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:

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

Instantiation:

```
    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;
```

Initialization:

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

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 – 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>
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**

Managing a collection of objects

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

```java
DVD
-
  - title : String
  - director : String
  - year : int
  - cost : double
  - bluRay : boolean

+ toString() : String
```

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 ("0.0");
System.out.println ("Averages:");
for (int soda=0; soda < SODAS; soda++)
  System.out.println ("Soda #" + (soda+1) + " : " + fmt.format ((float)sodaSum[soda]/PEOPLE));

System.out.println ();
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
```
What if we want to store more DVDs?

* Use an **array** of DVD objects:

```java
// MyTenMovies.java  Author: M A Papalaskari
// Test client for DVD.java
//********************************************************************
public class MyTenMovies {
    // Creates some DVD objects and prints their info
    public static void main(String[] args) {
        DVD[] list = new DVD[10];
        list[0] = new DVD("Casablanca", "Michael Curtis", 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!

* No limit like this:

```java
DVD[] list = new DVD[10];
```
Arrays, Part 2

Managing a collection of objects

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

```java
public class Movies
{
    // Demonstrates the use of an array of objects.
    // Movie database (collection of DVD objects)

    public static void main(String[] args)
    {
        // Creates a DVDCollection object and adds some DVDs to it. Prints the status of the collection.
        DVDCollection movies = new DVDCollection();

        movies.addDVD("Casablanca", "Michael Curtiz", 1942, 19.95, false);
        movies.addDVD("Iron Man 2", "Jon Favreau", 2010, 22.99, false);
        movies.addDVD("All About Eve", "Joseph Mankiewicz", 1950, 17.50, false);
        movies.addDVD("District 9", "Neill Blomkamp", 2009, 19.95, false);
        movies.addDVD("The Godfather", "Francis Ford Coppola", 1972, 24.95, true);

        System.out.println("DVD List:"
            + 
            "Average cost: 
            + 
            "Total cost: 
            + 
            "Number of DVDs: 
            + 
            "
        );

        System.out.println("My DVD Collection
            + 
            "Number of DVDs: 5
            + 
            "Average cost: 
            + 
            "Total cost: 
            + 
            "Number of DVDs: 
            + 
            "
        );
    }
}
```

```java
public class DVD
{
    // The DVD class represents a DVD in the collection.

    private String title;
    private String director;
    private int year;
    private double cost;
    private boolean isBluRay;

    public DVD(String title, String director, int year, double cost, boolean isBluRay)
    {
        this.title = title;
        this.director = director;
        this.year = year;
        this.cost = cost;
        this.isBluRay = isBluRay;
    }
}
```

```java
public class DVDCollection
{
    private DVD[] dvds;
    private int size;

    public DVDCollection()
    {
        dvds = new DVD[10];
        size = 0;
    }

    public void addDVD(DVD dvd)
    {
        if (size < dvds.length)
        {
            dvds[size] = dvd;
            size++;
        }
    }

    public void print() // Prints the status of the collection.
    {
        System.out.println("My DVD Collection
            + 
            "Number of DVDs: 5
            + 
            "Average cost: 
            + 
            "Total cost: 
            + 
            "Number of DVDs: 
            + 
            "
        );
    }

    public void use(DVD[] dvds, int size)
    {
        // Use the DVDCollection object to perform some operations.
    }
}
```

---

Output

```
My DVD Collection
Number of DVDs: 5
Average cost: $19.66
Total cost: $98.30
DVD List:
- The Matrix, Andy & Lana Wachowski, 1999, 19.99, false
- Iron Man, Jon Favreau, 2008, 15.95, false
- The Godfather, Francis Ford Coppola, 1972, 24.95, true
- District 9, Neill Blomkamp, 2009, 19.95, false
- All About Eve, Joseph Mankiewicz, 1950, 17.50, false
```

My DVD Collection
Number of DVDs: 7
Average cost: $20.18
Total cost: $141.24
DVD List:
- The Matrix, Andy & Lana Wachowski, 1999, 19.99, false
- Iron Man 2, Jon Favreau, 2010, 22.99, false
- Casablanca, Michael Curtiz, 1942, 19.95, false
- District 9, Neill Blomkamp, 2009, 19.95, false
- All About Eve, Joseph Mankiewicz, 1950, 17.50, false
- The Godfather, Francis Ford Coppola, 1972, 24.95, true
- The Matrix, Andy & Lana Wachowski, 1999, 19.99, false
- Casablanca, Michael Curtiz, 1942, 19.95, false
```

---

Dr. Papalaskari

CSC 1051 Villanova University
Arrays, Part 2

CSC 1051 Villanova University

Dr. Papalaskari

CSC 1051 M.A. Papalaskari, Villanova University

//DVDCollection.java  Author: Lewis/Loftus/Papalaskari
//Represents a collection of DVD objects
//("** modified from textbook version **")
//-----------------------------------------------------------------------------
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;
    }
    //-----------------------------------------------------------------
    // Adds a DVD to the collection, using information from a String
    // and increasing the size of the collection array if necessary.
    //-----------------------------------------------------------------
    public void addDVD(String info)
    {
        if (count == collection.length)
            increaseSize();
        collection[count] = new DVD(info);
        count++;
    }
    //-----------------------------------------------------------------
    // 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);
        count++;
    }
    //-----------------------------------------------------------------
    // Returns a report describing the DVD collection.
    //-----------------------------------------------------------------
    public String toString()
    {
        NumberFormat fmt = NumberFormat.getCurrencyInstance();
        String report = "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
My DVD Collection

Number of DVDs: "+ count + "n"

DVD List:

for (int i = 0; i < count; i++)
    report += collection[i].toString() + "n";
return report;
    }
    //-----------------------------------------------------------------
    // 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 i = 0; i < collection.length; i++)
            temp[i] = collection[i];
        collection = temp;
    }
}
Arrays as Parameters

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

```java
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

Example: A method that adds 3 to the value of each element in an array of type int[].

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

Try this: Write a method that adds n (an int) to the value of each element in an array of type int[].

Try this method with the TrianglePanel:
- invoke it with the array xPoints
- add code to draw another triangle in a different color, using the updated array xPoints

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 an array”?

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

1) Copying elements:

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

Afterwards, what is the effect of the following?

```java
a[1] = 1000;
```

2) Copying array variables:

```java
a = b;
```

Afterwards, what is the effect of the following?

```java
a[1] = 0;
b[2] = 0;
```
Arrays, Part 2

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;
}
```