Arrays II

Lecture 15, Thu Mar 2 2006

based on slides by Kurt Eiselt

http://www.cs.ubc.ca/~tmm/courses/cpsc111-06-spr
News

- Assignment 2
  - do not wait until last minute
  - corrections to ASCIIArtiste.java posted (Sun)
  - definitely read WebCT bboards!
- Remember CSLC available!
  - Mon-Thu 10-6, Fri 10-4, x150 (near Reboot)
- extra TA lab coverage for A2 help:
  - Sun 12-2 Parker
  - Tue 4-6 Hastings, 6-8 Leavitt
Reading

- This week: 8.1, 8.5-8.7, topics 6.3 and 6.4

- Next week: no new reading
Common confusion on midterm: what's wrong?

```java
public class RoachPopulation {
    private int myPopulation;
    private static final double KILL_PERCENT = 0.10;
    public RoachPopulation(int population) {
        this.myPopulation = population;
    }
    public void waitForDoubling(int newPopulation)
        newPopulation *= 2;
    }
    public int getRoaches() {
        return this.myPopulation
    }
}
```
Recap: Array Declaration and Types

- Just like ordinary variable, must
  - declare array before we use it
  - give array a type

- Since cansSold contains integers, make integer array:

  ```
  int[] cansSold = new int[10]
  ```

- Looks like variable declaration, except:
  - empty brackets on the left tell Java that cansSold is an array...
  - the number in the brackets on the right tell Java that array should have room for 10 elements when it's created
## Recap: Array Declaration and Types

```java
public class ArrayTest1 {
    public static void main(String[] args) {
        final int ARRAYSIZE = 10;
        int[] cansSold = new int[ARRAYSIZE];
        cansSold[0] = 185;
        cansSold[1] = 92;
        cansSold[2] = 370;
        cansSold[3] = 485;
        cansSold[5] = 128;
        cansSold[6] = 84;
        cansSold[7] = 151;
        cansSold[8] = 32;
        cansSold[9] = 563;

        // do useful stuff here
        System.out.println("Element 4 is "+
                           cansSold[4]);
    }
}
```
Recap: Array Declaration and Types

```java
public class ArrayTest2
{
    public static void main(String[] args)
    {
        int[] cansSold = {185, 92, 370, 485, 209, 128, 84, 151, 32, 563};

        // do useful stuff here
        System.out.println("Element 4 is " + cansSold[4]);
    }
}
```

- Can also use **initializer list**
- Right side of declaration does not include type or size
  - Java figures out size by itself
- Types of values on right must match type declared on left
- Initializer list may only be used when array is first declared
Now use same data as basis for histogram

Write one loop to look at value associated with each row of array

- For each value print a line with that many asterisks
- For example, if program reads value 6 from the array, should print line of 6 asterisks
- Program then reads the value 8, prints a line of 8 asterisks, and so on.

Need outer loop to read individual values in the array

Need inner loop to print asterisks for each value
Objectives

- Understanding when and how to use
  - arrays of objects
  - 2D arrays
### Storing Different Data Types

<table>
<thead>
<tr>
<th>cansSold</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>185</td>
</tr>
<tr>
<td>1</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>370</td>
</tr>
<tr>
<td>3</td>
<td>485</td>
</tr>
<tr>
<td>4</td>
<td>209</td>
</tr>
<tr>
<td>5</td>
<td>128</td>
</tr>
<tr>
<td>6</td>
<td>84</td>
</tr>
<tr>
<td>7</td>
<td>151</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td>9</td>
<td>563</td>
</tr>
</tbody>
</table>
## Storing Different Data Types

<table>
<thead>
<tr>
<th>cansSold</th>
<th>cashIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 185</td>
<td>0 201.25</td>
</tr>
<tr>
<td>1 92</td>
<td>1 100.50</td>
</tr>
<tr>
<td>2 370</td>
<td>2 412.75</td>
</tr>
<tr>
<td>3 485</td>
<td>3 555.25</td>
</tr>
<tr>
<td>4 209</td>
<td>4 195.00</td>
</tr>
<tr>
<td>5 128</td>
<td>5 160.00</td>
</tr>
<tr>
<td>6 84</td>
<td>6 105.00</td>
</tr>
<tr>
<td>7 151</td>
<td>7 188.75</td>
</tr>
<tr>
<td>8 32</td>
<td>8 40.00</td>
</tr>
<tr>
<td>9 563</td>
<td>9 703.75</td>
</tr>
</tbody>
</table>

Could use two arrays of same size but with different types
## Storing Different Data Types

- Write program to compare what's been collected from each machine vs. how much should have been collected?

<table>
<thead>
<tr>
<th>cansSold</th>
<th>cashIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 185</td>
<td>0 201.25</td>
</tr>
<tr>
<td>1 92</td>
<td>1 100.50</td>
</tr>
<tr>
<td>2 370</td>
<td>2 412.75</td>
</tr>
<tr>
<td>3 485</td>
<td>3 555.25</td>
</tr>
<tr>
<td>4 209</td>
<td>4 195.00</td>
</tr>
<tr>
<td>5 128</td>
<td>5 160.00</td>
</tr>
<tr>
<td>6 84</td>
<td>6 105.00</td>
</tr>
<tr>
<td>7 151</td>
<td>7 188.75</td>
</tr>
<tr>
<td>8 32</td>
<td>8 40.00</td>
</tr>
<tr>
<td>9 563</td>
<td>9 703.75</td>
</tr>
</tbody>
</table>

Could use two arrays of same size but with different types
Storing Different Data Types

- Write program to compare what's been collected from each machine vs. how much should have been collected?

```java
public class ArrayTest4 {
    public static void main(String[] args) {
        double expected;
        int[] cansSold = {185, 92, 370, 485, 209, 128, 84, 151, 32, 563};
        double[] cashIn = {201.25, 100.50, 412.75, 555.25, 195.00, 160.00, 105.00, 188.75, 40.00, 703.75};
        for (int i = 0; i < cansSold.length; i++) {
            expected = cansSold[i] * 1.25;
            System.out.println("Machine " + (i + 1) + " off by "+
                                 (expected - cashIn[i]));
        }
    }
}
```

Could use two arrays of same size but with different types
Could use two arrays of same size but with different types

What happens when we run the program?

public class ArrayTest4{
    public static void main(String[] args){
        double expected;
        int[] cansSold = {185, 92, 370, 485, 209, 128, 84, 151, 32, 563};
        double[] cashIn = {201.25, 100.50, 412.75, 555.25, 195.00, 160.00, 105.00, 188.75, 40.00, 703.75};
        for (int i = 0; i < cansSold.length; i++){
            expected = cansSold[i] * 1.25;
            System.out.println("Machine " + (i + 1) + " off by $" + (expected - cashIn[i]));
        }
    }
}
### Storing Different Data Types

<table>
<thead>
<tr>
<th>cansSold</th>
<th>cashIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 201.25</td>
</tr>
<tr>
<td>1</td>
<td>1 100.50</td>
</tr>
<tr>
<td>2</td>
<td>2 412.75</td>
</tr>
<tr>
<td>3</td>
<td>3 555.25</td>
</tr>
<tr>
<td>4</td>
<td>4 195.00</td>
</tr>
<tr>
<td>5</td>
<td>5 160.00</td>
</tr>
<tr>
<td>6</td>
<td>6 105.00</td>
</tr>
<tr>
<td>7</td>
<td>7 188.75</td>
</tr>
<tr>
<td>8</td>
<td>8 40.00</td>
</tr>
<tr>
<td>9</td>
<td>9 703.75</td>
</tr>
</tbody>
</table>

Machine 0 off by $30.0
Machine 1 off by $14.5
Machine 2 off by $49.75
Machine 3 off by $51.0
Machine 4 off by $66.25
Machine 5 off by $0.0
Machine 6 off by $0.0
Machine 7 off by $0.0
Machine 8 off by $0.0
Machine 9 off by $0.0

Somebody has been stealing from the machines after all! We need an anti-theft plan…
Arrays With Non-Primitive Types

Great if you're always storing primitives like integers or floating point numbers

- What if we want to store String types too?
- Remember that String is an object, not a primitive data type

<table>
<thead>
<tr>
<th>cansSold</th>
<th>cashIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 185</td>
<td>0 201.25</td>
</tr>
<tr>
<td>1  92</td>
<td>1 100.50</td>
</tr>
<tr>
<td>2 370</td>
<td>2 412.75</td>
</tr>
<tr>
<td>3 485</td>
<td>3 555.25</td>
</tr>
<tr>
<td>4 209</td>
<td>4 195.00</td>
</tr>
<tr>
<td>5 128</td>
<td>5 160.00</td>
</tr>
<tr>
<td>6  84</td>
<td>6 105.00</td>
</tr>
<tr>
<td>7 151</td>
<td>7 188.75</td>
</tr>
<tr>
<td>8  32</td>
<td>8  40.00</td>
</tr>
<tr>
<td>9 563</td>
<td>9 703.75</td>
</tr>
</tbody>
</table>
## Arrays With Non-Primitive Types

<table>
<thead>
<tr>
<th>cansSold</th>
<th>cashIn</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>185</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>92</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>370</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>485</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>209</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>128</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>84</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>151</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>563</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>201.25</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100.50</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>412.75</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>555.25</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>195.00</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>160.00</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>105.00</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>188.75</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>40.00</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>703.75</td>
<td>9</td>
</tr>
</tbody>
</table>

- Then we create **array of objects**
  - In this case objects will be **Strings**
  - Array won't hold actual object
    - holds references: pointers to objects

```java
String[] location = new String[10];
```
Arrays of Objects

Now we can put references to Strings in our String array.

location[0] = "Chan Centre";}
### Arrays of Objects

Now we can put references to Strings in our String array.

```java
location[0] = "Chan Centre";
location[1] = "Law School";
```
Arrays of Objects

- Now we can put references to Strings in our String array.

```java
location[0] = "Chan Centre";
location[1] = "Law School";
location[2] = "Main Library";
```
Arrays of Objects

<table>
<thead>
<tr>
<th>cansSold</th>
<th>cashIn</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>185</td>
<td>0: &quot;Chan Centre&quot;</td>
</tr>
<tr>
<td>1</td>
<td>92</td>
<td>1: &quot;Law School&quot;</td>
</tr>
<tr>
<td>2</td>
<td>370</td>
<td>2: &quot;Main Library&quot;</td>
</tr>
<tr>
<td>3</td>
<td>485</td>
<td>3: &quot;Koerner Library&quot;</td>
</tr>
<tr>
<td>4</td>
<td>209</td>
<td>4: &quot;Business&quot;</td>
</tr>
<tr>
<td>5</td>
<td>128</td>
<td>5: &quot;Biology&quot;</td>
</tr>
<tr>
<td>6</td>
<td>84</td>
<td>6: &quot;Education&quot;</td>
</tr>
<tr>
<td>7</td>
<td>151</td>
<td>7: &quot;Applied Science&quot;</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>8: &quot;Agriculture&quot;</td>
</tr>
<tr>
<td>9</td>
<td>563</td>
<td>9: &quot;Computer Science&quot;</td>
</tr>
</tbody>
</table>

Now we can put references to Strings in our String array.

```java
location[0] = "Chan Centre";
location[1] = "Law School";
location[2] = "Main Library";
...and so on...
```
Arrays of Objects

<table>
<thead>
<tr>
<th>cansSold</th>
<th>cashIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 185</td>
<td>0 201.25</td>
</tr>
<tr>
<td>1 92</td>
<td>1 100.50</td>
</tr>
<tr>
<td>2 370</td>
<td>2 412.75</td>
</tr>
<tr>
<td>3 485</td>
<td>3 555.25</td>
</tr>
<tr>
<td>4 209</td>
<td>4 195.00</td>
</tr>
<tr>
<td>5 128</td>
<td>5 160.00</td>
</tr>
<tr>
<td>6 84</td>
<td>6 105.00</td>
</tr>
<tr>
<td>7 151</td>
<td>7 188.75</td>
</tr>
<tr>
<td>8 32</td>
<td>8 40.00</td>
</tr>
<tr>
<td>9 563</td>
<td>9 703.75</td>
</tr>
</tbody>
</table>

- Or we could have done this:

```java
String[] location = {
    "Chan Centre", "Law School", "Main Library", ....
};
```
Arrays of Objects

- Each individual String object in array of course has all String methods available
- For example, what would this return?

```
location[2].length()
```
Arrays of Objects

- Each individual String object in array of course has all String methods available
- For example, what would this return?

```java
location[2].length() // 12
```
Think about a cleaner way to do all this…
Arrays and Object Design

- Multiple array approach not very object-oriented
  - can create arrays of objects
  - can create objects of our own design...

<table>
<thead>
<tr>
<th>location</th>
<th>cashIn</th>
<th>cansSold</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 185</td>
<td>0 201.25</td>
<td>0 185</td>
</tr>
<tr>
<td>1 92</td>
<td>1 100.50</td>
<td>1 92</td>
</tr>
<tr>
<td>2 370</td>
<td>2 412.75</td>
<td>2 370</td>
</tr>
<tr>
<td>3 485</td>
<td>3 555.25</td>
<td>3 485</td>
</tr>
<tr>
<td>4 209</td>
<td>4 195.00</td>
<td>4 209</td>
</tr>
<tr>
<td>5 128</td>
<td>5 160.00</td>
<td>5 128</td>
</tr>
<tr>
<td>6 84</td>
<td>6 105.00</td>
<td>6 84</td>
</tr>
<tr>
<td>7 151</td>
<td>7 188.75</td>
<td>7 151</td>
</tr>
<tr>
<td>8 32</td>
<td>8 40.00</td>
<td>8 32</td>
</tr>
<tr>
<td>9 563</td>
<td>9 703.75</td>
<td>9 563</td>
</tr>
</tbody>
</table>

"Chan Centre" "Law School" "Main Library" "Koerner Library" "Business" "Biology" "Education" "Applied Science" "Agriculture" "Computer Science"
Arrays and Object Design

- Cokematic object design - contains
  - number of cans remaining: integer
  - location: String,
  - number of cans sold: integer
  - cash collected: double
Cokematic

- Cokematic object design - contains
  - number of cans remaining: integer
  - location: String,
  - number of cans sold: integer
  - cash collected: double

```java
public class Cokematic {
    private int numberOfCans;
    private String location;
    private int cansSold;
    private double cashIn;

    public Cokematic(int cans, String loc, int sold, double cash) {
        numberOfCans = cans;
        location = loc;
        cansSold = sold;
        cashIn = cash;
        System.out.println("Adding machine");
    }
}
```
Cokematic

- Cokematic object design - contains
  - number of cans remaining: integer
  - location: String,
  - number of cans sold: integer
  - cash collected: double

```java
public void buyCoke()
{
  if (numberOfCans > 0)
  {
    numberOfCans = numberOfCans - 1;
    cansSold = cansSold + 1;
    cashIn = cashIn + 1.25;
    System.out.println("Have a Coke");
    System.out.println(numberOfCans + " remaining");
  }
  else
  {
    System.out.println("Sold out.");
  }
}
```
public String getLocation()
{
    return location;
}

public int getCansSold()
{
    return cansSold;
}

public double getCashIn()
{
    return cashIn;
}

public void reloadMachine(int newCans)
{
    numberOfCans = numberOfCans + newCans;
    System.out.println("reloading machine");
}

public int getNumberOfCans()
{
    return numberOfCans;
}

public String toString()
{
    return (location + " sold: " + cansSold + " left: " + numberOfCans + " made: " + cashIn);
}
In driver, executing

Cokematic machine1 = new Cokematic(100, "Chan Centre", 185, 201.25);
Cokematic

- In driver, executing

```java
Cokematic machine1 = new Cokematic(100, "Chan Centre", 185, 201.25);
```

- Results in

```
Cokematic
- numberOfCans: 100
- location: "Chan Centre"
- cansSold: 185
- cashIn: 201.25

machine1
```
Cokematic

- In driver, executing

```
Cokematic machine1 = new Cokematic(100, "Chan Centre", 185, 201.25);
```

- Results in

```
Cokematic

- numberOfCans: 100
- location: "Chan Centre"
- cansSold: 185
- cashIn: 201.25

- buyCoke()
- getLocation()
- getCansSold()
- getCashIn()
- reloadMachine()
- toString()
```

- Note: leaving out methods in UML diagrams from now on to fit on page
Cokematic

- In driver, executing

```java
Cokematic machine1 = new Cokematic(100, "Chan Centre", 185, 201.25);
```

- Results in

```
Cokematic
- numberOfCans: 100
- location: "Chan Centre"
- cansSold: 185
- cashIn: 201.25
```

- Note: leaving out methods in UML diagrams from now on to fit on page
public class CokeEmpire
{
    private Cokematic[] collection; // what does this do?

    public CokeEmpire()
    {
        collection = new Cokematic[10]; // what does this do?
    }

    public void addCokematic(int index, int cans, String loc, int sold, double cash)
    {
        collection[index] = new Cokematic(cans, loc, sold, cash);
    }

    public Cokematic getCokematic(int index)
    {
        return collection[index];
    }
}
CokeEmpire

- In driver, executing:

CokeEmpire myMachines = new CokeEmpire();
In driver, executing

CokeEmpire myMachines = new CokeEmpire();

results in
CokeEmpire

- Populate array with Cokematic objects

```cpp
myMachines = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```
CokeEmpire

- Populate array with Cokematic objects

```java
myMachines.addCokematic(0, 100, "Chan Centre", 185, 201.25);
```
CokeEmpire

Populate array with Cokematic objects

```java
myMachines.addCokematic(1, 150, "Law School", 92, 100.50);
```

- Cokematic
  - numberOfCans: 100
  - location: "Chan Centre"
  - cansSold: 185
  - cashIn: 201.25

- Cokematic
  - numberOfCans: 150
  - location: "Law School"
  - cansSold: 92
  - cashIn: 100.50
CokeEmpire

- Populate array with Cokematic objects

```java
myMachines.addCokematic(2, 200, "Main Library", 370, 412.75);
```

Cokematic
- numberOfCans: 100
- location: "Chan Centre"
- cansSold: 185
- cashIn: 201.25

Cokematic
- numberOfCans: 150
- location: "Law School"
- cansSold: 92
- cashIn: 100.50

Cokematic
- numberOfCans: 200
- location: "Main Library"
- cansSold: 370
- cashIn: 412.75
CokeEmpire

What does this return?

myMachines.getCokematic(1).getCansSold()
What does this return?

```java
myMachines.getCokematic(1).getCansSold()
```
What does this return?

```javascript
myMachines.getCokematic(1).getCansSold()
```
What does this return?

```java
myMachines.getCokematic(1).getCansSold()
```

<table>
<thead>
<tr>
<th>Cokematic</th>
<th>numberOfCans: 100</th>
<th>location: &quot;Chan Centre&quot;</th>
<th>cashIn: 201.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cokematic</td>
<td>numberOfCans: 150</td>
<td>location: &quot;Law School&quot;</td>
<td>cashIn: 100.50</td>
</tr>
<tr>
<td>Cokematic</td>
<td>numberOfCans: 200</td>
<td>location: &quot;Main Library&quot;</td>
<td>cashIn: 412.75</td>
</tr>
</tbody>
</table>
Arrays of Arrays

0 1 2 3

0 1 2 0 1 2 0 1 2
Arrays of Arrays

- In any given array, all data must be of same type
Arrays of Arrays

- In any given array, all data must be of same type.
- All arrays in array of arrays must be of same type.
In any given array, all data must be of same type
All arrays in array of arrays must be of same type
So easier to use a two-dimensional array!
Two-Dimensional Arrays

- In Java, 2D array implemented internally as array of arrays
- but externally syntax of 2D array may seem easier to use
Two-Dimensional Arrays

- In Java, 2D array implemented internally as array of arrays
  - but externally syntax of 2D array may seem easier to use
- Typical control structure for computing with 2D array is nested loop
  - loop within another loop
- Let's write program to
  - load array with values shown
  - print contents of array
Two-Dimensional Arrays

```java
public class ArrayTest5 {
    public static void main(String[] args) {
    }
}
```

```
0     1     2
0     0     0
0     1     2
0     2     4
0     3     6
```

```
rows

columns
```

```java
public class ArrayTest5 {
    public static void main(String[] args) {
    }
}
```
public class ArrayTest5 {
    public static void main(String[] args) {
        int[][] multTable = new int[4][3];
        // Initialize the table
        multTable[0][0] = 0;
        multTable[0][1] = 0;
        multTable[0][2] = 0;
        multTable[1][0] = 0;
        multTable[1][1] = 1;
        multTable[1][2] = 2;
        multTable[2][0] = 0;
        multTable[2][1] = 2;
        multTable[2][2] = 4;
        multTable[3][0] = 0;
        multTable[3][1] = 3;
        multTable[3][2] = 6;
    }
}
Two-Dimensional Arrays

```java
public class ArrayTest5 {
    public static void main(String[] args) {
        int[][] multTable = new int[4][3];

        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                multTable[row][col] = row * col;
            }
        }
    }
}
```

```
<table>
<thead>
<tr>
<th>columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2</td>
</tr>
<tr>
<td>0 0 0</td>
</tr>
<tr>
<td>0 1 2</td>
</tr>
<tr>
<td>0 2 4</td>
</tr>
<tr>
<td>0 3 6</td>
</tr>
</tbody>
</table>
```

rows
```
columns

<table>
<thead>
<tr>
<th>0 1 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
</tr>
<tr>
<td>0 1 2</td>
</tr>
<tr>
<td>0 2 4</td>
</tr>
<tr>
<td>0 3 6</td>
</tr>
</tbody>
</table>
```
Two-Dimensional Arrays

public class ArrayTest5 {
    public static void main(String[] args) {
        int[][] multTable = new int[4][3];
        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                multTable[row][col] = row * col;
            }
        }
    }
}

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

0 1 2

rows

columns
Two-Dimensional Arrays

```java
public class ArrayTest5 {
    public static void main(String[] args) {
        int[][] multTable = new int[4][3];

        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                multTable[row][col] = row * col;
            }
        }

        for (int col = 0; col < multTable[row].length; col++) {
            System.out.print(multTable[row][col] + " ");
        }
    }
}
```
Two-Dimensional Arrays

```java
public class ArrayTest5 {
    public static void main(String[] args) {
        int[][] multTable = new int[4][3];

        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                multTable[row][col] = row * col;
            }
        }

        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                System.out.print(multTable[row][col] + " ");
            }
            System.out.println();
        }
    }
}
```

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

0 1 2

rows

columns
Two-Dimensional Arrays

```java
public class ArrayTest5 {
    public static void main(String[] args) {
        int[][] multTable = new int[4][3];

        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                multTable[row][col] = row * col;
            }
        }

        for (int row = 0; row < multTable.length; row++) {
            for (int col = 0; col < multTable[row].length; col++) {
                System.out.print(multTable[row][col] + " ");
            }
            System.out.println();
        }
    }
}
```

```
0 1 2
0 0 0
0 1 2
0 2 4
0 3 6
```