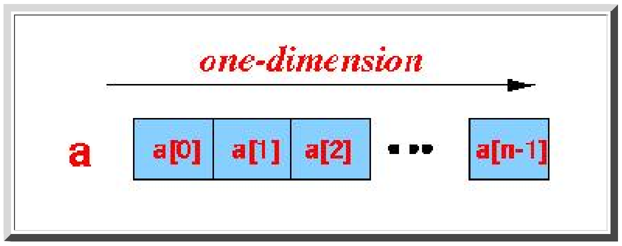
2-dimensional arrays

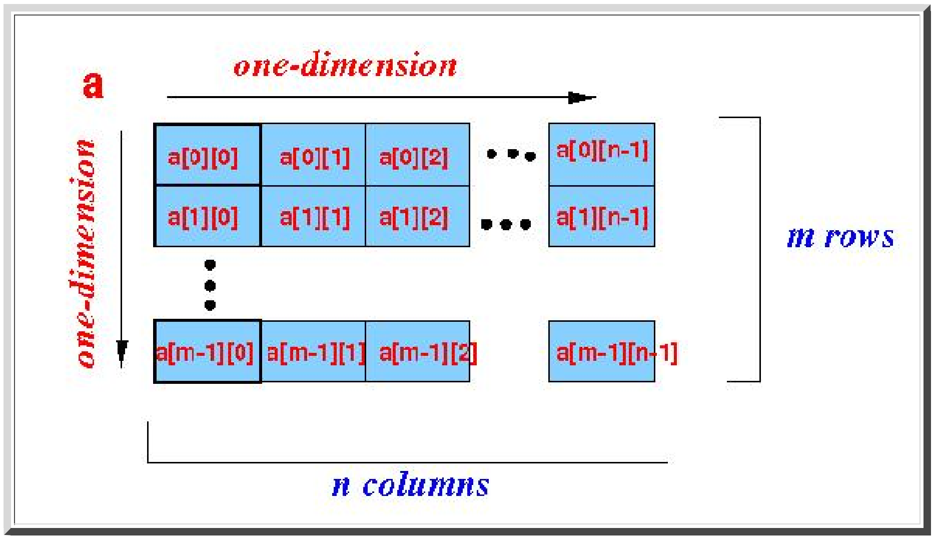
- ***Two*-dimensional arrays**
  - We have seen a ***one*-dimensional array**:

    The **array elements** are ***selected*** using a ***one* index**

  - A ***two*-dimensional array** is an **array** where its **elements are selected (identified)** using ***two* indices**.

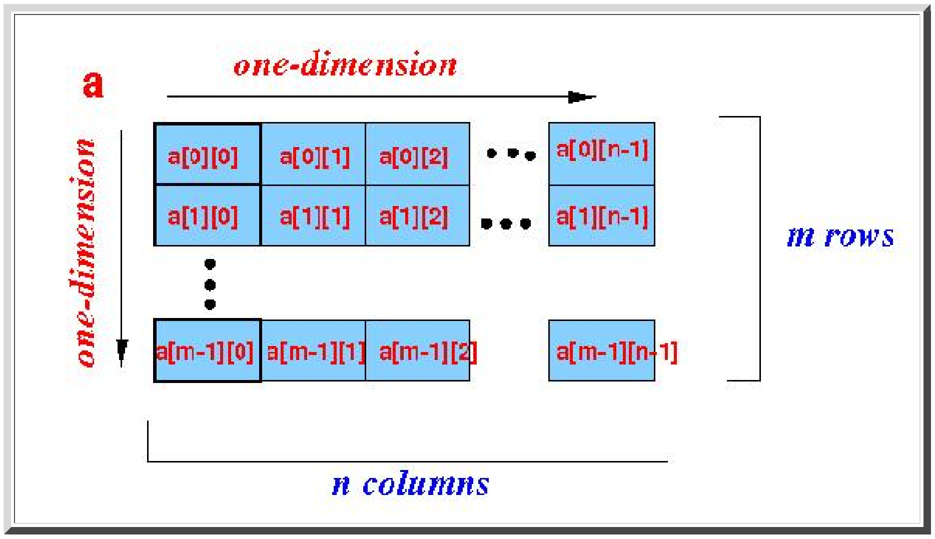
    Example:

- **A note on *Java's* 2-dimensional arrays**
  - What I am going to teach here is a ***simpler* version** of **Java's 2-dimensional arrays**
  - **Many programming languages** (such as **C and C++**) that provide ***static* array**, allow the programmer to define ***rectangular* 2-dimensional arrays**

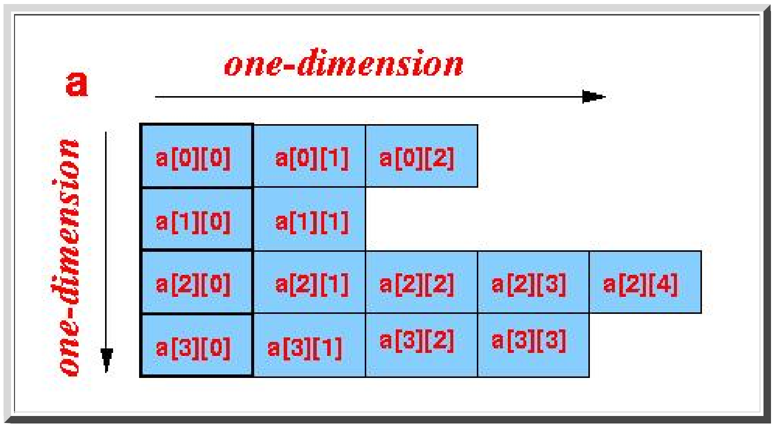
    In a ***rectangular* 2-dimensional arrays**, **each *row* and *column*** in the **array** has the ***same* number of array elements**.

    Example:



- A **2-dimensional array** in **Java** can have ***different* numbers of elements** in **different rows**

  Example:



  (We will **only discuss** and use ***rectanglar* 2-dimensional arrays** in this course).

- **Defining a (retangular) 2-dimensional array**

  - **Defining a *2-dimensional* array: (just like a *one-dimensional* array - is a 2 step process)**

    - **Step 1:**

      - **Define** an **array object *reference* variable** refering to a **2-dimensional array**

      Example:

```
double[][]  a;   // double[][] means:
                 //    a reference (location) of a 2 dim. array
```

    - **Step 2:** (same as a ***one*-dimensional array** !!!!)

      - **Create the (2-dimensional) array** and **store the location** of the **first element of the array** in the (array) object reference variable

Example:

```
a = new double[3][4];  // new double[3][4] creates a 3x4 array
                       // A 3x4 array = an array with 3 rows and 4 columns
```

- **Defining** an ***initialized*** **2-dim. array:**

  - Is very **similar** to the **syntax** used to define an ***initialized*** **one-dimensional array**
  - The **initial value** are **separated** by **nested { ... }**

  Example:

```
public class TwoDimArray1
{
   public static void main(String[] args)
   {
      double[][] a = {
                        { 1.0, 2.0, 3.0, 4.0 },   // *** Defining an
                        { 2.0, 5.0, 1.0, 7.0 },   // *** initialized
                        { 4.0, 1.0, 2.0, 8.0 }    // *** 2-dim. array
                     };

      ******************************************
      Ignore the rest of the program for now...
      ******************************************

      int i, j;          // Array indices

      /* ----------------- Print 2-dim array a -------------------- */

      // Print elements in row i
      for ( i = 0 ; i < a.length ; i++ )
      {

         // Print column j in row i
         for ( j = 0 ; j < a[i].length ; j++ )
         {
            System.out.print( a[i][j] + "   " );
         }

         System.out.println();
      }

   }
}
```

- **Using elements of a 2-dimensional array**

  - **In *general***, using an ***n*-dimensional array** proceeds as follows:

    - **Each array element** of a ***n*-dimensional array** is a **ordinary variable**

      (This is **true** for a **one-dimensional array** and it is ***equally true*** for a **two-dimensional array**).

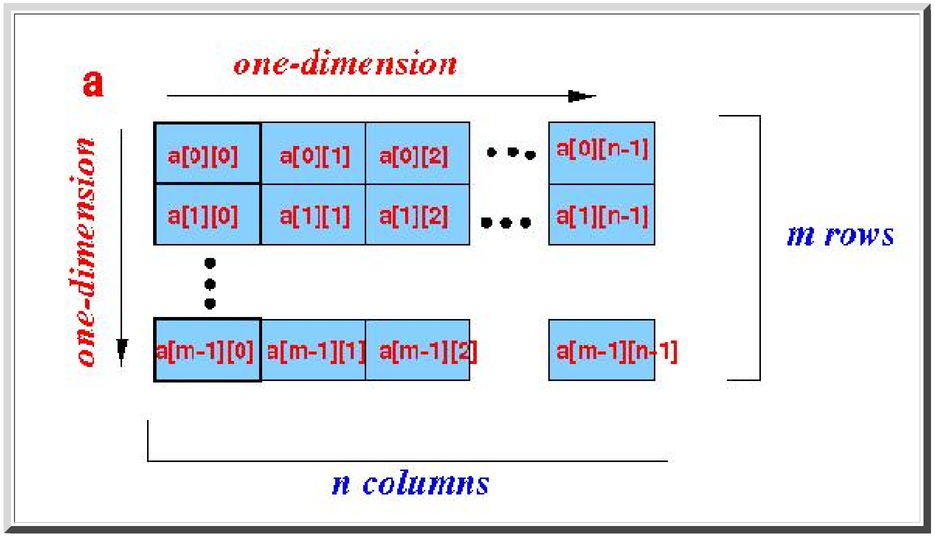
    - An **array element** of a ***n*-dimensional array** consists of:

- the **name of the array reference variable** and
- ***n* array indices**

- **Therefore**, an **element** of a **two-dimensional array** is specified as:

```
   ArrayRefVariable [ index1 ] [ index2 ]
```

Example: (***assuming*** that `a` is a **reference variable** to a **2-dimensional array**)



- **Traversing (visiting) all elements in a 2-dimensional array: *rectanglar* 2-dim. arrays**
  - If the **2-dim. array** is **retangular**, and we know that:

```
   m = # rows
   n = # columns
```

Then we can use the following **nest *for*-loop** to visit all elements:

```
   for ( int i = 0; i < m; i++ )
   {
       // variable i runs through all rows in the array

       for ( int j = 0; j < n; j++ )
       {
           Use array element a[i][j]        // Visit a[i][j]
       }
   }
```

  - If rows have **different number** of elements, we need to find out the **number of elements** in each **row *individually***.

This is discussed next.

- **Traversing (visiting) *all* elements in a *2-dimensional array*: *non-rectanglar* 2-dim. arrays**
  - **Previously discussed:** traversing ***all* elements** in a **1-dimensional array**

    (Code segment)

    ```
    double[] a;

    a = new double[ anyVlaue ];

    for ( i = 0; i < a.length; i++ )
    {
       visit (= use) array element a[i]
    }
    ```

  - Traversing ***all* elements** in a **2-dimensional array** in **Java** is ***usually*** done in the **following manner**:

    - Visit **each row** in **seccession**
    - In **each row**, visit ***all* elements** in that row.

    Pseudo code:

    ```
    for ( each row index i = 0, 1, 2, .... a.length )
    {
        visit (= use) all array elements in row i
    }
    ```

    Note:

    - We need to use an **index** to go through **all rows *and* all columns**
    - The **variable** `a.length` contains the **number of *rows***

  - **$64,000 question:**

    - **Where** do we find the information on the **number of *columns*** ???

  - **Information** on the **number of *columns*** of a **2-dimensional array**:

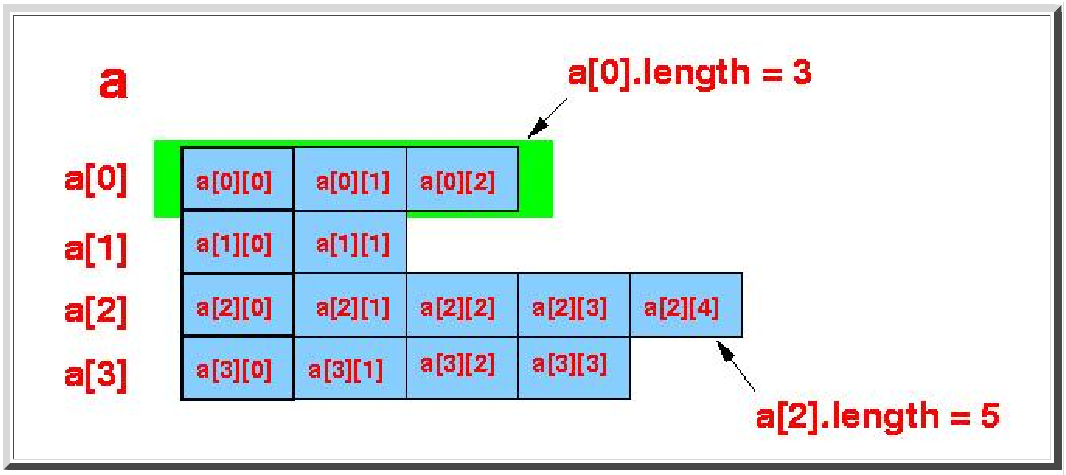
    - In **Java**:

      - *If*: `a` is a **2-dimensional array**

        then: `a[i]` is a **1-dimensional array**

    - In other words: a **2-dimensional array** in **Java** is made up with **many rows of 1-dimensional arrays**

      Illustrated:



- **Therefore**:
  - The **variable** `a[i].length` contains the **number of columns** in **row i**.

    (Because if `x` is an **array**, then `x.length` is the **number of elements** in the **array**)

We can **now refine** the **psuedo code** to visit ***all* elements** in a **2-dimensional array**:

```
for ( each row index i = 0, 1, 2, .... a.length )
{
    visit (= use) all array elements in row i
}
```

Refined pseudo code:

```
for ( each row index i = 0, 1, 2, .... a.length )
{
    for ( each column index j = 0, 1, 2, ...., a[i].length )
    {
        visit (= use) a[i][j]
    }
}
```

- **Java program** that **prints** the **elements in a 2-dim. array**:

```
public class TwoDimArray1
{
   public static void main(String[] args)
   {
      double[][] a = {
                       { 1.0, 2.0, 3.0, 4.0 },
                       { 2.0, 5.0, 1.0, 7.0 },
                       { 4.0, 1.0, 2.0, 8.0 }
                     };

      int i, j;          // Array indices

      /* ----------------- Print 2-dim array a -------------------- */

      // Print elemenst in row i
      for ( i = 0 ; i < a.length ; i++ )
      {
           // Print element j in row i
           for ( j = 0 ; j < a[i].length ; j++ )
           {
              System.out.print( a[i][j] + "    " );
           }

           System.out.println();
        }

     }
  }
```

- **Example Program:** (Demo above code)
  - Prog file: click here

How to run the program:

- **Right click** on link and **save** in a scratch directory
- To compile: `javac TwoDimArray1.java`
- To run: `java TwoDimArray1`
