10/26/17
review test
arrays
note: homework # 5 due tuesday

Arrays

Chapter 7

5TH EDITION

Lewis & Loftus

java

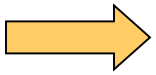
Software Solutions

Foundations of Program Design

Arrays

- **Arrays are objects that help us organize large amounts of information**
- **Chapter 7 focuses on:**
  - **array declaration and use**
  - **bounds checking and capacity**
  - **arrays that store object references**
  - **variable length parameter lists**
  - **multidimensional arrays**
  - **the `ArrayList` class**
  - **polygons and polylines**
  - **mouse events and keyboard events**

Outline



Declaring and Using Arrays

Arrays of Objects

Variable Length Parameter Lists

Two-Dimensional Arrays

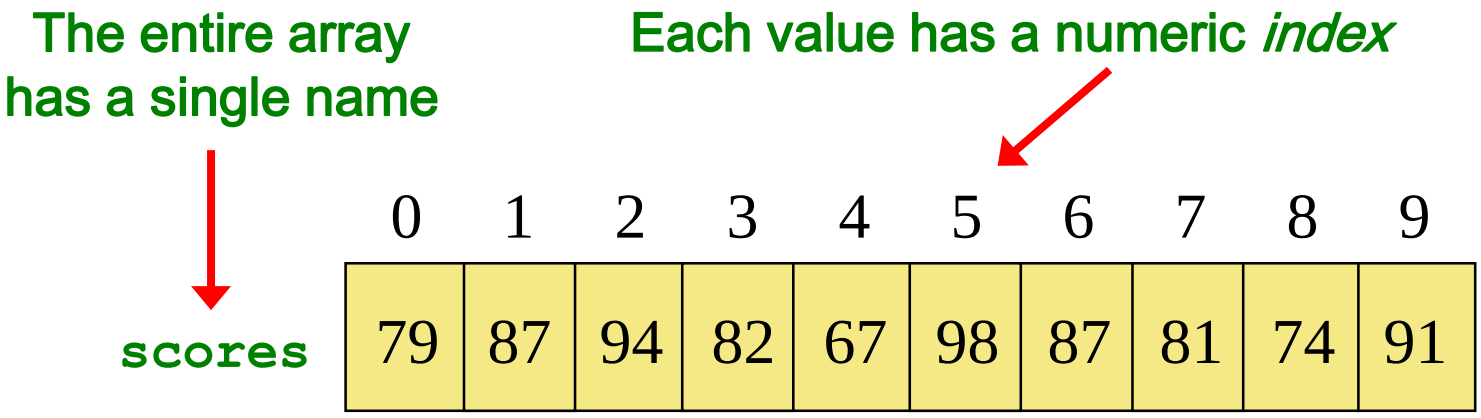
The `ArrayList` Class

Polygons and Polylines

Mouse Events and Key Events

Arrays

- **An *array* is an ordered list of values**



An array of size N is indexed from zero to N-1

This array holds 10 values that are indexed from 0 to 9

Arrays

- **A particular value in an array is referenced using the array name followed by the index in brackets**
- **For example, the expression**

```
scores[2]
```

refers to the value `94` (the 3rd value in the array)

- **That expression represents a place to store a single integer and can be used wherever an integer variable can be used**

Arrays

10/31/17
-more arrays
-assignment #6
-in class

For example, an array element can be assigned a value, printed, or used in a calculation:

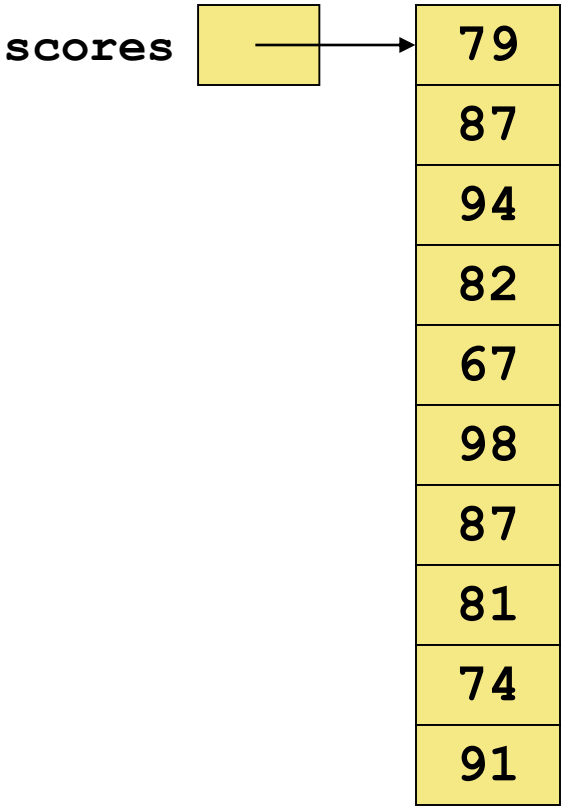
```
scores[2] = 89;

scores[first] = scores[first] + 2;

mean = (scores[0] + scores[1])/2;

System.out.println ("Top = " + scores[5]);
```

Arrays

- **The values held in an array are called *array elements***
- **An array stores multiple values of the same type – the *element type*** /base
- **The element type can be a primitive type or an object reference**
- **Therefore, we can create an array of integers, an array of characters, an array of `String` objects, an array of `Coin` objects, etc.**
- **In Java, the array itself is an object that must be instantiated**

Arrays

- **Another way to depict the `scores` array:**



Declaring Arrays

- **The `scores` array could be declared as follows:**

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

this means array of int

- **The type of the variable `scores` is `int[]` (an array of integers)**
- **Note that the array type does not specify its size, but each object of that type has a specific size**
- **The reference variable `scores` is set to a new array object that can hold 10 integers**

array has method length()

Declaring Arrays

- **Some other examples of array declarations:**

```
float[] prices = new float[500];

boolean[] flags;
flags = new boolean[20];

char[] codes = new char[1750];
```

Using Arrays

- **The iterator version of the `for` loop can be used when processing array elements**

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

this is like 'for each'

this is only a read method - it ccannot update?

- **This is only appropriate when processing all array elements from top (lowest index) to bottom (highest index)**
- **See `BasicArray.java` (page 374)**

Bounds Checking

if you try to store information in memory that was not allocated there will be an array out of bounds exception = this is a runtime error. java does not allow you to be out of bounds.

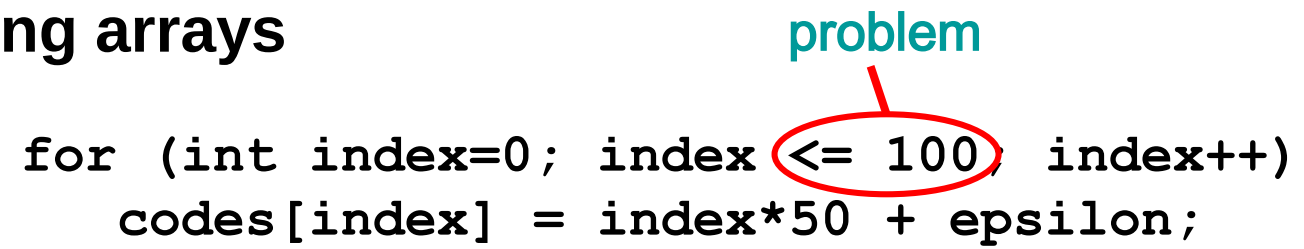
- **Once an array is created, it has a fixed size**
- **An index used in an array reference must specify a valid element**
- **That is, the index value must be in range 0 to N-1**
- **The Java interpreter throws an `ArrayIndexOutOfBoundsException` if an array index is out of bounds**
- **This is called automatic *bounds checking***

Bounds Checking

- For example, if the array `codes` can hold 100 values, it can be indexed using only the numbers 0 to 99
- If the value of `count` is 100, then the following reference will cause an exception to be thrown:

```
System.out.println (codes[count]);
```

- It's common to introduce *off-by-one errors* when using arrays



Bounds Checking

- **Each array object has a public constant called `length` that stores the size of the array**
- **It is referenced using the array name:**

```
scores.length
```

use this so you don't go out of bounds

- **Note that `length` holds the number of elements, not the largest index**
- **See `ReverseOrder.java` (page 377)**
- **See `LetterCount.java` (page 378)**

10/26/17 ended here

Alternate Array Syntax

- **The brackets of the array type can be associated with the element type or with the name of the array**
- **Therefore the following two declarations are equivalent:**

```
float[] prices;
```
this is java style

```
float prices[];
```
this is c++ style

- **The first format generally is more readable and should be used**

Initializer Lists

- **An *initializer list* can be used to instantiate and fill an array in one step**
- **The values are delimited by braces and separated by commas**
- **Examples:**

```
int[] units = {147, 323, 89, 933, 540,
               269, 97, 114, 298, 476};


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

you don't have to state a size. the size is set for you
you can only use {} when you first create the array

all methods are able to access the instance variables defined at the top of the class, so no real need to pass those as parameters

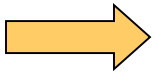
Initializer Lists

- **Note that when an initializer list is used:**
  - **the `new` operator is not used**
  - **no size value is specified**
- **The size of the array is determined by the number of items in the initializer list**
- **An initializer list can be used only in the array declaration**
- **See `Primes.java` (page 383)**

Arrays as Parameters

- **An entire array can be passed as a parameter to a method**
- **Like any other object, the reference to the array is passed, making the formal and actual parameters aliases of each other**
- **Therefore, changing an array element within the method changes the original**
- **An individual array element can be passed to a method as well, in which case the type of the formal parameter is the same as the element type**

Outline



Declaring and Using Arrays

Arrays of Objects

Variable Length Parameter Lists

Two-Dimensional Arrays

The `ArrayList` Class

Polygons and Polylines

Mouse Events and Key Events

Arrays of Objects

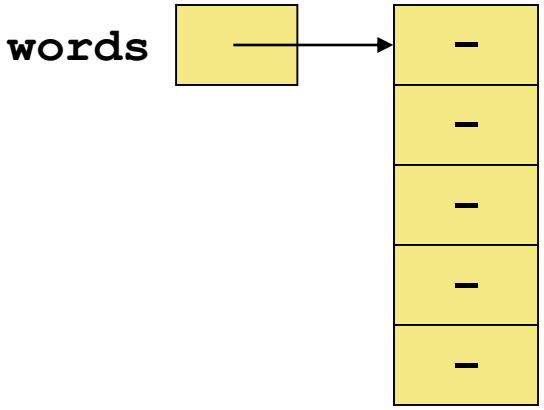
- **The elements of an array can be object references**
- **The following declaration reserves space to store 5 references to `String` objects**

```
String[] words = new String[5];
```

- **It does NOT create the `String` objects themselves**
- **Initially an array of objects holds `null` references**
- **Each object stored in an array must be instantiated separately**

Arrays of Objects

- **The `words` array when initially declared:**

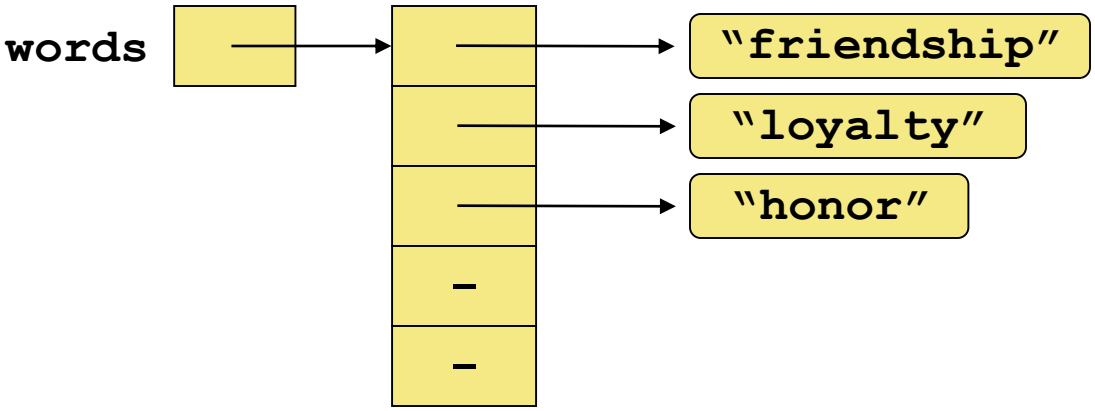


- **At this point, the following reference would throw a `NullPointerException`:**

```
System.out.println (words[0]);
```

Arrays of Objects

- **After some `String` objects are created and stored in the array:**



Arrays of Objects

- **Keep in mind that `String` objects can be created using literals**
- **The following declaration creates an array object called `verbs` and fills it with four `String` objects created using string literals**

```
String[] verbs = {"play", "work", "eat", "sleep"};
```

Arrays of Objects

- **The following example creates an array of `Grade` objects, each with a string representation and a numeric lower bound**
- **See `GradeRange.java` (page 386)**
- **See `Grade.java` (page 387)**
- **Now let's look at an example that manages a collection of `CD` objects**
- **See `Tunes.java` (page 389)**
- **See `CDCollection.java` (page 390)**
- **See `CD.java` (page 393)**

```
private void increaseSize ()
{
  CD[] temp = new CD[collection.length * 2];
  for (int cd = 0; cd < collection.length; cd++)
    temp[cd] = collection[cd];
collection = temp;
}
```

in this code, temp[1] points to collection[1]
then, sets collection points to temp
the original collection will get garbage collected because nothing is pointing to it anymore

looked at this example

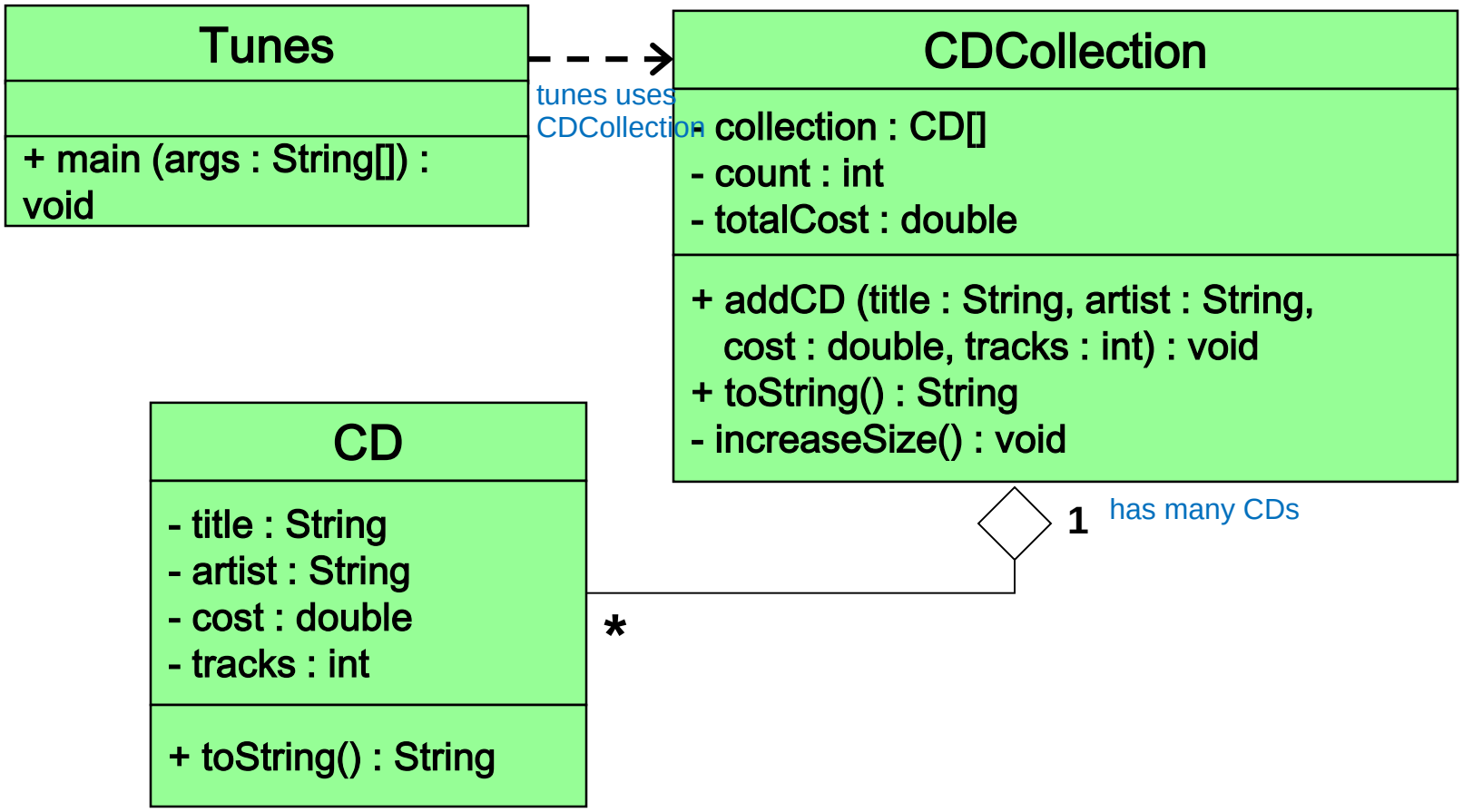
collection: array of CDs
colelction[1]: CD

limitation for arrays: if the space is filled, if it was 100 spaces, it can't be updated.

so don't initialize it unless you know how much space you need

Arrays of Objects

- **A UML diagram for the `Tunes` program:**



Command-Line Arguments

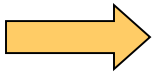
- The signature of the `main` method indicates that it takes an array of `String` objects as a parameter
- These values come from *command-line arguments* that are provided when the interpreter is invoked
- For example, the following invocation of the interpreter passes three `String` objects into `main`:

```
> java StateEval pennsylvania texas arizona
```

- These strings are stored at indexes 0-2 of the array parameter of the `main` method
- See `NameTag.java` (page 395)

Outline

not going to be tested on command line arguments,
variable length parameter, polygons and mouse events



Declaring and Using Arrays

Arrays of Objects

Variable Length Parameter Lists

Two-Dimensional Arrays

The `ArrayList` Class

Polygons and Polylines

Mouse Events and Key Events

Variable Length Parameter Lists

- **Suppose we wanted to create a method that processed a different amount of data from one invocation to the next**
- **For example, let's define a method called `average` that returns the average of a set of integer parameters**

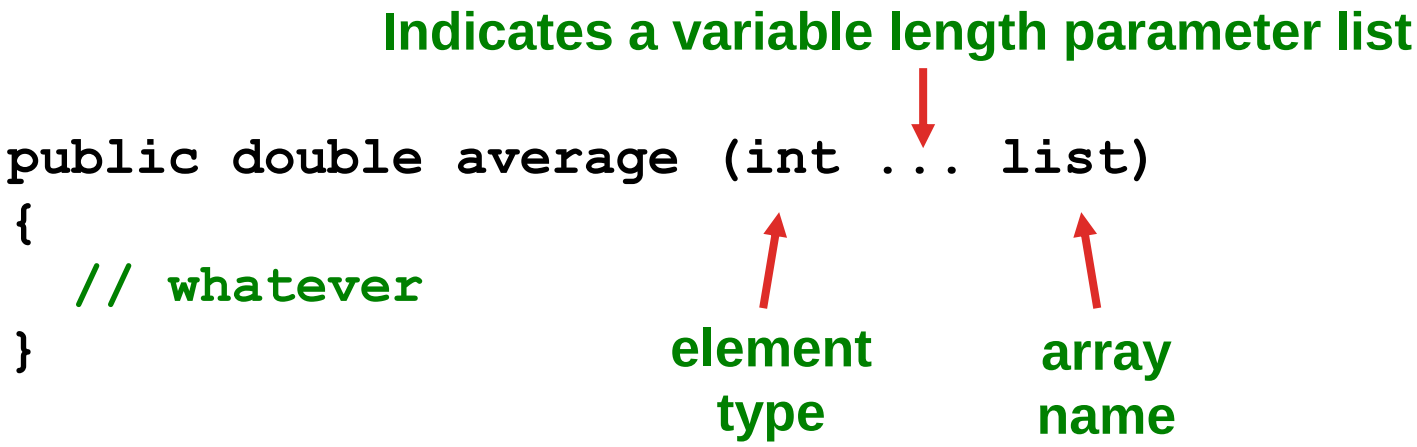
```
// one call to average three values
mean1 = average (42, 69, 37);

// another call to average seven values
mean2 = average (35, 43, 93, 23, 40, 21, 75);
```

Variable Length Parameter Lists

- **We could define overloaded versions of the `average` method**
  - **Downside: we'd need a separate version of the method for each parameter count**
- **We could define the method to accept an array of integers**
  - **Downside: we'd have to create the array and store the integers prior to calling the method each time**
- **Instead, Java provides a convenient way to create *variable length parameter lists***

Variable Length Parameter Lists

- **Using special syntax in the formal parameter list, we can define a method to accept any number of parameters of the same type**
- **For each call, the parameters are automatically put into an array for easy processing in the method**



Variable Length Parameter Lists

```
public double average (int ... list)
{
   double result = 0.0;

   if (list.length != 0)
   {
      int sum = 0;
      for (int num : list)
         sum += num;
      result = (double)num / list.length;
   }

   return result;
}
```

Variable Length Parameter Lists

- **The type of the parameter can be any primitive or object type**

```
public void printGrades (Grade ... grades)
{
   for (Grade letterGrade : grades)
      System.out.println (letterGrade);
}
```

Variable Length Parameter Lists

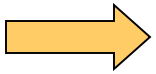
- **A method that accepts a variable number of parameters can also accept other parameters**
- **The following method accepts an `int`, a `String` object, and a variable number of `double` values into an array called `nums`**

```
public void test (int count, String name,
                  double ... nums)
{
   // whatever
}
```

Variable Length Parameter Lists

- **The varying number of parameters must come last in the formal arguments**
- **A single method cannot accept two sets of varying parameters**
- **Constructors can also be set up to accept a variable number of parameters**
- **See `VariableParameters.java` (page 398)**
- **See `Family.java` (page 399)**

Outline



Declaring and Using Arrays

Arrays of Objects

Variable Length Parameter Lists

Two-Dimensional Arrays

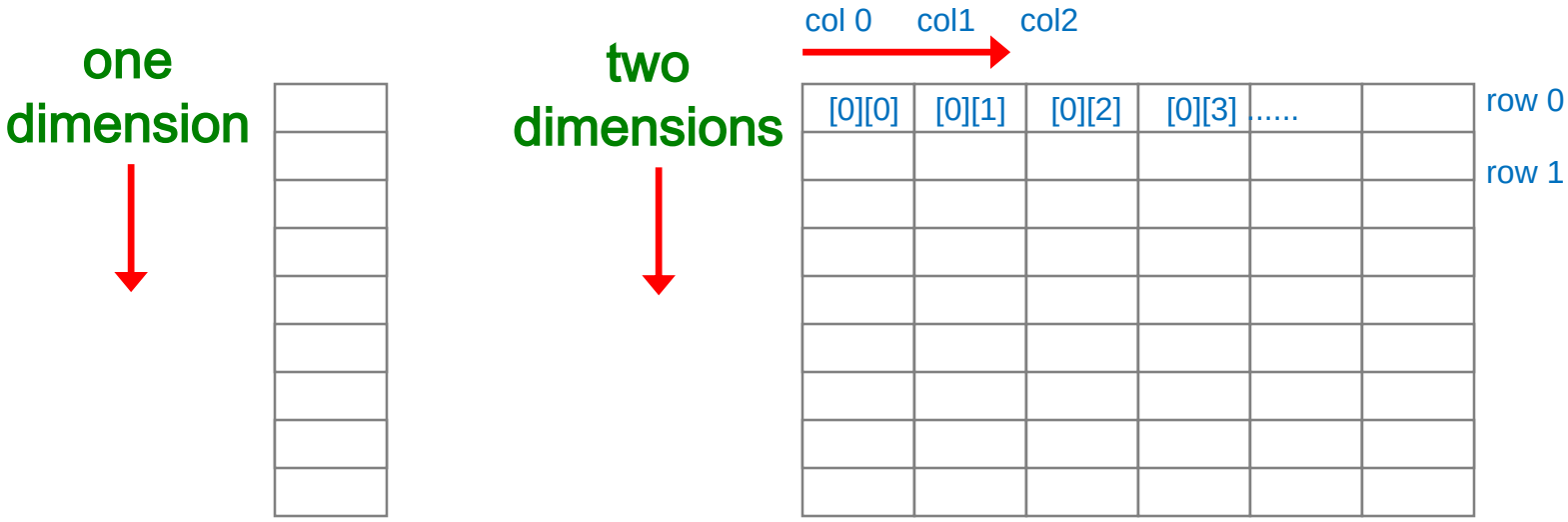
The `ArrayList` Class

Polygons and Polylines

Mouse Events and Key Events

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

- **To be precise, in Java a two-dimensional array is an array of arrays**
- **A two-dimensional array is declared by specifying the size of each dimension separately:**

row column?

```
int[][] scores = new int[12][50];
```

- **A array element is referenced using two index values:**

```
value = scores[3][6]
```

- **The array stored in one row can be specified using one index**

Two-Dimensional Arrays

Expression	Type	Description
`table`	`int[][]`	2D array of integers, or array of integer arrays
`table[5]`	`int[]`	array of integers
`table[5][12]`	`int`	integer

row column

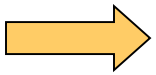
- See `TwoDArray.java` (page 401)
- See `SodaSurvey.java` (page 402)

	col 0	col 1									
row 0	[0][0]	[0][1]	[0][2]	[0][3]	[0][4]	[0][5]	[0][6]	[0][7]	[0][8]	[0][9]	[0][10]	[0][11]
row 1	[1][0]	[1][1]	[1][2]	[1][3]	[1][4]	[1][5]	[1][6]	[1][7]	[1][8]	[1][9]	[1][10]	[1][11]
row 3	[2][0]	[2][1]	[2][2]	[2][3]	[2][4]	[2][5]	[2][6]	[2][7]	[2][8]	[2][9]	[2][10]	[2][11]
	[3][0]	[3][1]	[3][2]	[3][3]	[3][4]	[3][5]	[3][6]	[3][7]	[3][8]	[3][9]	[3][10]	[3][11]
	[4][0]	[4][1]	[4][2]	[4][3]	[4][4]	[4][5]	[4][6]	[4][7]	[4][8]	[4][9]	[4][10]	[4][11]

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***

Outline



Declaring and Using Arrays

Arrays of Objects

Variable Length Parameter Lists

Two-Dimensional Arrays

The `ArrayList` Class

Polygons and Polylines

Mouse Events and Key Events

The ArrayList Class

- **The `ArrayList` class is part of the `java.util` package**
- **Like an array, it can store a list of values and reference each one using a numeric index**
- **However, you cannot use the bracket syntax with an `ArrayList` object**
- **Furthermore, an `ArrayList` object grows and shrinks as needed, adjusting its capacity as necessary**

The ArrayList Class

- **Elements can be inserted or removed with a single method invocation**
- **When an element is inserted, the other elements "move aside" to make room**
- **Likewise, when an element is removed, the list "collapses" to close the gap**
- **The indexes of the elements adjust accordingly**

The ArrayList Class

- An `ArrayList` stores references to the `Object` class, which allows it to store any kind of object
- See `Beatles.java` (page 407)
- We can also define an `ArrayList` object to accept a particular type of object
- The following declaration creates an `ArrayList` object that only stores `Family` objects

```
ArrayList<Family> reunion = new ArrayList<Family>
```

- This is an example of *generics*, which are discussed further in Chapter 12

ArrayList Efficiency

- **The `ArrayList` class is implemented using an underlying array**
- **The array is manipulated so that indexes remain continuous as elements are added or removed**
- **If elements are added to and removed from the end of the list, this processing is fairly efficient**
- **But as elements are inserted and removed from the front or middle of the list, the remaining elements are shifted**

Outline

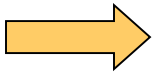
Declaring and Using Arrays

Arrays of Objects

Variable Length Parameter Lists

Two-Dimensional Arrays

The `ArrayList` Class



Polygons and Polylines

Mouse Events and Key Events

Polygons and Polylines

- **Arrays can be helpful in graphics processing**
- **For example, they can be used to store a list of coordinates**
- **A *polygon* is a multisided, closed shape**
- **A *polyline* is similar to a polygon except that its endpoints do not meet, and it cannot be filled**
- **See `Rocket.java` (page 411)**
- **See `RocketPanel.java` (page 412)**

The Polygon Class

- The `Polygon` class can also be used to define and draw a polygon
- It is part of the `java.awt` pacakage
- Versions of the overloaded `drawPolygon` and `fillPolygon` methods take a single `Polygon` object as a parameter instead of arrays of coordinates
- A `Polygon` object encapsulates the coordinates of the polygon

Outline

Declaring and Using Arrays

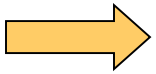
Arrays of Objects

Variable Length Parameter Lists

Two-Dimensional Arrays

The `ArrayList` Class

Polygons and Polylines



Mouse Events and Key Events

Mouse Events

- **Events related to the mouse are separated into *mouse events* and *mouse motion events***
- **Mouse Events:**

mouse pressed	**the mouse button is pressed down**
mouse released	**the mouse button is released**
mouse clicked	**the mouse button is pressed down and released without moving the mouse in between**
mouse entered	**the mouse pointer is moved onto (over) a component**
mouse exited	**the mouse pointer is moved off of a component**

Mouse Events

- **Mouse Motion Events:**

mouse moved	**the mouse is moved**
mouse dragged	**the mouse is moved while the mouse button is pressed down**

- **Listeners for mouse events are created using the `MouseListener` and `MouseMotionListener` interfaces**
- **A `MouseEvent` object is passed to the appropriate method when a mouse event occurs**

Mouse Events

- **For a given program, we may only care about one or two mouse events**
- **To satisfy the implementation of a listener interface, empty methods must be provided for unused events**
- **See `Dots.java` (page 415)**
- **See `DotsPanel.java` (page 416)**

Mouse Events

- ***Rubberbanding*** **is the visual effect in which a shape is "stretched" as it is drawn using the mouse**
- **The following example continually redraws a line as the mouse is dragged**
- **See `RubberLines.java` (page 419)**
- **See `RubberLinesPanel.java` (page 420)**

Key Events

- **A *key event* is generated when the user types on the keyboard**

key pressed	**a key on the keyboard is pressed down**
key released	**a key on the keyboard is released**
key typed	**a key on the keyboard is pressed down and released**

- **Listeners for key events are created by implementing the `KeyListener` interface**
- **A `KeyEvent` object is passed to the appropriate method when a key event occurs**

Key Events

- **The component that generates a key event is the one that has the current *keyboard focus***
- **Constants in the `KeyEvent` class can be used to determine which key was pressed**
- **The following example "moves" an image of an arrow as the user types the keyboard arrow keys**
- **See `Direction.java` (page 423)**
- **See `DirectionPanel.java` (page 424)**

Summary

- **Chapter 7 has focused on:**
  - **array declaration and use**
  - **bounds checking and capacity**
  - **arrays that store object references**
  - **variable length parameter lists**
  - **multidimensional arrays**
  - **the `ArrayList` class**
  - **polygons and polylines**
  - **mouse events and keyboard events**

what will be on the test:
no mouse activity
covered all the graphics that will cover in this class