

# Module 6:

## Arrays (Tableaux)

# Vue d'ensemble

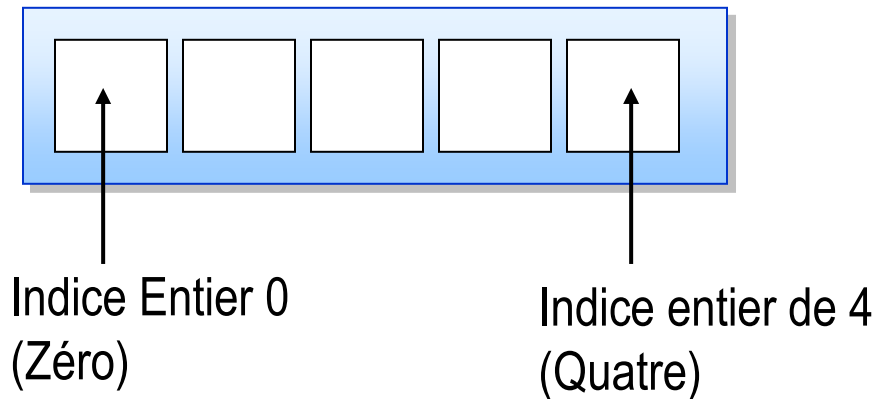
- Vue d'ensemble des tableaux
- Création de tableaux
- Utilisation des tableaux

# Vue d'ensemble des tableaux

- Qu'est-ce qu'un tableau?
- Notation de tableau en C#
- Tableau à plusieurs dimensions
- Accès aux éléments du tableau
- Vérification des bornes de tableaux
- Comparaison entre les tableaux et les collections

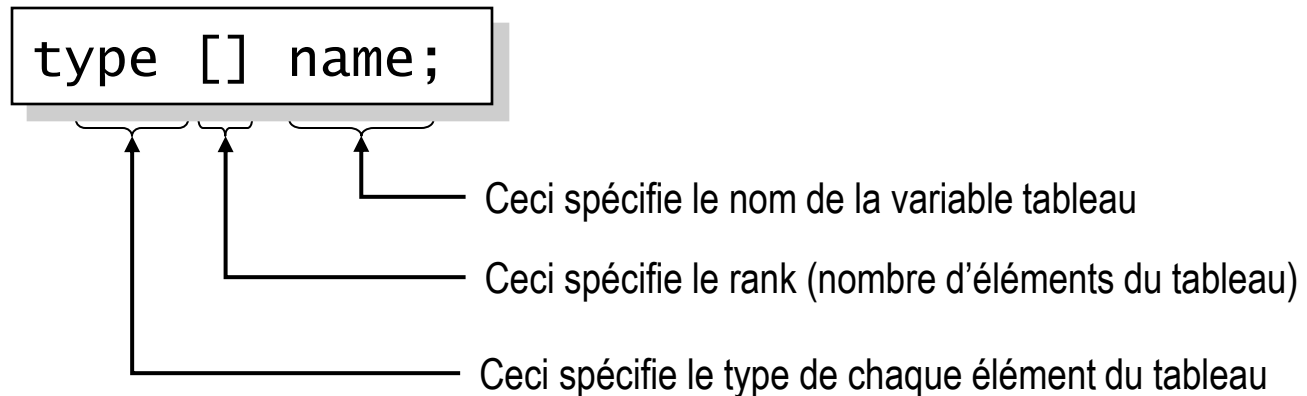
# Qu'est-ce qu'un tableau?

- Un tableau est une séquence d'éléments
  - Tous les éléments d'un tableau sont de même type
  - Les éléments individuels sont accessibles à l'aide d'indices entiers



# Notation de tableau en C #

- Vous déclarez une variable tableau en précisant:
  - Le type de chaque élément du tableau
  - Le rank (nombre d'éléments du tableau)
  - Le nom de la variable

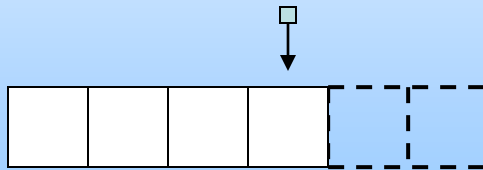


# Tableau à plusieurs dimensions

- Le rank permet aussi de définir les dimensions d'un tableau

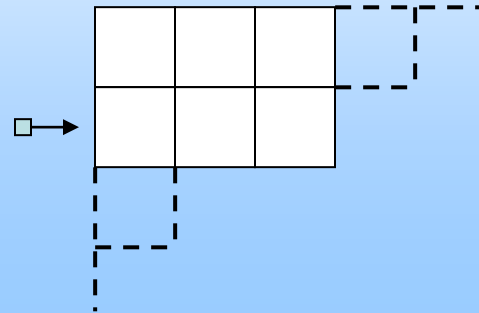
```
long[ ] row;
```

Rank 1: Une dimension  
Indices simples associés avec  
chaque élément **long**



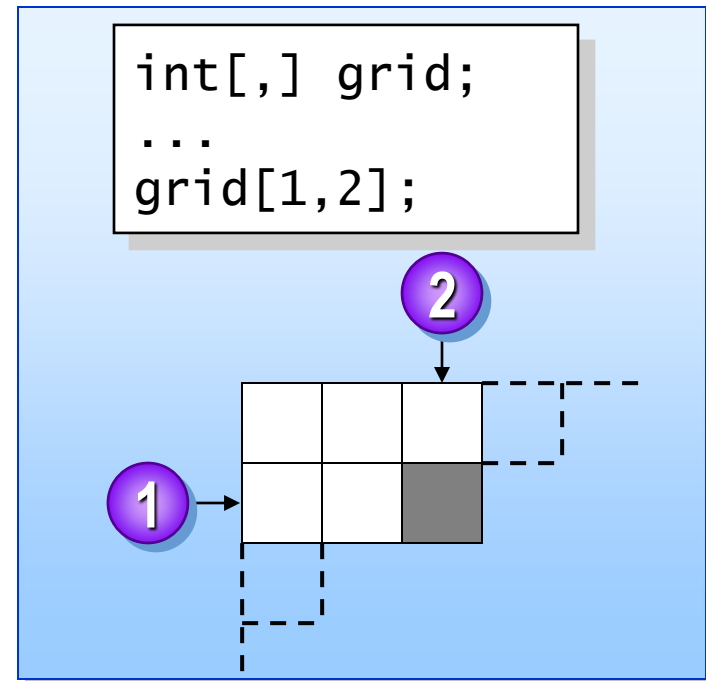
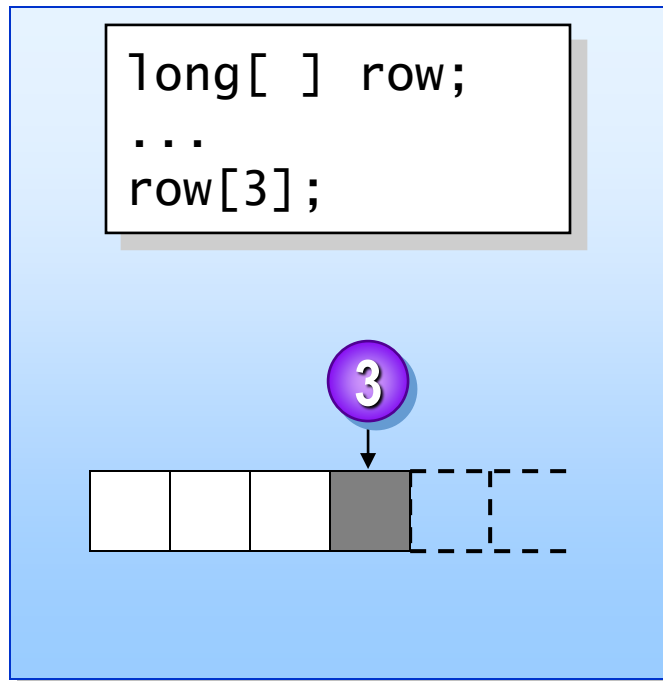
```
int[,] grid;
```

Rank 2: Deux dimensions  
Deux indices associés  
chaque élément **int**



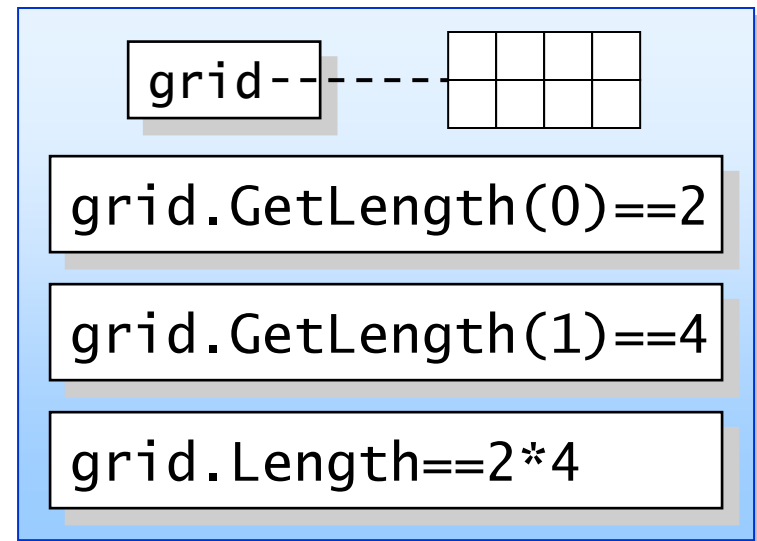
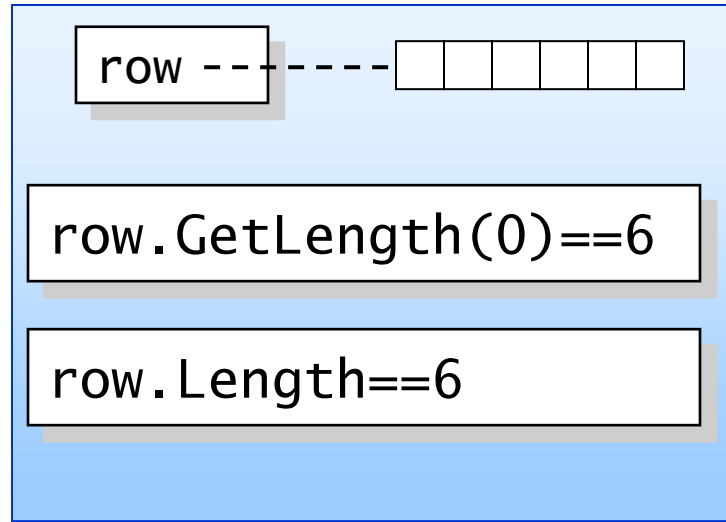
# Accès éléments du tableau

- Fournir un indice entier pour obtenir chaque element
  - Les index sont de base zéro



# Vérification des bornes de tableaux

- Toutes les tentatives d'accès sont vérifiées pour ne pas dépasser les limites
  - Un mauvais index lance une `IndexOutOfRangeException`
  - Utilisez la propriété **Length** et la méthode **GetLength**





# La comparaison entre les tableaux et les collections

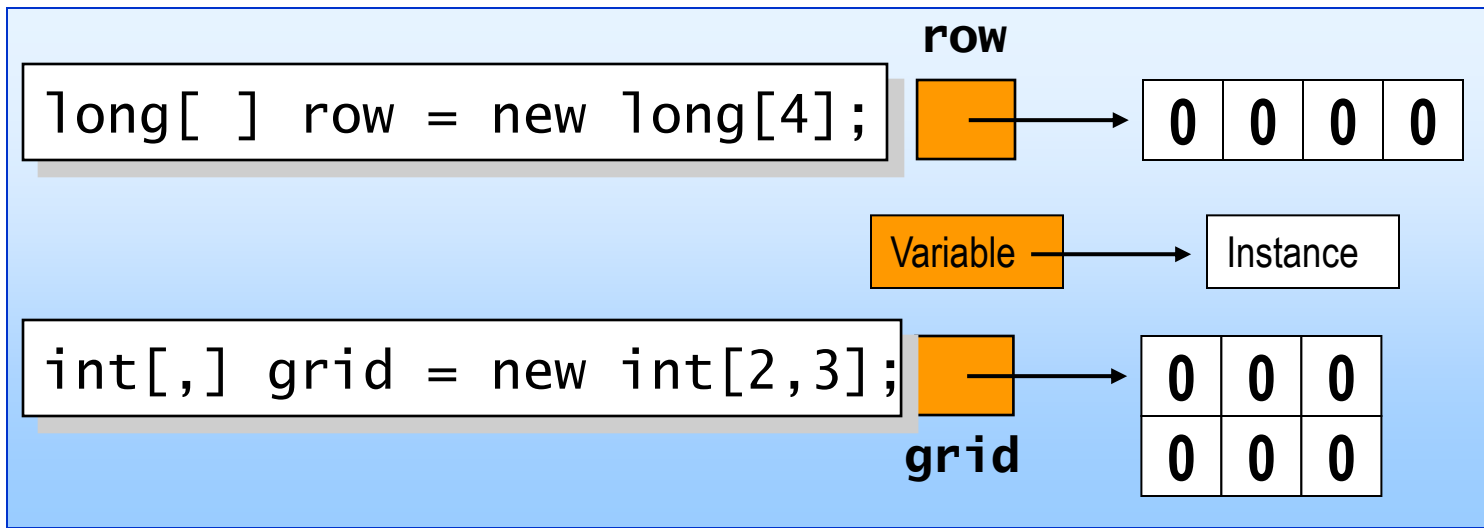
- Un tableau ne peut pas se redimensionner même s'il est plein
  - Une classe collection, telle que ArrayList, peut se redimensionner
- Un tableau est destiné à stocker des éléments d'un type
  - Une collection est conçue pour stocker des éléments de types différents
- Les éléments d'un tableau ne peuvent pas avoir d'accès en lecture seule
  - Une collection peut avoir accès en lecture seule
- En général, les tableaux sont plus rapides mais moins flexible
  - Les collections sont un peu plus lentes, mais plus souple

# Création de tableaux

- Création d'instances Array
- Initialisation éléments du tableau
- Initialisation des éléments de tableau multidimensionnel
- Création d'un tableau Taille informatisée
- Variables copie Array

# Création d'instances Array

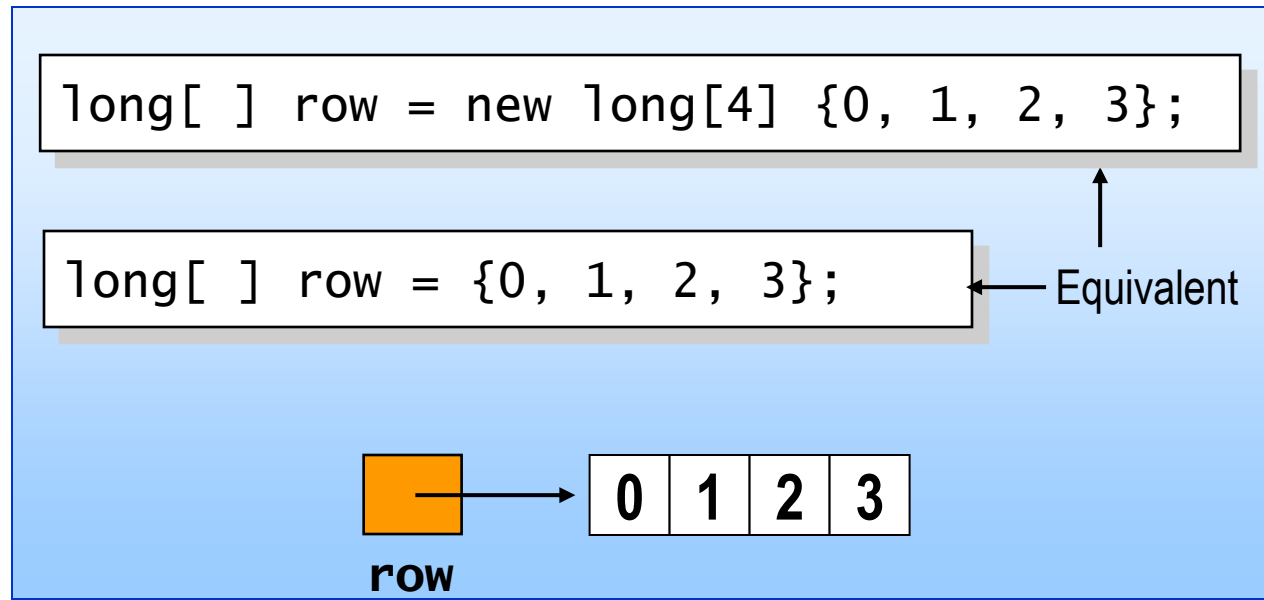
- Déclarer une variable tableau ne crée pas un tableau!
  - Vous devez utiliser **nouveau** de créer explicitement l'instance de tableau
  - Les éléments du tableau ont une valeur par défaut qui est la valeur par défaut du type de chaque élément du tableau



- La valeur par défaut d'un type peut être obtenue comme ça :  
`int defaultValueOfInt = default(int);`

# Initialisation éléments du tableau

- Les éléments d'un tableau peuvent être explicitement initialisés
  - Vous pouvez utiliser un raccourci commode




# Initialisation des éléments de tableau multidimensionnel

- Vous pouvez également initialiser les éléments de tableau multidimensionnel
  - Tous les éléments doivent être précisés

```
int[,] grid = {  
    {5, 4, 3},  
    {2, 1, 0}  
};
```

← Implicitly a new int[2,3] array

✓  **grid** → 

5	4	3
2	1	0

✗

```
int[,] grid = {  
    {5, 4, 3},  
    {2, 1 }  
};
```

# Création d'un tableau de taille prédéterminée

- Toute expression entière fonctionnera

```
long[ ] row = new long[4];
```

- La taille du tableau n'a pas besoin d'être une constante

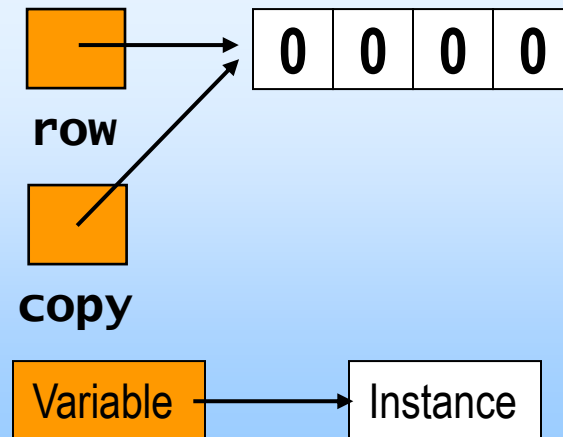
```
string s = Console.ReadLine();  
int size = int.Parse(s);  
long[ ] row = new long[size];
```

- L'accès aux éléments est aussi rapide dans tous les cas

# Variables copie Array

- Copier une variable tableau copie la variable tableau ne
  - Il ne copie pas l'instance de tableau mais la référence vers cette instance
  - Deux variables de tableau peuvent se référer à la même instance de tableau:

```
long[ ] row = new long[4];  
long[ ] copy = row;  
...  
row[0]++;  
long value = copy[0];  
Console.WriteLine(value);
```



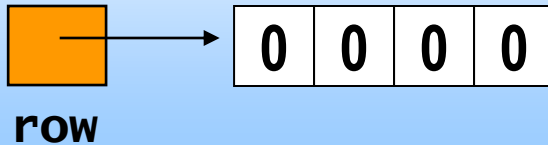
# Utilisation des tableaux

- Propriétés des tableaux
- Méthodes des tableaux
- Méthodes retournant un tableau
- Passage de tableaux en tant que paramètres
- Command-Line Arguments
- Démonstration: Arguments de Main
- Utilisation des tableaux avec foreach
- Quiz: Découvrir les Bugs



# Propriétés des tableaux

```
long[ ] row = new long[4];
```



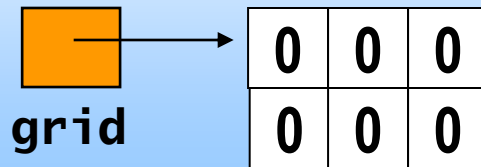
row.Rank

1

row.Length

4

```
int[,] grid = new int[2,3];
```



grid.Rank

2

grid.Length

6

# Méthodes des tableaux

- Méthodes couramment utilisées
  - **Sort** - Trie les éléments d'un tableau dont le rank est 1
  - **Clear** - Définit une série d'éléments à zéro ou **nul** (*valeur par défaut du type*)
  - **Clone** – Duplique le tableau
  - **GetLength** - Retourne la longueur d'une dimension donnée
  - **IndexOf** - Retourne l'index de la première occurrence d'une valeur passé comme paramètre

# Retour tableaux de méthodes

- Vous pouvez déclarer des méthodes qui renvoient des tableaux

```
class Example {  
    static void Main( ) {  
        int[ ] array = CreateArray(42);  
        ...  
    }  
    static int[ ] CreateArray(int size) {  
        int[ ] created = new int[size];  
        return created;  
    }  
}
```

# Passage de tableaux en tant que paramètres

- Un paramètre tableau n'est pas une copie mais une référence du tableau :

```
class Example2 {  
    static void Main( ) {  
        int[ ] arg = {10, 9, 8, 7};  
        Method(arg);  
        System.Console.WriteLine(arg[0]);  
    }  
    static void Method(int[ ] parameter) {  
        parameter[0]++;  
    }  
}
```

**Cette méthode va modifier  
le tableau original  
créée dans Main**

# Arguments de ligne de commande

- Le runtime passe les arguments passé à la ligne de commande à Main
  - **Main** peut prendre un tableau de string comme paramètre
  - Le nom du programme n'est pas un membre du tableau

```
class Example3 {  
    static void Main(string[ ] args) {  
        for (int i = 0; i < args.Length; i++) {  
            System.Console.WriteLine(args[i]);  
        }  
    }  
}
```

- Démo !

# Utilisation des tableaux avec foreach

- L'instruction foreach fait abstraction de nombreux détails de la manipulation de tableau

```
class Example4 {  
    static void Main(string[ ] args) {  
        foreach (string arg in args) {  
            System.Console.WriteLine(arg);  
        }  
    }  
}
```

# Quiz: Découvrir les Bugs

1

```
int [] tableau;  
tableau = {0, 2, 4, 6};
```

2

```
int [] tableau;  
System.Console.WriteLine (tableau[0]);
```

3

```
int [] tableau = new int [3];  
System.Console.WriteLine (tableau [3]);
```

4

```
int [] array = new int [];
```

5

```
int [] array = new int [3] {0, 1, 2, 3};
```

## Atelier 6.1: Création et utilisation de tableaux

