Tutoriel Exceptions en Java

Programme simulant une exception qui n'est pas récupérée

Une exception qui n'est pas récupéré et traité va produire le crash du programme

```
//*** Programme simulant une exception qui n'est pas récupérée
public class Sample1 {
     public static void main(String[] args) {
          int x;
          System.out.println("***  avant l erreur   ***");
          x=1/0;
          System.out.println("Résultat = "+x);
          System.out.println("***  apres l erreur   ***");
     }
}
```

Programme simulant une exception récupérée et traitée

La récupération (le traitement) de l'exception permet au programme de continuer son exécution.

```
//*** Programme simulant une exception récupérée et traitée
public class Sample1 {
     public static void main(String[] args) {
          int x;
          System.out.println("***  avant l erreur   ***");
          try {
               x=1/0;
               System.out.println("Résultat = "+x);
          } catch (ArithmeticException e) {
               System.out.println("oups!!erreur de division");
               System.out.println(e.getMessage());
               e.printStackTrace();
          }

          System.out.println("***  apres l erreur   ***");
     }
}
```

Propager l'exception

Si l'exception n'est pas récupérée et traitée à l'endroit ou elle se produit celle-ci va se propager à la méthode appelante et ainsi de suite jusqu'à arriver à la JVM et produire le crash du programme.

```
//****** Propager l'exception à la méthode appelante qui à son
tour va traiter l'exception ou la propager.
class Sample2 {
     public void division(){
          int x;
          System.out.println("***  avant l erreur   ***");
          x=1/0;
          System.out.println("***  apres l erreur   ***");
     }}
public class mainSample2{
     public static void main(String[] args) {
          Sample2 s=new Sample2();
          System.out.println("Début du programme");
          try{
          s.division();
          }catch(ArithmeticException e){
               System.out.println(e.getMessage());
          //   e.printStackTrace();

          }
          System.out.println("Fin du programme");

     }
}
```

Exceptions multiples

Il est possible d'intercepter plusieurs types d'exceptions différentes et de les traiter en faisant suivre le bloc try par plusieurs catch. Une seule clause catch sera au final exécutée.

- !!! l'ordre des blocs catch doit se faire **de l'erreur la plus spécifique à la plus générale**

```
/******** Exceptions multiples ******************//
public class Sample3 {
     public static void main(String[] args) {

/******** Exceptions multiples ******************/
     public static void main(String[] args) {
          Scanner sc=new Scanner(System.in);
          int tab[]=new int [3];

          System.out.println("*** avant remplissage du tableau
***");
          try {
          for (int i=0;i<=3;i++){
               System.out.println("donner a=");
               int a=sc.nextInt();
               System.out.println("donner b=");
               int b=sc.nextInt();
          tab[i]=a/b;
          }
          }
          catch (ArithmeticException e1){
          System.out.println(e1.getMessage());
          }
          catch (ArrayIndexOutOfBoundsException e2){
          System.out.println(e2.getMessage());
          System.out.println("Array exception déboremnet!!!");
          }
          catch(RuntimeException e0){
          System.out.println("Runtime exception="+e0.getMessage());
          }
          System.out.println("*** après remplissage du tableau
***");


     }}
```

Le mot clé finally

Le bloc finally est un bloc qui sera exécuté qu'il y est eu exception ou pas (dans tous les cas)

```
// *** le mot clé finally ***//
public class Sample4 {

	public static void main(String[] args) {
		int x;

		try {
			x=1/0;
			System.out.println("Résultat = "+x);
		} catch (ArithmeticException e) {
			System.out.println("oups!!erreur de division");
			System.out.println(e.getMessage());

		}
		finally {
		System.out.println("affichage du bloc finally !!!");
		}
	}
}
```

Exception personnalisée

```
public class AegalBException extends Exception {

	public AegalBException() {
		super();
	}

	public AegalBException(String message) {
		super(message);
	}
	public String toString(){
		return "erreur de type A égal B";
	}
}
```

```
public class EssaiException {
     public static void compareAetB(int a, int b)throws
AegalBException{
          if (a==b)throw new AegalBException("ouups a et b sont
égaux");
          }
     public static void main(String[] args) {

          try{
               EssaiException.compareAetB(5, 5);
          }catch(AegalBException e){
               System.out.println(e);
               System.out.println(e.getMessage());
          }
     }
     }
```