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/

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];
```

**Instantiation:**

```
size of array
```

**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 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();
}
```
Arrays as Parameters

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 \texttt{addX} that adds \texttt{x} (a \texttt{double}) to the value of each element in an array of type \texttt{double[]}.

Try this method with \texttt{Shapes.java}:

- add code to draw a third, \textcolor{blue}{blue} triangle \textit{shifted by some amount} \texttt{n}
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};
  ```

  **Copying elements vs. copying array variables:**

  ```java
  for (int i = 0; i < a.length; i++)
    a[i] = b[i];
  
  a[1] = 0;
  b[2] = 0;
  ```

Afterwards, what is the effect of the following?
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:

\[
a = b;
\]

\[
a[1] = 0;
\]

\[
b[2] = 0;
\]

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
// 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("Triangles and Pentagons");
    primaryStage.setScene(scene);
    primaryStage.show();
}

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

    // Creates and prints DVD info
    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);
    }
}
Managing a collection of objects

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

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
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);
    }
}
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
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);
    count++;
}
continue
public String toString()
{
    NumberFormat fmt = NumberFormat.getCurrencyInstance();

    String report = "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    My DVD Collection
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"
    report += "Number of DVDs: " + count + "\n"
    report += "\n\nDVD 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
double[][] courseGrade = new double[3][10];
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>

- 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**)

CSC 1051 M.A. Papalaskari, Villanova University
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];
    }
For another 2D array example from the textbook, consider the following code snippet:

```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));

System.out.println ("\n");
for (int person=0; person < PEOPLE; person++)
    System.out.println ("Person #" + (person+1) + " : " + fmt.format ((float)personSum[person]/SODAS));
```
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 \textit{tokens}
  – Examples: words, numbers, components of a url…

• The \texttt{Scanner} class is an iterator
  – \texttt{next()} returns the next scanned token (a String)
  – \texttt{nextLine()} returns the rest of the line (until the next new line).
  – \texttt{hasNext()} returns true if there is more data to be scanned

• Variations of \texttt{next()} and \texttt{hasNext()} methods:
  \begin{verbatim}
  nextInt()  hasNextInt()
  nextDouble()  hasNextDouble()
  \end{verbatim}
Using Scanner to read from a file

• Create a File object:
  ```java
  File myFile = new File("sample.txt");
  ```

• Create a Scanner to read from the File object:
  ```java
  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`

```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());
        }
    }
}
```
import java.util.Scanner;
import java.io.*;

public class FileInput
{
    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());
        }
    }
}

CSC 2014 M.A. Papalaskari, Villanova University
Try this: What gets printed?

```java
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!");
    }
}
```

CSC 1051 M.A. Papalaskari, Villanova University
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
import java.util.Scanner;
import java.io.*;

public class URLDissector
{
    //---------------------------------------------
    // Reads urls from a file and prints their path components.
    //---------------------------------------------
    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();
}
}
// 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
   gnu.html

URL: thelyric.com/calendar/
   thelyric.com
   calendar

URL: www.cs.vt.edu/undergraduate/about
   www.cs.vt.edu
   undergraduate
   about

URL: youtube.com/watch?v=EHCRimwRGLs
   youtube.com
   watch?v=EHCRimwRGLs