IUT2 Université Grenoble Alpes
Département INFO

R1.01
Initiation au développement

Compléments pour la partie B du TP11

- Déclaration et utilisation d'un tableau
- Exception définie par le programmeur

Déclaration d'un tableau

- Déclaration : `typeElem [] nomTableau;`

  Exemples

  ✓ déclaration d'un tableau d'entiers : `int[] tabInt;`

  ✓ déclaration d'un tableau de chaînes : `String[] tabStr;`

  ✓ déclaration d'un tableau de Polar : `Polar[] tabPolar;`

- Une variable de type tableau doit être dimensionnée au nombre d'éléments désirés - sa dimension n'est pas modifiable

  La dimension peut-être…

  ✓ explicite :
  ```
  tabInt = new int[5]; // tabInt pourra contenir 5 entiers
  ```

  ✓ implicite (suite à initialisation par lot) :
  ```
  tabInt = {1,3,4,5,9} // tabInt initialisé, dimension = 5
  ```

- Le premier indice d'un tableau est `0`
- La dimension d'un tableau `tab` est égale à `tab.length;`

Utilisation d'un tableau

Soit tab un tableau d'entiers

- Accès à un élément d'indice k : tab[k]

  Exemples

  ✓ initialisation / mofication de l'élément d'indice k : `tab[k] = 3;`

  ✓ test de la valeur de l'élément d'indice k : `if (tab[k] == ...)`

  ✓ affichage de tous les éléments :

```
for (int k = 0; k < tab.length; k++) {
  System.out.println("indice " + k + " : " + tab[k]);
}
```

Exception définie par le programmeur

A - DÉFINITION

*cf. cours 9 page 58

Exemple : déclaration d'une exception `ExceptionMonEx`

Dans le projet concerné…

1) Définir une classe publique `ExceptionMonEx` qui étend la classe `Exception`
2) Dans cette classe, définir 2 constructeurs de `ExceptionMonEx`
   - ✓ un constructeur par défaut
   - ✓ un constructeur avec un paramètre de type `String`, pour définir un message d'erreur expliquant la levée de l'exception

```
public class ExceptionMonEx extends Exception {
    // constructeur sans paramètre
    public ExceptionMonEx() {
        super();
    }
    // constructeur avec un paramètre String représentant
    // un message expliquant pourquoi l'exception a été levée
    public ExceptionMonEx(String s) {
        super(s);
    }
}
```

Exception définie par le programmeur

B - UTILISATION DANS LE CODE (1)

Une exception définie par le programmeur s'utilise de la même façon que toute exception du langage Java

- Pour lever l'`ExceptionMonEx` :
  - ✓ sans message : `throw new ExceptionMonEx();`
  - ✓ avec un message approprié : `throw new ExceptionMonEx("mess");`
- En-tête d'une procédure où `ExceptionMonEx` est levée :

```
public* [static] void nomProc([paramètres])
                        throws ExceptionMonEx {
```

- En-tête d'une fonction où `ExceptionMonEx` est levée :

```
public* [static] type_résultat nomFonc([paramètres])
                                  throws ExceptionMonEx {
```

DANS LE CODE DE LA PROCÉDURE OU DE LA FONCTION, AU MOINS UNE INSTRUCTION DOIT LEVER L'EXCEPTION (AVEC OU SANS MESSAGE)

*ou `private`

Exception définie par le programmeur

B - UTILISATION DANS LE CODE (2)

Exemple : utilisation dans le code de l'exception `ExceptionMonEx`

- Une instruction qui appelle une fonction ou une procédure qui lève l'`ExceptionMonEx` doit traiter l'exception levée

- *Exemple*

```java
import java.util.Scanner;
public class TestMonEx {
    public static int moitieDe(int x) throws ExceptionMonEx {
        if (x % 2 == 0) { return x/2; }
        else {
            throw new ExceptionMonEx("Problème !!! "+ x +" impair...");
        }
    }
    public static void main(String[] args) {
        Scanner lecteur = new Scanner(System.in);
        System.out.print("Entrer un entier pair : ");
        int x = lecteur.nextInt(); lecteur.nextLine();
        try {
            System.out.println("Moitié de " + x + " = " + moitieDe(x));
        } catch (ExceptionMonEx e) {
            System.out.println(e.getMessage());
        }
    }
}
```

TRACES D'EXÉCUTION

Entrer un entier pair : 16
Moitié de 16 = 8

Entrer un entier pair : 15
Problème !!! 15 impair...

Lever plusieurs exceptions…

- Une procédure ou une fonction peut lever plusieurs exceptions différentes (définies par le programmeur, ou non)
- Dans ce cas, il faut séparer par une virgule l'identifiant de chaque exception levée
- *Exemples - E1, E2, E3, E4 étant des exceptions définies*
  - ✓ `public* [static] void nomProc ([paramètres]) throws E1, E2 {`
  - ✓ `public* [static] type_résultat nomFonc([paramètres) throws E1, E3, E4 {`

CHAQUE EXCEPTION FIGURANT DANS L'EN-TÊTE D'UNE PROCÉDURE (OU FONCTION) DOIT ÊTRE LEVÉE DANS LE CORPS DE CETTE PROCÉDURE (OU FONCTION)

*ou `private`