TP2

Lectures préalables :	Thèmes du TP :

• tutorial
  ○ Getting Started
  ○ Learning the Java Language
	• Une Pile d'entiers
	• Plusieurs classes
	• Tableaux
	• Variables d'instance

- **Visualisez le sujet en ouvrant `index.html` du répertoire qui a été créé à l'ouverture de `tp2.jar` par BlueJ; vous aurez ainsi accès aux applettes et pourrez expérimenter les comportements qui sont attendus.**
- **Soumettez chaque question à l'outil d'évaluation junit3.**

.1) Développez la traditionnelle structure de données **Pile** (premier entré, dernier sorti).

Une Pile est ici une structure de données homogènes, de taille fixe, sur laquelle seul un nombre limité d'opérations peut être effectué :

- constructeur = créer une pile vide d'une certaine taille ou d'une taille par défaut
- procédure "empiler "= mettre une valeur au sommet de la pile
- fonction "dépiler" = retire la valeur du sommet de la pile et la retourne
- fonction booléenne "estVide" = retourne vrai ou faux selon que la pile est vide ou non
- fonction booléenne "estPleine "= retourne vrai ou faux selon que la pile est pleine ou non

Contraintes :

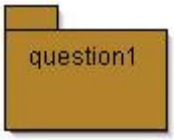
- la classe est de nom "Pile" donc enregistrée dans le fichier Pile.java
- La pile est implantée par un tableau.
- Les éléments sont de type **`int`**.
- Les méthodes dont les signatures figurent ci-dessous doivent être implantées dans la classe Pile, *(pour l'instant sans utiliser de tests ou d'exceptions en cas de débordement, dans un sens ou dans l'autre).*

```
public Pile()
public Pile( int taille )
public void empiler( int i )
public int dépiler()
public boolean estVide()
public boolean estPleine()
```

- Pour pouvoir visionner le contenu de la pile, implantez aussi la méthode ***`public String`***

toString()
Cette méthode retourne le contenu de la pile sous la forme :
"$[i_1, i_2, i_3, i_4]$" où i_1 est l'entier au sommet de la pile.

- Lors de l'appel du constructeur avec **une taille négative ou nulle**, c'est la taille par défaut TAILLE_PAR_DEFAUT qui est retenue.



.2) Développez la classe "UneUtilisation", et répondez aux questions.

Cette utilisation ne cherche volontairement pas à dépiler une pile vide ou à empiler sur une pile pleine (voir question 2.1)

public class **UneUtilisation**{

```
  public static void main(String[] args){
    Pile p1 = new Pile(5);
    Pile p2 = new Pile(10);
    // déjà complété par vos tests...
    System.out.println(" la pile p1 : " + p1.toString());
    // ou bien
    System.out.println(" la pile p1 : " + p1);

    Object o = new Object();
    o = p1;
    System.out.println(" o = " + o); // Quel est l'affichage résultant ? pou


    Pile p3 = new Pile(4); p3.empiler(5); p3.empiler(6);
    Pile p4 = new Pile(6); p4.empiler(8);
    p3 = p4;
    int res = p3.dépiler();
    System.out.println(p4.estVide());  // true ou false ? pourquoi ?
  }
}
```

Exemple de comportement attendu donc à vérifer : la pile résultant des actions p1.empiler(2) suivi de p1.empiler(5) est en String la valeur ***"[5, 2]"***

.1) Ajoutez à la classe Pile la gestion des exceptions "Pile Vide" et "Pile Pleine":

- l'exception "**Pile Vide**" est levée lorsque l'on tente de "dépiler" une pile qui est vide
- l'exception "**Pile Pleine**" est levée lorsque l'on tente d'"empiler" sur une pile déjà pleine.

- Ces **2** exceptions sont implantées par les **2** classes Java :

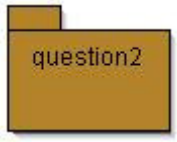
```
public class PileVideException extends Exception {}
  // fichier PileVideException.java
```

```
public class PilePleineException extends Exception {}
  // fichier PilePleineException.java
```

Il n'est pas demandé de compléter ces classes pour stocker des informations sur la cause de l'exception.

Modifiez la classe **ApplettePile** afin de prendre en compte les exceptions susceptibles d'être levées et d'en informer l'utilisateur. Exceptions comme NumberFormatException, PileVideException, PilePleineException

ApplettePile avec exceptions



.2) Proposez une amélioration de la classe ApplettePile2 qui invalide le bouton "empiler" lorsque la pile est pleine et qui invalide le bouton "depiler", comme le suggère cette mise en oeuvre (exemple : appel de **`boutonEmpiler.setEnabled(false);`**).

Vous devez obtenir une applette avec le comportement ci-dessous :

ApplettePile2 de taille 3

Bien sûr cette IHM est perfectible...
mais ne changez rien avant d'avoir soumis votre travail aux outils d'évaluation.
