COMP 110/L Lecture 18

Mahdi Ebrahimi

Some slides adapted from Dr. Kyle Dewey

Outline

- JUnit `fail()`
- More two-dimensional array examples

JUnit `fail()`

`fail()`

Triggers immediate test failure

`fail()`

Triggers immediate test failure

```
import static org.junit.Assert.fail;
```

fail()

Triggers immediate test failure

```
import static org.junit.Assert.fail;


    @Test
    public void testSomething() {
      if (someFailureCondition) {
        fail();
      }
    }
```

Example

- `FailExample.java`
- `FailExampleTest.java`

`fail()` Utility

- Some test failures cannot be easily phrased as one value equals another value
- Occasionally more convenient
- We can define our own `assertEquals()` and `assertArrayEquals()` using `fail()`

Some cases where it is useful:

1- mark a test that is incomplete, so it fails and warns you until you can finish it

2- making sure an exception is thrown:

There are three states that your test case can end up in
Passed: The function under test executed successfully and returned data as expected
Not Passed: The function under test executed successfully but the returned data was not as expected
Failed: The function did not execute successfully and this was not intended (Unlike negative test cases that expect a exception to occur).

If you are using eclipse there three states are indicated by a **Green**, **Blue** and **red** marker respectively.

We can use the fail operation for the third scenario.
e.g.:

```
public Integer add(integer a, Integer b) { return new Integer(a.intValue() + b.intValue())}
```

Passed Case: a = new Interger(1), b= new Integer(2) and the function returned 3
Not Passed Case: a = new Interger(1), b= new Integer(2) and the function returned any value other than 3
Failed Case: a = null , b = null and the function throws a NullPointerException

Two-Dimensional Arrays

» Thus far, you have used one-dimensional arrays to model linear collections of elements. You can use a two-dimensional array to represent a matrix or a table.

Declaring Variables of Two-Dimensional Arrays and Creating Two-Dimensional Arrays

» Here is the syntax for declaring a two-dimensional array:

```
dataType[][] arrayRefVar;
```

» As an example, here is how you would declare a two-dimensional array variable matrix of int values:

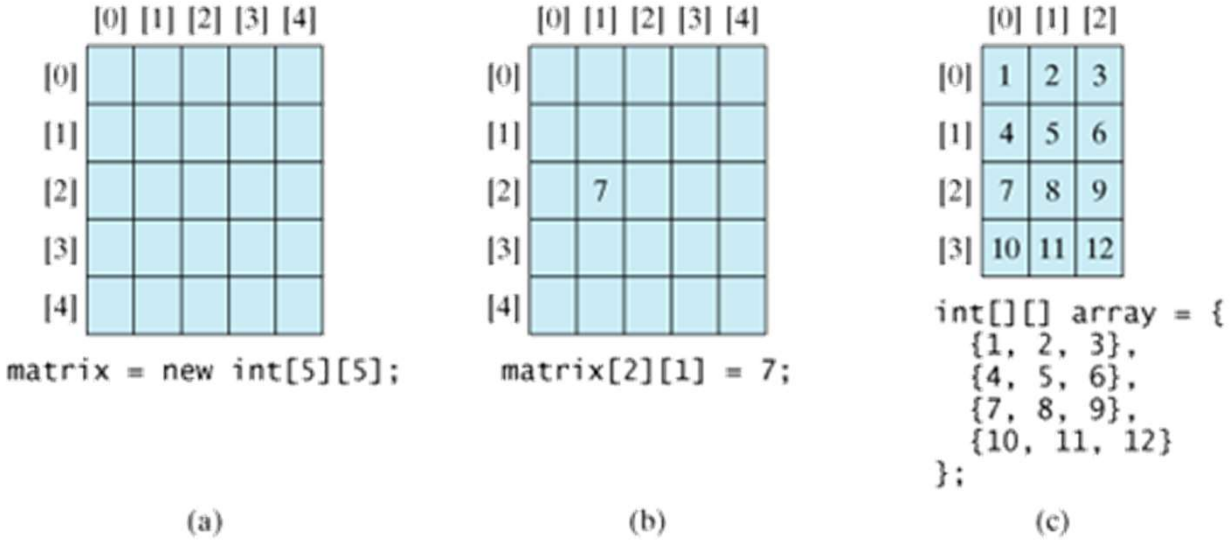
```
int[][] matrix;
```

Two-Dimensional Arrays

» You can create a two-dimensional array of 5 by 5 int values and assign it to matrix using this syntax:

```
matrix = new int[5][5];
```

» Two subscripts are used in a two-dimensional array, one for the row, and the other for the column. As in a one-dimensional array, the index for each subscript is of the int type and starts from 0.

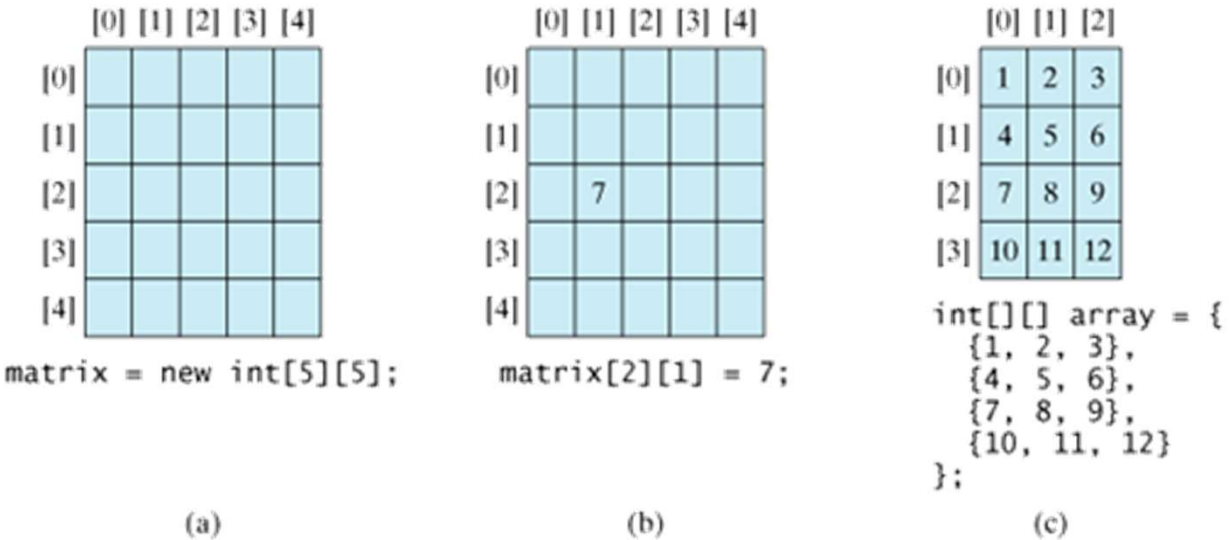


Two-Dimensional Arrays

» You can create a two-dimensional array of 5 by 5 int values and assign it to matrix using this syntax:

```
matrix = new int[5][5];
```

» Two subscripts are used in a two-dimensional array, one for the row, and the other for the column. As in a one-dimensional array, the index for each subscript is of the int type and starts from 0.



Two-Dimensional Arrays

» You can also use an array initializer to declare, create, and initialize a two-dimensional array. For example, the following code in (a) creates an array with the specified initial values, This is equivalent to the code in (b).

```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

(a)

Equivalent

```
int[][] array = new int[4][3];
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

(b)

Two-Dimensional Arrays

Obtaining the Lengths of Two-Dimensional Arrays

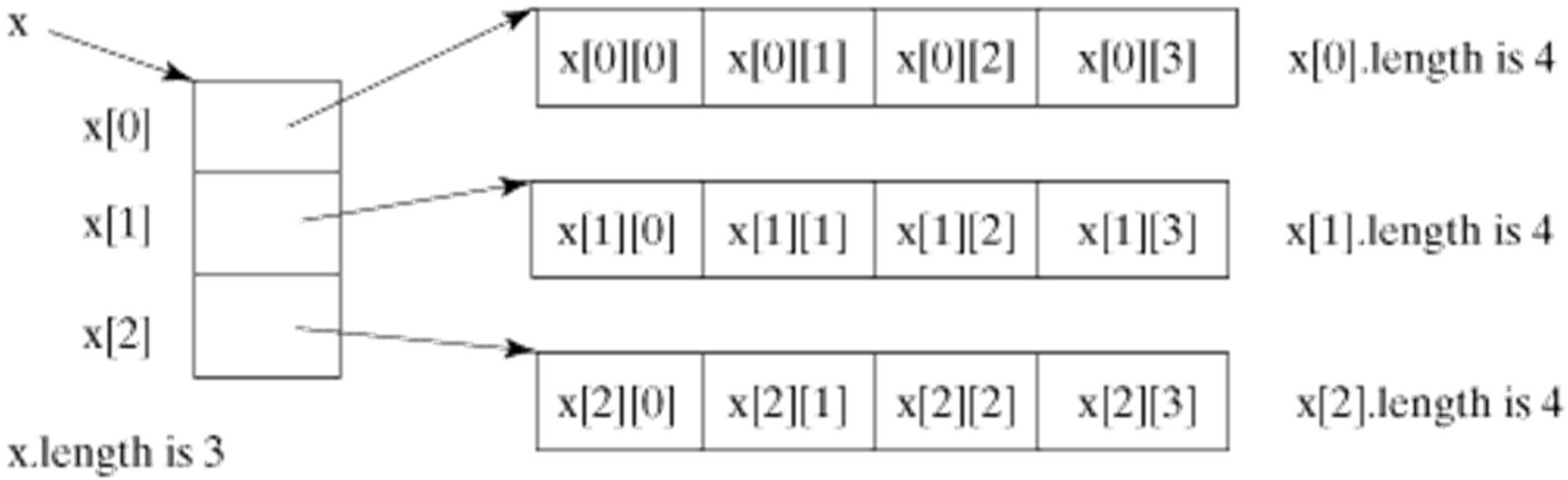
» A two-dimensional array is actually an array in which each element is a one-dimensional array. The length of an array x is the number of elements in the array, which can be obtained using x.length. x[0], x[1], …, and x[x.length-1] are arrays. Their lengths can be obtained using x[0].length, x[1].length, …, and x[x.length-1].length.

» For example, suppose x = new int[3][4], x[0], x[1], and x[2] are one-dimensional arrays and each contains four elements, x.length is 3, and x[0].length, x[1].length, and x[2].length are 4.

Two-Dimensional Arrays

Obtaining the Lengths of Two-Dimensional Arrays

» A two-dimensional array is a one-dimensional array in which each element is another one-dimensional array.

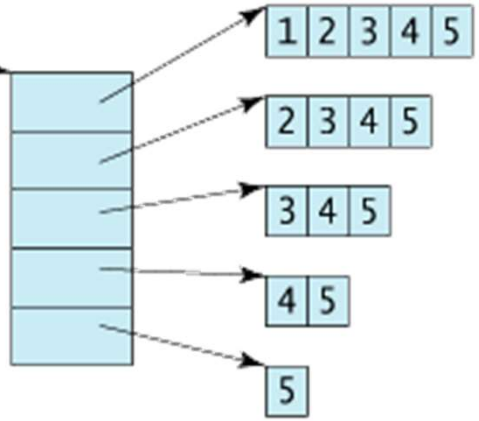


Two-Dimensional Arrays

Ragged Arrays

» Each row in a two-dimensional array is itself an array. Thus the rows can have different lengths. An array of this kind is known as a *ragged array*. Here is an example of creating a ragged array:

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



» As can be seen, triangleArray[0].length is 5, triangleArray[1].length is 4, triangleArray[2].length is 3, triangleArray[3].length is 2, and triangleArray[4].length is 1.

Two-Dimensional Arrays

Processing Two-Dimensional Arrays

» (Initializing arrays with random values) The following loop initializes the array with random values between 0 and 99:

```
package arrays;
    public class initialize_two_timensional {
        public static void main(String args[]) {

      int[][] matrix = new int[5][5];

      for(int row = 0; row < matrix.length; row++)
        {
            for(int column = 0; column < matrix[row].length; column++)
            {
                matrix[row][column] = (int)(Math.random() * 100);
            }
        }

        }
    }
```

Two-Dimensional Arrays

Processing Two-Dimensional Arrays

» (Printing arrays) To print a two-dimensional array, you have to print each element in the array using a loop like the following:

```
package arrays;
    public class initialize_two_timensional {
        public static void main(String args[]) {

      int[][] matrix = new int[5][5];

      for(int row = 0; row < matrix.length; row++)
        {
            for(int column = 0; column < matrix[row].length; column++)
            {
                matrix[row][column] = (int)(Math.random() * 100);
                System.out.println(matrix[row][column]+ " ");
            }
            System.out.println();
        }

        }
    }
```

Two-Dimensional Arrays

Processing Two-Dimensional Arrays

» (Summing all elements) Use a variable named total to store the sum. Initially total is 0. Add each element in the array to total using a loop like this:

```
package arrays;
    public class initialize_two_timensional {
        public static void main(String args[]) {

      int[][] matrix = {
              {1,2,3,6},
              {1,2,3,5},
              {4,5,6,8},
              {1,2,5,3}
              };

      int total = 0;
      for(int row = 0; row < matrix.length; row++)
        {
            for(int column = 0; column < matrix[row].length; column++)
            {
               total += matrix[row][column];
                System.out.println(total+ " ");
            }
            System.out.println();
        }
```

More 2D Array Examples

- `PrintRow2D.java`
- `PrintCol2D.java`