Arrays, Part 2

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:**

The entire array has a single name

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

**Element type**

```
scores[2]
```

**Index**

```
scores[2] = 9.4;
```

**Array element**

```
scores[2] = 9.4;
```

**Size of array**

```
scores.length = 10
```

**Initialization:**

```
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;
```

**Declaration, Instantiation, & Initialization combined:**

```
double[] scores = {7.9, 8.7, 9.4, 8.2, 6.7, 9.8, 8.7, 8.1, 7.4, 9.1};
```
Arrays as Parameters

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

```java
// Draws a triangle and a V-shape using polygons and polylines.
public void paintComponent(Graphics page) {
    super.paintComponent(page);

    int[] xPoints = {100, 120, 150};
    int[] yPoints = {150, 40, 110};

    page.setColor(Color.cyan);
    page.fillPolygon(xPoints, yPoints, xPoints.length);

    page.setColor(Color.red);
    page.drawPolyline(xPoints, yPoints, xPoints.length);
}
```

see TrianglePanel.java
Arrays as Parameters

**Example:** A method that adds 3 to the value of each element in an array.

```java
// Draws a triangle and a V-shape using polygons and polylines.
public void paintComponent(Graphics page) {
    super.paintComponent(page);

    int[] xPoints = {100, 120, 150};
    int[] yPoints = {150, 40, 110};

    page.setColor(Color.cyan);
    page.fillPolygon(xPoints, yPoints, xPoints.length);

    addTen(xPoints);
    page.setColor(Color.red);
    page.drawPolyline(xPoints, yPoints, xPoints.length);
}

public void addTen(int[] a) {
    for (int i = 0; i < a.length; i++)
        a[i] += 10;
}
```

Example: A method that adds 3 to the value of each element in an array.
Write a method that adds \( n \) (an \texttt{int} \) to the value of each element in an array of type \texttt{int[]} .

Try this method with the TrianglePanel:

• add code to draw another triangle \textit{shifted by some amount} \( n \) in a \textit{different color}
It turns out we have been using arrays as parameters all along!

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 an array”? 

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

  Copying elements vs. copying array variables:

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

  a = b;

Afterwards, what is the effect of the following?

  a[1] = 0;
  b[2] = 0;
1) Copying elements:

Trace this code. What changes in the arrays?

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

a[1] = 0;
b[2] = 0;
```
2) Copying array variables:

a

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

b

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

Trace this code. What changes in the arrays?
Array parameters revisited

- How is using an array as a parameter like “copying an array”?

```java
// Draws a triangle and a V-shape using polygons and polylines.
public void paintComponent(Graphics page) {
    super.paintComponent(page);

    int[] xPoints = {100, 120, 150};
    int[] yPoints = {150, 40, 110};

    page.setColor(Color.cyan);
    page.fillPolygon(xPoints, yPoints, xPoints.length);

    addThree(xPoints);

    page.setColor(Color.red);
    page.drawPolyline(xPoints, yPoints, xPoints.length);
}

public void addThree(int[] a) {
    for (int i = 0; i < a.length; i++)
        a[i] += 3;
}
```
Managing a collection of objects

• Example: a Movie database (collection of DVD objects)

```
<table>
<thead>
<tr>
<th>DVD</th>
</tr>
</thead>
<tbody>
<tr>
<td>title : String</td>
</tr>
<tr>
<td>director : String</td>
</tr>
<tr>
<td>year : int</td>
</tr>
<tr>
<td>cost : double</td>
</tr>
<tr>
<td>blueray : boolean</td>
</tr>
</tbody>
</table>

+ toString() : String |
```
package java.text.NumberFormat;

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

    // Constructor: Creates a new DVD with the specified information.
    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
// Returns a string description of this DVD.
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;
}
Test client – create a few DVDs, print their info:

```java
//********************************************************************
// TestDVD.java          Author: M A Papalaskari
//
// Test client for DVD.java
//********************************************************************

public class TestDVD {
    //----------------------------------------------------------------------------
    // Creates some DVD objects and prints their info
    //----------------------------------------------------------------------------
    public static void main(String[] args) {
        DVD one = new DVD("Casablanca", "Michael Curtiz", 1942, 19.95, false);
        DVD two = new DVD("District 9", "Neill Blomkamp", 2009, 19.95, false);
        DVD three = new DVD("Iron Man", "Jon Favreau", 2008, 15.95, false);
        System.out.println(one);
        System.out.println(two);
        System.out.println(three);
    }
}
```
What if we want to store more DVDs?

```java
public class MyTenMovies {
//--- MyTenMovies.java
// Test client for DVD.java

public static void main(String[] args) {
    // Create and print infos
    DVD[] list = new DVD[10];
    list[0] = new DVD("Casablanca", "Michael Curtiz", 1942, 19.95, false);
    list[1] = new DVD("District 9", "Neill Blomkamp", 2009, 19.95, false);
    list[2] = new DVD("Iron Man", "Jon Favreau", 2008, 15.95, false);

    for (DVD item: list)
        System.out.println(item);
}
}

What if we want to store more DVDs?

Use an array of DVD objects:
```
What if we want to store more DVDs?

```java
public class MyTenMovies {
    public static void main(String[] args) {
        DVD[] list = new DVD[10];
        list[0] = new DVD("Casablanca", "Michael Curtiz", 1942, 19.95, false);
        list[1] = new DVD("District 9", "Neill Blomkamp", 2009, 19.95, false);
        list[2] = new DVD("Iron Man", "Jon Favreau", 2008, 15.95, false);

        for (DVD item: list) {
            System.out.println(item);
        }
    }
}
```

Next: A collection of DVD's that can grow to accommodate as many items as needed!
Managing a collection of objects

- Example: a Movie database (collection of DVD objects)

```
Movies.java

+ main (args : String[]) : void
```

```
DVDCollection.java

- collection : DVD[]
- count : int

+ addDVD(title : String, director : String,
  year : int, cost : double, blueray : boolean) : void
+ toString() : String
- increaseSize() : void
```

```
DVD.java

- title : String
- director : String
- year : int
- cost : double
- blueray : boolean

+ toString() : String
```
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);
    }
}
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);
        movies.addDVD("Iron Man 2", "Jon Favreau", 2010, 22.99, false);
        movies.addDVD("Casablanca", "Michael Curtiz", 1942, 19.95, false);
        System.out.println(movies);
    }
}
```java
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);
        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
continue

**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
```
```
import java.text.NumberFormat;

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

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

continue
// Returns a report describing the DVD collection.
// ---------------------------------------------------------
public String toString()
{
    NumberFormat fmt = NumberFormat.getCurrencyInstance();

    String report = ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    String report += "My DVD Collection\n\n"
    report += "Number of DVDs: " + count + "\n"
    report += "\n\nDVD List:\n\n"

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

    return report;
}
private void increaseSize()
{
    DVD[] temp = new DVD[collection.length * 2];

    for (int i = 0; i < collection.length; i++)
    {
        temp[i] = collection[i];
    }

    collection = temp;
}
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
2D Array Example

declaration

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

2D array element

courseGrade[1][4]

array element (a row)

courseGrade[2]
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();
        }
    }
}
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();
    }
}
## Two-Dimensional Arrays – Types?

<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>
import java.text.DecimalFormat;

public class SodaSurvey
{
    public static void main(String[] args)
    {
        int[][] scores = {
            {3, 4, 5, 2, 1, 4, 3, 2, 4, 4},
            {2, 4, 3, 4, 3, 3, 2, 1, 2, 2},
            {3, 5, 4, 5, 5, 3, 2, 5, 5, 5},
            {1, 1, 1, 3, 1, 2, 1, 3, 2, 4}
        };

        final int SODAS = scores.length;
        final int PEOPLE = scores[0].length;

        int[] sodaSum = new int[SODAS];
        int[] personSum = new int[PEOPLE];

        continue
    }
}
Another 2D Array Example from textbook

```java
for (int soda=0; soda < SODAS; soda++)
    for (int person=0; person < PEOPLE; person++)
        { 
            sodaSum[soda] += scores[soda][person];
            personSum[person] += scores[soda][person];
        }

    DecimalFormat fmt = new DecimalFormat ("0.#");
    System.out.println ("Averages:");
    for (int soda=0; soda < SODAS; soda++)
        System.out.println ("Soda #" + (soda+1) + ": " + fmt.format ((float)sodaSum[soda]/PEOPLE));

    for (int person=0; person < PEOPLE; person++)
        System.out.println ("Person #" + (person+1) + ": " + fmt.format ((float)personSum[person]/SODAS));
```

Output

Averages:
Soda #1: 3.2  
Soda #2: 2.6  
Soda #3: 4.2  
Soda #4: 1.9  
Person #1: 2.2  
Person #2: 3.5  
Person #3: 3.2  
Person #4: 3.5  
Person #5: 2.5  
Person #6: 3  
Person #7: 2  
Person #8: 2.8  
Person #9: 3.2  
Person #10: 3.8
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*