Arrays and File Input

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/

Arrays - Review

Declaration:

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

Initialization:

```java
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:

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

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 start(Stage primaryStage)
{
    double[] trianglePoints = {100, 150, 120, 40, 150, 110};
    Polygon triangle = new Polygon(trianglePoints);
    triangle.setFill(Color.RED);

    double[] pentagonPoints = {35, 70, 35, 40, 60, 20, 80, 40, 80, 60};
    Polygon pentagon = new Polygon(pentagonPoints);
    pentagon.setFill(Color.MAROON);

    Polyline vee = new Polyline(trianglePoints); // note: using same array
    vee.setStroke(Color.GREEN);
    vee.setStrokeWidth(3);

    Group root = new Group(triangle, pentagon, vee);
    Scene scene = new Scene(root, 200, 200, Color.BLACK);
    primaryStage.setTitle("Shapes");
    primaryStage.setScene(scene);
    primaryStage.show();
}
```

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

```java
public void addTen(double[] a)
{
    for (int i = 0; i < a.length; i++)
        a[i] += 10;
}
```
Write a method `addX` that adds `x` (a `double`) to the value of each element in an array of type `double[]`.

Try this method with `Shapes.java`:
- add code to draw a third, blue triangle *shifted by some amount n*

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

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};
  ```

```java
Copying elements vs. copying array variables:
for (int i=0; i<a.length; i++)
a[i] = b[i];
```

Afterwards, what is the effect of the following?
- ```
a[1] = 0;
b[2] = 0;
```
2) Copying array variables:

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
147 & 323 & 89 & 933 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
100 & 200 & 300 & 400 \\
\end{array}
\]

\[
a = b; \\
a[1] = 0; \\
b[2] = 0;
\]

Trace this code. What changes in the arrays?

Arrays as Parameters - revisited

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

```java
public void start(Stage primaryStage)
{
    double[] trianglePoints = {100, 150, 120, 40, 150, 110};
    Polygon triangle = new Polygon(trianglePoints);
    triangle.setFill(Color.RED);
    double[] pentagonPoints = {35, 70, 35, 40, 60, 20, 80, 40, 80, 60};
    Polygon pentagon = new Polygon(pentagonPoints);
    pentagon.setFill(Color.MAROON);

    Polyline vee = new Polyline(trianglePoints); // note: using same array
    vee.setStroke(Color.GREEN);
    vee.setStrokeWidth(3);

    Group root = new Group(triangle, pentagon, vee);
    Scene scene = new Scene(root, 200, 200, Color.BLACK);
    primaryStage.setTitle("Triangles and Pentagons");
    primaryStage.setScene(scene);
    primaryStage.show();
}
```

Managing a collection of objects

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

```java
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;
    }
}
```
Arrays & File Input

CSC 1051 Villanova University

Test client – create a few DVDs, print their info:

```java
public class TestDVD {
    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);
    }
}
```

Managing a collection of objects

- Example: a Movie database (collection 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);
        }
    }
}
```
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)
    {
        DVDCollection movies = new DVDCollection();
        movies.addDVD(“The Godfather”, “Francis Ford Coppala”, 1972, 24.95, true);
        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);
        movies.addDVD(“Casablanca”, “Michael Curtiz”, 1942, 19.95, false);
        System.out.println(movies);
        System.out.println(“My DVD Collection”);
        System.out.println(“DVD List:”);
        System.out.println(“Total cost: $98.30”);
        System.out.println(“Number of DVDs: 5”);
        System.out.println(“Average cost: $19.66”);
    }
}

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 Blokamp
$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 Blokamp
$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);
}
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++;
}

public String toString() {
    NumberFormat fmt = NumberFormat.getCurrencyInstance();
    String report = "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    My DVD Collection\n    Number of DVDs: "+ count + "\n    DVD List:\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.
Arrays & File Input

Dr. Papalaskari

CSC 1051 Villanova University

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>

- 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).
Arrays & File Input  

CSC 1051 Villanova University

Dr. Papalaskari

PEC 1051 M.A. Papalaskari, Villanova University

Copyright © 2014 Pearson Education, Inc.

//********************************************************************
// SodaSurvey.java       Author: Lewis/Loftus
//
// Demonstrates the use of a two-dimensional array.
//********************************************************************
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
        for (int soda=0; soda < SODAS; soda++)
            for (int person=0; person < PEOPLE; person++)
                sodaSum[soda] += scores[soda][person];
        for (int person=0; person < PEOPLE; 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));
        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

Iterators
• **Iterating**: to process a collection of items, one at a time
• Typical iterator methods:
  - `next()` returns the next item
  - `hasNext()` - returns true if there is at least one more item to process
• Several classes in the Java standard class library are iterators

Tokens
• items to be processed are called **tokens**
  – Examples: words, numbers, components of a url...
• The **Scanner** class is an iterator
  – `next()` returns the next scanned token (a String)
  – `nextLine()` returns the rest of the line (until the next new line).
  – `hasNext()` returns true if there is more data to be scanned
• Variations of `next()` and `hasNext()` methods:
  - `nextInt()` `hasNextInt()`
  - `nextDouble()` `hasNextDouble()`
Using Scanner to read from a file
• Create a File object:
  File myFile = new File("sample.txt");
• Create a Scanner to read from the File object:
  Scanner fileScan = new Scanner (myFile);
• Use next() to obtain next token
• Use nextLine() to obtain entire line of text (until \n)
• Use hasNext() to test whether you are done

File Input Example: FileInput.java
import java.util.Scanner;
import java.io.*;
public class FileInput
{
  // Reads text from a file and prints it in uppercase.
  public static void main (String[] args) throws IOException
  {
    String line;
    File myFile = new File("sample.txt");
    Scanner fileScan = new Scanner (myFile);
    // Read and process each line of the file
    while (fileScan.hasNext())
    {
      line = fileScan.nextLine();
      System.out.println(line.toUpperCase());
    }
  
  }
}

Try this: What gets printed?
//****************************************************************
// SomethingToDoWithFiles.java Author: MAP
//****************************************************************
import java.util.Scanner;
import java.io.*;
public class SomethingToDoWithFiles
{
  public static void main (String[] args) throws IOException
  {
    String line1, line2;
    Scanner fileScan1, fileScan2;
    fileScan1 = new Scanner (new File("sample1.txt"));
    fileScan2 = new Scanner (new File("sample2.txt"));
    while (fileScan1.hasNext() && fileScan2.hasNext())
    {
      line1 = fileScan1.nextLine();
      line2 = fileScan2.nextLine();
      System.out.println(line1 + line2 + line1);
    }
    System.out.println(fileScan1.hasNext()? "ping!": "pong!");
  
  }
}
Scanner – another example: reading from a file AND from a String

- Suppose we wanted to read and process a list of URLs (or other data items) stored in a file
- One scanner can be set up to read each line of the input until the end of the file is encountered
- Another scanner can be set up to process each line, i.e., separating the components of each URL (at each occurrence of '/')
- Example: URL: www.linux.org/info/gnu.html
  This URL specifies a path consisting of the following components:
  - www.linux.org
  - info
  - gnu.html
- See URLDissector.java

```java
import java.util.Scanner;
import java.io.*;
public class URLDissector {
    public static void main (String[] args) throws IOException {
        String url;
        Scanner fileScan, urlScan;
        fileScan = new Scanner (new File("urls.txt"));
        continue
        // Read and process each line of the file
        while (fileScan.hasNext()) {
            url = fileScan.nextLine();
            System.out.println("URL: " + url);
            urlScan = new Scanner (url);
            urlScan.useDelimiter("/");
            // Print each part of the url
            while (urlScan.hasNext()) {
                System.out.println("   " + urlScan.next());
                System.out.println();
            }
        }
    }
}
```

Sample Run

URL: www.google.com
www.google.com
URL: www.linux.org/info/gnu.html
www.linux.org
info
www.linux.org
jurk.html
URL: thelyric.com/calendar/thelyric.com/calender
URL: www.cs.vt.edu/undergraduate/about
www.cs.vt.edu
theblues/about
URL: youtube.com/watch?v=EHCRimwRGLs
youtube.com
watch?v=EHCRimwRGLs