**PROJET NOUVEAUTÉ (PILES, FILES)**

1. PILE

Une **pile** est un contenant pour des objets insérés et retirés selon le principe dernier entré, premier sorti (*last-in-first-out*, ou *LIFO*). Les objets peuvent être insérés à tout moment, mais seulement le dernier (le plus récemment inséré) peut être retiré.

Une pile est un type abstrait de données (TAD) qui supporte deux méthodes principales:

* **Push (0) :** Insère l’objet *o* sur le dessus de la pile.
* **Pop () :** Retire l’objet du dessus de la pile et le retourne; si la pile est vide, alors une erreur survient.

Pile a également des méthodes secondaires qui devraient aussi être définies :

* **Size () :** Retourne le nombre d’objets dans la pile.
* **IsEmpty () :** Retourne un booléen indiquant si la pile est vide.
* **Top () :** Retourne l’objet du dessus de la pile, sans le retirer; si la pile est vide, alors une erreur survient.

**A PROPOS DE JAVA**

Étant donné le TAD pile, nous devons coder cet ADT afin de l’utiliser dans nos programmes.

Il est important de comprendre ces deux concepts de programmation: les **interfaces** et les **exceptions.**

1. L’INTERFACE

Une **interface** est une façon de déclarer ce qu’une classe peut faire. Elle n’indique pas comment le faire.

Pour une **interface**, vous écrivez simplement les **noms de méthodes** et leurs **paramètres**. Ce qui est important dans un paramètre est son **type**. Plus tard, quand vous écrirez une **classe** pour cette interface, vous coderez alors le contenu de ces méthodes.

**NB :** Séparer l’**interface** de la **réalisation** est une technique de programmation très utile.

EXEMPLE D’INTERFACE EN JAVA

Public interface Pile {

// méthodes accessible

Public int size (); // retourne le nombre d’élément dans la pile

Public boolean isEmpty (); // Retourne un booléen indiquant si la pile est vide

Public Object top () // Retourne l’objet du dessus de la pile

Public void push (élément); // ajoute l’élément dans la pile

Public Object pop () // Retire l’objet du dessus de la pile et le retourne

}

1. LES EXCEPTIONS

Les e **exceptions** sont un autre concept de programmation très utile, surtout dans un contexte de gestion d’erreurs. Quand vous détectez une erreur (ou un cas ***exception*nel**), vous lancez (***throw***) une exception.

Aussitôt l’exception lancée, le flux de contrôle sort de la méthode en cours d’exécution.

Exemple :

public void mangePizza () throws MalAuVentreException.

{

...

if (tropMangé)

throw new MalAuVentreException(“Ouch”).

...

}

Alors quand **MalAuVentreException** est lancée, nous sortons de la méthode **mangePizza ()** pour aller là où cette méthode a été appelée.

Supposons que le fragment de code suivant ait appelé la méthode mangePizza () en premier lieu.

Private void simuleRencontre ()

{

...

try

{

unStupideAE.mangePizza ();

}

Catch (MalAuVentreException e)

{

System.out.println(“quelqu’un a mal au ventre”);

}

...

}

Le bloc **try** et le bloc **catch** indiquent que nous sommes à l’écoute des exceptions qui sont spécifiées dans le paramètre de **catch**. Parce que **catch** est à l’écoute de **MalAuVentreException,** le contrôle ira au bloc **catch**, et **System.out.println** sera alors exécuté.

Notez que le bloc **catch** peut contenir n’importe quoi, pas seulement un **System.out.println.** Vous pouvez gérer les erreurs détectées comme bon vous semble, et vous pouvez même les relancer. Notez aussi que si vous lancez une exception dans votre méthode, vous devez ajouter une clause **throws** à la suite du nom de votre méthode.

1. Le TAD File (ou Queue)

Une file se distingue d’une pile par ses routines d’insertion et de retrait qui suivent le principe **premier entré, premier sorti** (*first-in-first-out*, ou ***FIFO***). Des éléments peuvent être insérés à tout moment, mais seulement l’élément qui a été le plus longtemps dans la file peut être retiré.

Les éléments sont enfilés (***enqueued***) par l’arrière (***rea****r*) et défilé (***dequeued***) par l’avant (***front***).

La file supporte deux méthodes fondamentales :

* Enqueue (o) : Insère l’objet *o* à l’arrière de la file
* Dequeue () : Retire l’objet du devant de la file et retourne-le; une erreur survient lorsque la file est vide

File a également des méthodes secondaires qui devraient aussi être définies :

* **Size () :** Retourne le nombre d’objets dans la file
* **IsEmpty () :** Retourne un booléen indiquant si la pile est vide
* **Front () :** Retourne, sans le retirer, l’objet au-devant de la file; si la pile est vide, alors une erreur survient