l'exécution du constructeur ci-dessous :

```
public class ArrayStack implements Stack {
    private E[] elements;
    private int top;

    public ArrayStack(int capacity) {
        top = 0;
    }
}
```

Tableaux et types génériques

ArrayStack et types génériques

- Afin que les programmes puissent tourner sur des **machines virtuelles de générations précédentes (avant 1.5)**, on ne peut créer des tableaux dont le type des éléments est générique.
- Nous avons tout de même **un problème à résoudre**.

ArrayStack et types génériques

Solution :

```
public class ArrayStack<E> implements Stack<E> {
    private E[] elems;
    private int top;

    public ArrayStack( int capacity ) {
        elems = (E[]) new Object[ capacity ];
        top = 0;
    }
    // ...
```

Hélas, cette solution produira une **alerte lors de la compilation**.

```
Note: ArrayStack.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

ArrayStack et types paramétrés

Directive locale afin de supprimer une alerte lors de la compilation :

```
public class ArrayStack<E> implements Stack<E> {
    private E[] elems;
    private int top;

    @SuppressWarnings( "unchecked" )
    public ArrayStack( int capacity ) {
        elems = (E[]) new Object[ capacity ];
        top = 0;
    }
    // ...
```

Ce qui est préférable à la **suppression globale des alertes** !

```
> javac -Xlint:unchecked ArrayStack.java
```

ArrayStack<E> : isEmpty()

ArrayStack<E> : E peek()

ArrayStack<E> : void push(E element)

ArrayStack<E> : E pop()

ArrayStack : isEmpty()

ArrayStack : isEmpty() (prise 2)

ArrayStack : E peek()

Implémentation à l'aide d'un tableau

Tableaux dynamiques

Faiblesse de notre implémentation

- Notre implémentation à une **faiblesse majeure**, quelle est-elle ?

Tableaux dynamiques

Certains langages de programmation nous permettent de **changer la taille des tableaux** lors de l'**exécution** des programmes. Ces langages utilisent la technique ci-dessous.

Implémentation à l'aide d'un tableau

Piège

Piège !

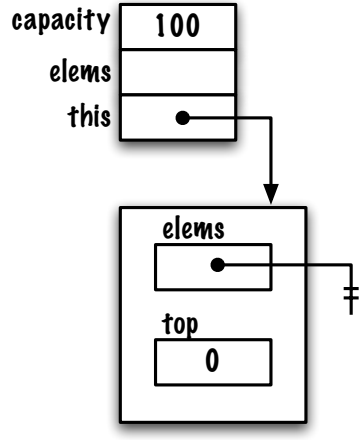
```
public class ArrayStack<E> implements Stack<E> {

    // Instance variables
    private E[] elems;
    private int top;

    // Constructor
    public ArrayStack( int capacity ) {
        E[] elems = (E[]) new Object[ capacity ];
        top = 0;
    }
    // Returns true if this ArrayStack is empty
    public boolean isEmpty() {
        return top == 0;
    }
    // ...
}
```

Piège

Working memory for ArrayStack

formal parameter(s)

local variable(s)

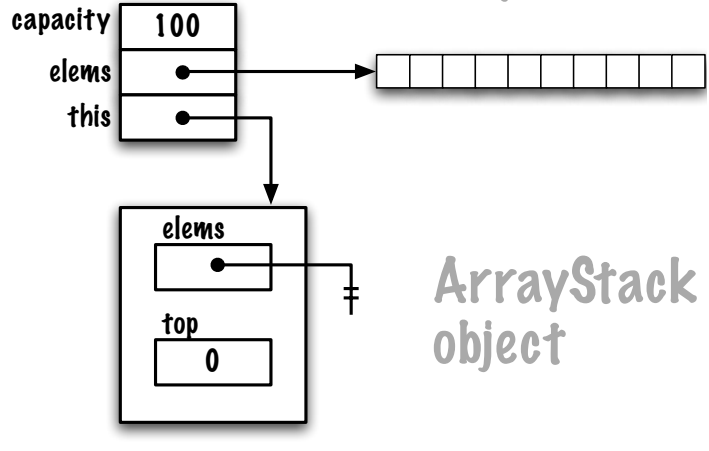


ArrayStack object

Activation Frame for ArrayStack

formal parameter(s)

local variable(s)



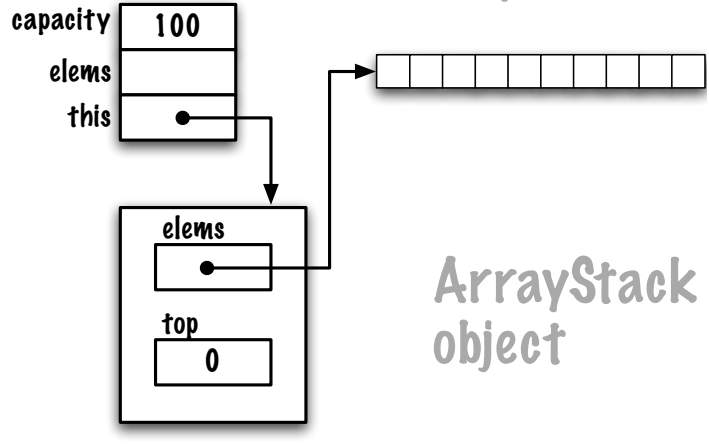
ArrayStack object

Piège

Activation Frame for ArrayStack

formal parameter(s)

local variable(s)



ArrayStack object

Prologue

Résumé

- On utilise **une pile** lorsqu'on souhaite traiter les éléments **dans l'ordre inverse**.
- Il existe deux implémentations principales : **à l'aide d'un tableau** et **à l'aide d'éléments chaînés.**

Prochain module

- **Pile** : applications

References I


E. B. Koffman and Wolfgang P. A. T.
Data Structures : Abstraction and Design Using Java.
John Wiley & Sons, 3e edition, 2016.

Marcel **Turcotte**

Marcel.Turcotte@uOttawa.ca

École de **science informatique** et de génie électrique (SIGE)

Université d'Ottawa