DIC1
Java && POO
Lab 6

Exercice 1 : Prise en main

Partie A

1) Créer cette classe

```
1  package figures;
2
3  public class Pile {
4      int taille_max = 3;
5      private int sommet;
6      private int[ ] elements;
7
8      public Pile (int [] tab){
9          this.sommet = 0;
10         this.elements = tab;
11     }
12
13     public void empiler (int x) {
14             elements[++sommet] = x;
15     }
16 }
```

2) Puis celle-ci

```
1  package figures;
2
3  public class Test {
4
5      public static void main(String[] args)  {
6          // TODO Auto-generated method stub
7          int [] t = new int [4];
8          Pile p = new Pile(t);
9              int x = 1, y = 2, z = 3, q = 4;
10             p.empiler (x); System.out.println(x);
11             p.empiler(y);  System.out.println(y);
12             p.empiler(z);  System.out.println(z);
13             p.empiler(q);  System.out.println(q);
14
15         }
16
17
18 }
```

Que constatez vous ?

Partie B

1- Créer cette classe

```java
package figures;

public class Pile {
    int taille_max = 3;
    private int sommet;
    private int[ ] elements;

    public Pile (int [] tab){
        this.sommet = 0;
        this.elements = tab;
    }

    public void empiler (int x) throws ErreurPilePleine {
    if(sommet == taille_max)
         throw new ErreurPilePleine();
    else    elements[++sommet] = x;
    }
}
```

2- Ensuite celle là

```java
package figures;

public class ErreurPilePleine extends java.lang.Exception {
    ErreurPilePleine(){
        System.out.println("Pile Pleine");
    }
}
```

3) Enfin celle là

```java
package figures;

public class Test {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int [] t = new int [4];
        Pile p = new Pile(t);
        try {
            int x = 1, y = 2, z = 3, q = 4;
            p.empiler (x); System.out.println(x);
            p.empiler(y);  System.out.println(y);
            p.empiler(z);  System.out.println(z);
            p.empiler(q);  System.out.println(q);
        } catch (ErreurPilePleine e) {

        }




    }

}
```

Exercice2 : Application …

Pour implémenter cet exercice il faut créer un package nommé *« ensemble.gestion »* dans laquelle on crée nos classes.

1. Les classes exceptions

Ecrire la classe *OrdreException* qui est lancée quand les éléments de l'ensemble à créer ne sont pas ordonnés. Cette classe peut être écrite comme suit.

```
public class OrdreException extends Exception {
        public OrdreException(){
                super("l'ensemble doit être ordonné : un élément dérange l'ordre");
        }
}
```

Ecrire la classe *TailleMaxException* qui est lancée quand la tailleMax de l'ensemble à créer dépasse 100. Cette classe peut être écrite comme suit.

```
public class TailleMaxException extends Exception {
        public TailleMaxException(){
                super("la taille max d'un ensemble : inférieure à 100");
        }
}
```

Ecrire la classe *TailleEnsUnionException* qui est lancée quand la tailleMax de l'ensemble à créer union veut dépasser 100. Cette classe peut être écrite comme suit.

```
public class TailleEnsUnionException extends Exception {
        public TailleEnsUnionException(){
                super("la tailleMax d'un ensemble : inférieure à 100");
        }
}
```

NB : D'autres exceptions peuvent être écrites.
On peut aussi utiliser les exceptions de l'API Java comme *NumberFormatException* pour des erreurs de formats, etc.

2. La classe EntierOrd

Pour l'implémentation de la classe EntierOrd, nous fixons la taille maximum de l'ensemble à 100. Ensuite nous utilisons les exceptions pour lancer des messages dans les cas où :

- On a atteint la taille max égale à 100.
- L'union de deux ensembles veut dépasser 100 éléments.
- Le tableau servant à créer l'ensemble n'est pas ordonné.
- On a une erreur de format : le tableau contient des éléments non numériques.

a. Les attributs

```
private int taille;
private int [] ensemble = new int [100];
private int indice;
```

b. Constructeur

Ecrire le constructeur EntierOrd qui construit un ensemble d'entiers ordonnés. Ce constructeur doit gérer une exception TailleMaxException écrite précédemment.

Voici sa signature :

```
public EntierOrd(int taille, int []tab) throws TailleMaxException{...}
```

c. Les autres méthodes

Implémenter les méthodes insertion, suppression, appartient, union et intersection en lançant les exceptions *TailleMaxException, TailleEnsUnionException* écrites précédemment. Les signatures de ces méthodes sont les suivantes :

```
public void insertion (int x) throws TailleMaxException{...}
public void suppression(int x){ ...}
public boolean appartient(int x){ ...}
public EntierOrd unionEnsemble(EntierOrd e) throws TailleMaxException,
                                    TailleEnsUnionException{...}
public EntierOrd intersectionEnsemble(EntierOrd e) throws TailleMaxException
                                                    {...}
public String toString(){...}
```

3. La classe CreerEntierOrd

Cette classe utilise les arguments de la ligne de commande pour instancier deux ensembles d'entiers ordonnés et utiliser les methodes de la classe EntierOrd. C'est dans cette classe que les exceptions sont capturées.

```
public class CreerEntierOrd {

	public static void main(String[] args) throws OrdreException {
		try{
		int taille1=Integer.parseInt(args[0]);
		int taille2=Integer.parseInt(args[1]);
		int [] tab1 = new int [100];
		int [] tab2 = new int [100];
		for(int i=0; i<taille1; i++){
			if((i>0) && (Integer.parseInt(args[i+2]) <

Integer.parseInt(args[i+1])))
				throw new OrdreException();
			tab1[i]=Integer.parseInt(args[i+2]);
		}
		for(int j=0; j<taille2; j++){
			if((j>0)&&(Integer.parseInt(args[j+2+taille1]) <

Integer.parseInt(args[j+1+taille1])))
				throw new OrdreException();
			tab2[j]=Integer.parseInt(args[j+2+taille1]);
		}
		EntierOrd e1= new EntierOrd(taille1,tab1);
		EntierOrd e2= new EntierOrd(taille2,tab2);
		System.out.println(e1);
		System.out.println(e2);
		e1.insertion(6);
		System.out.println(e1);
		e1.suppression(6);
		System.out.println(e1);
		EntierOrd e3=e1.unionEnsemble(e2);
		System.out.println(e3);
		EntierOrd e4=e1.intersectionEnsemble(e2);
		System.out.println(e4);
		}
		catch(TailleMaxException me){
			System.out.println(me.getMessage());
		}
		catch(OrdreException Oe){
			System.out.println(Oe.getMessage());
		}
		catch(NumberFormatException Ne){
			System.out.println("Erreur de format!!!");
		}
		catch(TailleEnsUnionException me){
			System.out.println(me.getMessage());
		}
	}
}
```