Arrays of Objects & 2D Arrays

CSC 1051 – Data Structures and Algorithms I

Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Course website:
www.csc.villanova.edu/~map/1051/

Some slides in this presentation are adapted from the slides accompanying Java Software Solutions by Lewis & Loftus
Arrays - Review

• Declaration:

```java
double[] scores = new double[10];
```

The entire array has a single name

- **element type**
- **index**
- **array element**

- **Declaration:**

```java
double[] scores = new double[10];
```

- **Initialization:**

```java
scores[0] = 7.9;
scores[1] = 8.7;
scores[2] = 9.4;
scores[3] = 8.2;
scores[4] = 6.7;
scores[5] = 9.8;
scores[6] = 8.7;
scores[7] = 8.1;
scores[8] = 7.4;
scores[9] = 9.1;
```

- ▶️ This array holds 10 values of type `double` that are indexed from 0 to 9
- ▶️ The size of the array is given by: `scores.length = 10`

CSC 1051 M.A. Papalaskari, Villanova University
Example: An array of Strings

- `String[] words = new String[5];`

At this point, the following line of code would throw a `NullPointerException`:

```java
System.out.println(words[0]);
```
Example: An array of Strings

- `String[] words = new String[5];`
- Now, store some `String` objects in the array:

  ```java
  words[0] = "friendship";
  words[1] = "loyalty";
  words[2] = "honor";
  ```

```
  words
  "friendship"
  "loyalty"
  "honor"
```
Arrays of Objects

- The following declaration creates an array object called `verbs` and fills it with four `String` objects created using string literals

```java
String[] verbs = {"play", "work", "eat", "sleep", "run"};
```
Arrays of Objects

- Example: managing a collection of DVD objects
public class Movies {
    public static void main (String[] args) {
        DVDCollection movies = new DVDCollection();
        movies.addDVD("The Godfather", "Francis Ford Coppala", 1972, 24.95, true);
        movies.addDVD("District 9", "Neill Blomkamp", 2009, 19.95, false);
        movies.addDVD("Iron Man", "Jon Favreau", 2008, 15.95, false);
        movies.addDVD("All About Eve", "Joseph Mankiewicz", 1950, 17.50, false);
        movies.addDVD("The Matrix", "Andy & Lana Wachowski", 1999, 19.95, true);
        System.out.println(movies);
        System.out.println(movies);
        System.out.println(movies);
    }
}
public class Movies {
  public static void main(String[] args) {
    DVDCollection movies = new DVDCollection();
    movies.addDVD("The Godfather", "Francis Ford Coppola", 1972, 24.95, true);
    movies.addDVD("District 9", "Neill Blomkamp", 2009, 19.95, false);
    movies.addDVD("Iron Man", "Jon Favreau", 2008, 15.95, false);
    movies.addDVD("All About Eve", "Joseph Mankiewicz", 1950, 17.50, false);
    movies.addDVD("The Matrix", "Andy & Lana Wachowski", 1999, 19.95, true);
    System.out.println(movies);
    movies.addDVD("Iron Man 2", "Jon Favreau", 2010, 22.99, false);
    movies.addDVD("Casablanca", "Michael Curtiz", 1942, 19.95, false);
    System.out.println(movies);
  }
}
Output

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
My DVD Collection
Number of DVDs: 5
Total cost: $98.30
Average cost: $19.66
DVD List:
$24.95  1972  The Godfather  Francis Ford Coppala  Blu-Ray
$19.95  2009  District 9  Neill Blomkamp
$15.95  2008  Iron Man  Jon Favreau
$17.50  1950  All About Eve  Joseph Mankiewicz
$19.95  1999  The Matrix  Andy & Lana Wachowski  Blu-Ray
System.out.println (movies);

Output (continued)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
My DVD Collection
Number of DVDs: 7
Total cost: $141.24
Average cost: $20.18
DVD List:
$24.95  1972  The Godfather  Francis Ford Coppala  Blu-Ray
$19.95  2009  District 9  Neill Blomkamp
$15.95  2008  Iron Man  Jon Favreau
$17.50  1950  All About Eve  Joseph Mankiewicz
$19.95  1999  The Matrix  Andy & Lana Wachowski  Blu-Ray
$22.99  2010  Iron Man 2  Jon Favreau
$19.95  1942  Casablanca  Michael Curtiz
System.out.println (movies);
import java.text.NumberFormat;

public class DVDCollection
{
    private DVD[] collection;
    private int count;
    private double totalCost;

    //-----------------------------------------------------------------
    // Constructor: Creates an initially empty collection.
    //-----------------------------------------------------------------
    public DVDCollection ()
    {
        collection = new DVD[100];
        count = 0;
        totalCost = 0.0;
    }
}

continue
// Adds a DVD to the collection, increasing the size of the collection array if necessary.

public void addDVD (String title, String director, int year, double cost, boolean bluRay)
{
    if (count == collection.length)
        increaseSize();
    collection[count] = new DVD (title, director, year, cost, bluRay);
    totalCost += cost;
    count++;
}

continue
public String toString() {

    NumberFormat fmt = NumberFormat.getCurrencyInstance();
    String report = "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";
    report += "My DVD Collection\n\n";

    report += "Number of DVDs: " + count + "\n";
    report += "Total cost: " + fmt.format(totalCost) + "\n";
    report += "Average cost: " + fmt.format(totalCost/count);

    report += "\n\nDVD List:\n\n";

    for (int dvd = 0; dvd < count; dvd++)
        report += collection[dvd].toString() + "\n";

    return report;
}
continue

// Increases the capacity of the collection by creating a larger array and copying the existing collection into it.
private void increaseSize ()
{
    DVD[] temp = new DVD[collection.length * 2];

    for (int dvd = 0; dvd < collection.length; dvd++)
        temp[dvd] = collection[dvd];

    collection = temp;
}
import java.text.NumberFormat;

public class DVD {
  private String title, director;
  private int year;
  private double cost;
  private boolean bluRay;

  //--- Represents a DVD video disc. ---/
  public DVD (String title, String director, int year, double cost, boolean bluRay) {
    this.title = title;
    this.director = director;
    this.year = year;
    this.cost = cost;
    this.bluRay = bluRay;
  }
}

continue
public String toString() {
    NumberFormat fmt = NumberFormat.getCurrencyInstance();
    String description;
    description = fmt.format(cost) + "\t" + year + "\t";
    description += title + "\t" + director;
    if (bluRay)
        description += "\t" + "Blu-Ray";
    return description;
}
Arrays as Parameters

• An entire array can be passed as a parameter to a method (just like any other object). For example:

```java
int[] ratings = {4, 3, 3, 1, 4, 3, 1, 0, 3, 4};
average(ratings);
```

• Assumes a definition for method `average()`:

```java
public double average(int[] a)
{
    for (int num: a)
        sum += num;
    return ((double)sum/a.length);
}
```
Try this: Write a method that adds 2 to the value of each element in an array of type `double[]`.
Command-Line Arguments

• It turns out we have been using arrays as parameters all along!

```java
public static void main (String[] args)
```
Command-Line Arguments

• It turns out we have been using arrays as parameters all along!

```java
public class Test {
    public static void main (String[] args)
    {
        System.out.println ();
        System.out.println (" " + args[0]);
        System.out.println (" " + args[1]);
    }
}
```

• These values come from command-line arguments that are provided when the interpreter is invoked

• jGrasp calls them “Run Arguments”
What does it mean to “copy of an array”? 

• Suppose we have two arrays:

```java
int[] a = {147, 323, 89, 933};
int[] b = {100, 200, 300, 400};
```

```java
for (int i=0; i<a.length; i++)
    a[i] = b[i];
```
What does it mean to “copy of an array”?

• Suppose we have two arrays:
  ```java
  int[] a = {147, 323, 89, 933};
  int[] b = {100, 200, 300, 400};
  ```

Copying elements vs. copying array variable:

• Compare the following:

  ```java
  for (int i=0; i<a.length; i++)
      a[i] = b[i];
  ```

• Now, what is the effect of the following?

  ```java
  a[1] = 1111;
  b[2] = 2222;
  ```
1) Copying elements:

```java
for (int i=0; i<a.length; i++)
    a[i] = b[i];
```

What changes?

<table>
<thead>
<tr>
<th>a</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>147</td>
<td>323</td>
<td>89</td>
<td>933</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
<td>200</td>
<td>300</td>
<td>400</td>
</tr>
</tbody>
</table>

```java
a[1] = 1111;
b[2] = 2222;
```
2) Copying array variables:

What changes?

```
0  1  2  3
a  147 323 89 933
b  100 200 300 400
```

```
a = b;
```

```
a[1] = 1111;
b[2] = 2222;
```
Two-Dimensional Arrays

- A *one-dimensional array* stores a list of elements
- A *two-dimensional array* can be thought of as a table of elements, with rows and columns
Two-Dimensional Arrays

• To be precise, in Java a two-dimensional array is an array of arrays

• A two-dimensional array is declared by specifying the size of each dimension separately:

```
int[][] table = new int[12][50];
```

• A array element is referenced using two index values:

```
value = table[3][6]
```

• The array stored in one row can be specified using one index
2D Arrays - Overview

declaration

double[][][] courseGrade = new double[3][10];

2D array element

courseGrade[1][4]

array element (a row)
courseGrade[2]
Two-Dimensional Arrays

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>int[][]</td>
<td>2D array of integers, or array of integer arrays</td>
</tr>
<tr>
<td>table[5]</td>
<td>int[]</td>
<td>array of integers</td>
</tr>
<tr>
<td>table[5][12]</td>
<td>int</td>
<td>integer</td>
</tr>
</tbody>
</table>

- See TwoDArray.java
- See SodaSurvey.java
public class TwoDArray
{
  //-------------------------------------------------------------
  // Creates a 2D array of integers, fills it with increasing
  // integer values, then prints them out.
  //-------------------------------------------------------------
  public static void main (String[] args)
  {
    int[][] table = new int[5][10];

    // Load the table with values
    for (int row=0; row < table.length; row++)
      for (int col=0; col < table[row].length; col++)
        table[row][col] = row * 10 + col;

    // Print the table
    for (int row=0; row < table.length; row++)
    {
      for (int col=0; col < table[row].length; col++)
        System.out.print (table[row][col] + "\t");
      System.out.println();
    }
  }
}
```java
public class TwoDArray {

    public static void main (String[] args) {

        int[][] table = new int[5][10];

        // Load the table with values
        for (int row=0; row < table.length; row++) {
            for (int col=0; col < table[row].length; col++)
                table[row][col] = row * 10 + col;

        }

        // Print the table
        for (int row=0; row < table.length; row++) {
            for (int col=0; col < table[row].length; col++)
                System.out.print (table[row][col] + "\t");
            System.out.println();
        }

    }
}
```
Multidimensional Arrays

- An array can have many dimensions – if it has more than one dimension, it is called a *multidimensional array*.

- Each dimension subdivides the previous one into the specified number of elements.

- Each dimension has its own *length* constant.

- Because each dimension is an array of array references, the arrays within one dimension can be of different lengths.
  - These are sometimes called *ragged arrays*.