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:**

  ```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;
  ```

The entire array has a single name.

- **Instantiation:**

  - 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;
    ```

- **Array element:**

  ```java
  scores[2]
  ```

- **Size of array:**

  ```java
  scores.length
  ```

  10

CSC 1051 M.A. Papalaskari, Villanova University
Initializer Lists

• Alternative way to declare, instantiate, and initialize an array. For example:

```cpp
int[] ratings = {4, 3, 3, 1, 4, 2, 1, 0, 3, 4};

char[] grades = {'A', 'B', 'C', 'D', 'F'};
```

• **NOTE:**
  - the `new` operator is **not** used
  - size of array is determined by the number of items listed
  - can only be used in the array declaration

try this with the `vowel` array
The “for-each” Loop

• A simple way of processing every array element:

```java
for (double score : scores)
    System.out.println(score);
```

**NOTE:**

• Only appropriate when processing all array elements starting at index 0

• It can't be used to set the array values

try this with the `vowel` array

```
vowel = {'a', 'e', 'i', 'o', 'u'}
```

CSC 1051 M.A. Papalaskari, Villanova University
Another example

String[] animals = {"dog", "cat", "mouse", "fox"};

for (String word : animals)
    System.out.println("The " + word + " ate the cake");

for (String word : animals)
    for (String otherWord : animals)
        System.out.println("The " + word + " ate the " + otherWord);
Try this: Use the “for each” loop to scan through an array of `int` containing ratings (range: 0 - 4) and count up how many 4’s.

```java
int[] ratings = {4, 3, 3, 1, 4, 3, 1, 0, 3, 4};
```
Try this: Repeat, but now count up the 0’s, 1’s, … 4’s – Use a separate array for this

```c
int[] ratings = {4, 3, 3, 1, 4, 3, 1, 0, 3, 4};
```
More array examples (see textbook):

- BasicArray.java
- Primes.java
- ReverseOrder.java
- LetterCount.java
import java.util.Scanner;

public class ReverseOrder
{
    public static void main (String[] args)
    {
        Scanner scan = new Scanner (System.in);

        double[] numbers = new double[10];

        System.out.println ("The size of the array: " + numbers.length);

        continue
    }
}
continue

    for (int index = 0; index < numbers.length; index++)
    {
        System.out.print ("Enter number " + (index+1) + ": ");
        numbers[index] = scan.nextDouble();
    }

    System.out.println ("The numbers in reverse order:");

    for (int index = numbers.length-1; index >= 0; index--)
        System.out.print (numbers[index] + " ");
    }
```
for (int index = 0; index < numbers.length; index++)
{
    System.out.print("Enter number "+(index+1)+": ");
    numbers[index] = scan.nextDouble();
}
System.out.println("The numbers in reverse order:");
for (int index = numbers.length-1; index >= 0; index--)
System.out.print(numbers[index] + "  ");
```

Sample Run

The size of the array: 10
Enter number 1: 18.36
Enter number 2: 48.9
Enter number 3: 53.5
Enter number 4: 29.06
Enter number 5: 72.404
Enter number 6: 34.8
Enter number 7: 63.41
Enter number 8: 45.55
Enter number 9: 69.0
Enter number 10: 99.18
The numbers in reverse order:
99.18  69.0  45.55  63.41  34.8  72.404  29.06  53.5  48.9  18.36
```

places numbers in an array, then prints them out *backward*

… alternatively, we could place the numbers in the array *backward* and then print them *forward*
Another example: Computing letter frequency counts

Sample run:

Enter a sentence:
In Casablanca, Humphrey Bogart never says "Play it again, Sam."

<table>
<thead>
<tr>
<th>Letter</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
</tr>
</tbody>
</table>

Let's write a program to do this
import java.util.Scanner;

public class LetterCount
{
    public static void main (String[] args)
    {
        final int NUMCHARS = 26;

        Scanner scan = new Scanner (System.in);

        int[] upper = new int[NUMCHARS];
        int[] lower = new int[NUMCHARS];

        char current; // the current character being processed
        int other = 0; // counter for non-alphabetics

        continue
continue
System.out.println("Enter a sentence:");
String line = scan.nextLine();

// Count the number of each letter occurrence
for (int ch = 0; ch < line.length(); ch++)
{
    current = line.charAt(ch);
    if (current >= 'A' && current <= 'Z')
        upper[current-'A']++;
    else
    if (current >= 'a' && current <= 'z')
        lower[current-'a']++;
    else
        other++;
}

// Print the results
System.out.println();
for (int letter=0; letter < upper.length; letter++)
{
    System.out.print ( (char) (letter + 'A') );
    System.out.print (": " + upper[letter]);
    System.out.print ("		" + (char) (letter + 'a') );
    System.out.println (": " + lower[letter]);
}

System.out.println();
System.out.println("Non-alphabetic characters: " + other);
}
Sample Run

Enter a sentence:
In Casablanca, Humphrey Bogart never says "Play it again, Sam."

A: 0    a: 10
B: 1    b: 1
C: 1    c: 1
D: 0    d: 0
E: 0    e: 3
F: 0    f: 0
G: 0    g: 2
H: 1    h: 1
I: 1    i: 2
J: 0    j: 0
K: 0    k: 0
L: 0    l: 2
M: 0    m: 2
N: 0    n: 4
O: 0    o: 1
P: 1    p: 1
Q: 0    q: 0

Sample Run (continued)
R: 0    r: 3
S: 1    s: 3
T: 0    t: 2
U: 0    u: 1
V: 0    v: 1
W: 0    w: 0
X: 0    x: 0
Y: 0    y: 3
Z: 0    z: 0

Non-alphabetic characters: 14
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
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)
```
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] = 1000;
```
1) Copying elements:

<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>
<tr>
<td>b</td>
<td>100</td>
<td>200</td>
<td>300</td>
<td>400</td>
</tr>
</tbody>
</table>

What changes?

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

a[1] = 1000;
```
2) Copying array variables:

What changes?

```c
a = b;
a[1] = 1000;
```
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)
import 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
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:

//*****************************************************************************
// 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?

- Use an *array* of DVD objects:

```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);
    }
}
```
Let’s store the data in a tab-delimited file:

<table>
<thead>
<tr>
<th>Movie Title</th>
<th>Director</th>
<th>Year</th>
<th>Price</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Godfather</td>
<td>Francis Ford Coppola</td>
<td>1972</td>
<td>24.95</td>
<td>true</td>
</tr>
<tr>
<td>District 9</td>
<td>Neill Blomkamp</td>
<td>2009</td>
<td>19.95</td>
<td>false</td>
</tr>
<tr>
<td>Iron Man</td>
<td>Jon Favreau</td>
<td>2008</td>
<td>15.95</td>
<td>false</td>
</tr>
<tr>
<td>All About Eve</td>
<td>Joseph Mankiewicz</td>
<td>1950</td>
<td>17.50</td>
<td>false</td>
</tr>
<tr>
<td>The Matrix</td>
<td>Andy &amp; Lana Wachowski</td>
<td>1999</td>
<td>19.95</td>
<td>true</td>
</tr>
<tr>
<td>Iron Man 2</td>
<td>Jon Favreau</td>
<td>2010</td>
<td>22.99</td>
<td>false</td>
</tr>
<tr>
<td>Casablanca</td>
<td>Michael Curtiz</td>
<td>1942</td>
<td>19.95</td>
<td>false</td>
</tr>
</tbody>
</table>

Client code fragment to input lines from file:

```java
Scanner fileScan = new Scanner(new File(args[0]));

int i = 0;
while (fileScan.hasNextLine() && i < list.length)
{
    list[i] = new DVD(fileScan.nextLine());
    i++;
}
```

We need constructor to handle this
The current DVD constructor CANNOT be used in this manner:

```java
list[i] = new DVD(fileScan.nextLine());
```

Add a new constructor to the DVD class and try with `MyTenMovies.java`.
Next: A collection of DVD’s that can grow to accommodate as many items as needed!

• No limit like this:

        DVD[] list = new DVD[10];
Managing a collection of objects

- Example: a Movie database (collection of DVD objects)
import java.util.Scanner;
import java.io.*;

public class Movies
{
    //  Creates a DVDCollection object and adds some DVDs to it. Prints
    //  reports on the status of the collection.

    public static void main(String[] args) throws IOException
    {
        Scanner fileScan = new Scanner(new File(args[0]));
        DVDCollection movies = new DVDCollection();

        while (fileScan.hasNextLine())
        {
            movies.addDVD(fileScan.nextLine());
            System.out.println(movies);
        }
    }
}
import java.util.Scanner;
import java.io.*;
public class Movies {
    //-----------------------------------------------------------------
    //  Creates a DVDCollection object and adds some DVDs to it. Prints
    //  reports on the status of the collection.
    //-----------------------------------------------------------------
    public static void main(String[] args) throws IOException {
        Scanner fileScan = new Scanner(new File(args[0]));
        DVDCollection movies = new DVDCollection();
        while (fileScan.hasNextLine())
            movies.addDVD(fileScan.nextLine());
        System.out.println(movies);
    }
}

Output
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
My DVD Collection

Number of DVDs: 5
Total cost: $98.30

DVD List:
Sc
DV
$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
//********************************************************************
// 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;
    }

    continue
public void addDVD(String info)
{
    if (count == collection.length)
        increaseSize();

    collection[count] = new DVD(info);
    count++;
}

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
continue

// Returns a report describing the DVD collection.
public String toString()
{
    NumberFormat fmt = NumberFormat.getCurrencyInstance();
    String report = "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
My DVD Collection

Number of DVDs: " + count + "
DVD List:

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

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

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

    collection = temp;
}