ANNEXE 17 - Les collections – Stack , Vector , List, etc.

Stack/pile :

Un peu comme un liste chaine LIFO last in first out

Pour rajouter une assiette on push

Pop

File :

FIFO first in firt out

## Les Vectors

Size :nb élément

Capacity: nb d’espace ou place

A habituellement 10 place de 0 à 9

Si on met un pion a l’indice 0 et on veut mettre le roi dans l’indice 0 aussi le pion va se tasser a l’indice 1

* Une Stack est une mise en œuvre simple d’une classe plus générale : Vector
* Un Vector vise à représenter un tableau d’objets, chaque item du tableau pouvant être accessible à partir de son index dans le tableau.
* Cependant, la capacité d’un Vector peut s’agrandir au fur et à mesure qu’on ajoute des items dans le Vector 🡪 allocation dynamique de mémoire
* Attention, il ne peut pas avoir d'éléments vides entre des éléments occupés à l'intérieur du Vector

EX. :

***Constructeurs :***

Vector premier = new Vector(); avec une capacite initiale de 10 places, s’il est plein on double la capacité (10,20,40,80)

Vector deuxieme = new Vector(2);on créer un vecteur avec une capacité initial deux places, si plein on doubles la capacité

Vector deuxieme = new Vector(2.1);on créer un vecteur avec une capacité initial deux places, si plein on ajoute une places

Vector<Double> troisieme = new Vector(3);créer un vecteur de capacité initial de 3 places et ne pouvons contenir que des objets de type Doubles

Classe enveloppe/wrapper classe

Int -- integer

double – Double

float – Float

char—Character

c’est un objet avec une seul varaible d’instance qui est l’objet. Si on a int 5 c,est objet Interger qui a 5 dedans à l’indice 0, car des vector ne peut seulement avoir des objet

***Méthodes principales :***

void add ( Object o ) : Permet d’ajouter l’objet o à la fin des élémemnt déjà présent dans le vecteur

void insertElementAt ( Object o, int index ) : Permet d’ajouter o à l’indice donner (index) on augment alors, les indices des éléments subsequent de 1, ils se tasse pour mettre le vecteur a l’indice 0. L’infex doit être entre 0, ne laisse pas de trou, s’il n’y a rien a l’indice 4 on ne peut rien mettre a l’indice 5.

Object get ( int index ) : retourner une référence à l’objet présent dans le vecteur a cette indice ( sans l’enlever)

Object remove ( int index): Permet d’enlver l’objet a l’indice et retranche 1 aux indices des éléments suivant ( pas de troue)

boolean remove ( Object o ): enlve la premiere occurrence identique a l’objet en utilisant equals et retourne true ou false s’il a trouve ou non et retire l’objet

boolean contains (Object o ): verifie si l’objet o est present dans le vecteur true or flase, mais ne le retire pas. En utilisant equals

void trimToSize(): s’il y a des espaces vides il va dimuner la taill du vecteur selon la taille des espaces pleins

EXEMPLES :

Les trois vecteurs :

Premier : 0 \_ \_ \_ \_ \_ \_ \_ \_ \_

Deuxième : ­\_\_

Troisième \_ \_ \_

premier.add ( new Integer ( 5 ) );

le 5 va a l’indice 0

premier.size() ?? ( *nombre d'éléments du Vecteur* ) = 1

premier.capacity() ?? ( *capacité totale du Vecteur* ) = 10

deuxieme.add ( new Character ( 'u' )) ;

le u va a 0 d’un autre espace

DEPUIS JDK 1.5 🡪 AUTOBOXING :il n’est plus nécessaire d’envelopper explicitement les valeurs en objet enveloppe. C’est fait automatiquement.

deuxieme.add ('t' );

va aux deuxiem vecteur indice 1

Character lettre = new Character ( 'y' );

Il manque de place et va alors doubler la capacite

Deuxième : ­ V Y T

deuxieme.insertElementAt ( lettre, 1 );

deuxieme.indexOf ('t'); ?? = 2

deuxieme.size();?? = 3

deuxieme.capacity(); ?? = 4

troisieme.insertElementAt (22.2 , 7 ); ?? = impossible

deuxieme.remove ( lettre ); ?? = true

deuxieme.indexOf ( new Character ('u') ); ?? = 0

premier.remove ( 4 );?? = pas de chiffre 4 pas d’indice 4 alors false

Une image contenant texte, Visage humain, personne, capture d’écran

Description générée automatiquement