Java: An Eventful Approach

Chapter 14

Numbering Is Helpful

the pages of a book
(chapters and sections too)
days of a month
years

small collections-often use distinct names
days of the week are Mon, Tues, etc.
large collections- numbering is handy
pages of a book

Arrays

A collection of primitive values or objects

• Useful in programs that manipulate relatively large collections of similar data
• Number items in the data collection instead of using distinct names
• Each item in the collection is associated with a number called its index

Declaring Array Names

• The collection needs a name
  Ex: say that page is the name of an array and its items are Strings corresponding to pages of a book.
  We say page[12] to access 12th element of the collection
• To declare page as the name of the collection
  private String[ ] page

Choosing an Array Name

• Why not use pages as the name of the collection of pages?
  page[12] makes more sense than pages[12]

Types of Array Elements

• Array elements may be any type
  private FilledOval[ ] piece;
  (the ovals rep pieces on a checkerboard)
• But all elements of a given array must have same type.

  No mixing of types!
Creating an Array

- Declaration of a collection’s name does not create the collection or the items in it
  
ex: private FilledOval[ ] piece;
- Need to
  - construct array
  - construct individual Filled Ovals

Constructing an Array

- Need to provide
  - Types of items
  - Total size
  
  piece= new FilledOval[24];

Constructs the collection - not the individual elements

Array Constructions in Declarations

- Common to use array constructions as initial values in array name declarations
  
  Ex 1: our array of checkers
  
  private FilledOval[ ] piece=new FilledOval[24];

  Ex 2: Array of Strings to hold text of pages of a book
  
  private String[ ] page = new String[723];
  assuming the book has 723 pages.

Array Elements

After
  
  private FilledOval[ ] piece=new FilledOval[24];

we have an array but no FilledOvals.

piece[3].setColor(Color.RED);

will result in a null pointer exception

Use an assignment statement to associate members of an array with index values

piece[3]=new FilledOval(checkerLeft, checkerTop, SIZE, SIZE, canvas);

Indexed Variables

- An array name followed by an index value is called an indexed variable
- Can be used in any context where a variable of the array’s element type can be used
- Java arrays use 0 as the first index value
- Last usable index is 1 less than size of array.
  
  piece[3] refers to fourth array element

Array Initializers

These combine creation of an array and association of values with its elements into a single step

- List values to be associated with array’s elements
- Separate values by commas
- Surround with curly braces

private int[ ] monthLength = {31, 28, 30, 31, 30, 31, 31, 30, 31, 30, 31, 31};

Using Arrays: A Triangle Class

Say you want to define a Triangle class—need 3 instance variables to refer to the three lines that make up a triangle or can use an array:

```java
private Line[] edge = new Line[3];
```

Initially, `edge[0]`, `edge[1]`, `edge[2]` all have `null` as their values.

A Simple Triangle Class

- Simplest way to describe a triangle is to provide coordinates of vertices
- Define constructor to expect 4 parameters: 3 Locations and the canvas:

```java
class Triangle{
    private Line[] edge = new Line[3];

    public Triangle(Location vert1, Location vert2, Location vert3, DrawingCanvas canvas){
        edge[0] = new Line(vert1, vert2, canvas);
        edge[1] = new Line(vert2, vert3, canvas);
        edge[2] = new Line(vert3, vert1, canvas);
    }
}
```

Additional Triangle Methods

- Might want to include implementations of methods like `move`, `setColor`, or `hide`
- Could write:

```java
public void hide(){
    edge[0].hide();
    edge[1].hide();
    edge[2].hide();
}
```

- Or even better:

```java
public void hide(){
    for(int edgeNum = 0; edgeNum < edge.length; edgeNum++ ){
        edge[edgeNum].hide();
    }
}
```

Array-processing Loops

General Form:

```java
for(int elementPosition = 0; elementPosition < array.length; elementPosition++ ){
    if perform desired operation on array[elementPosition] 
    ... 
}
```

Why loop?
- Flexibility—triangles, hexagons, etc can all be handled the same way
- Short, simple, and descriptive

If Triangle constructor invoked with Locations (100, 50), (50, 150), (250, 50):

Note that elements of the edge array refer to components of a Triangle.

Or even better:

```java
public void hide(){
    for( int edgeNum = 0; edgeNum < edge.length; edgeNum++ ){
        edge[edgeNum].hide();
    }
}
```

- The desired element of an array can be specified by a variable or any other expression that describes an int
- Name of an array variable followed by `.length` produces the number of elements in the array.
Additional examples

```java
public void move(double dx, double dy) {
    for (int edgeNum = 0; edgeNum < edge.length; edgeNum++) {
        edge[edgeNum].move(dx, dy);
    }
}

public void show() {
    for (int edgeNum = 0; edgeNum < edge.length; edgeNum++) {
        edge[edgeNum].show();
    }
}
```

Arrays are Objects

- can pass entire arrays as parameters
- can write methods that return arrays

```java
public Location[] getVertices() {
    Location[] result = new Location[edge.length];
    for (int edgeNum = 0; edgeNum < edge.length; edgeNum++) {
        result[edgeNum] = edge[edgeNum].getStart();
    }
    return result;
}
```

Enhanced for loop (Java 1.5)

Makes it easier to iterate through arrays

```java
public void hide() {
    for (Line nextLine: edge) {
        nextLine.hide();
    }
}
```

Gathering Information

- Often useful to gather info about a collection rather than process its elements independently.
- Ex 1. Determining the perimeter of a Triangle
- Ex 2. Computing traffic statistics

Computing stats in a traffic radar trailer

- Say we want to determine the number of speeders passing the trailer during each of 24 hrs.
- 24 numbers to count speeders can be kept in array

```java
private int[] speedersAt = new int[24];
```
- `speedersAt[hour]` accesses number of speeders at “hour” (using 24-hr clock)
Program Organization

- **RadarController**
  - acts as "controller"
  - event-handling method to be invoked when vehicle detected
- **RadarStats**
  - responsible for recording stats
  - update `speedersAt` when speeder detected
  - provide access to collected statistics

Counting Speeders

- Method in **RadarStats** class
- Invoked by **RadarController** when vehicle detected

```java
public void vehicleReport(double speed, int hour, int minute) {
    if (speed > speedLimit) {
        speedersAt[hour]++;
    }
}
```

Remember that hour is based on a 24-hr clock

Summing Values in an Array

```java
private int speedersSeen() {
    int total = 0;
    for (int hour = 0; hour < speedersAt.length; hour++) {
        total += speedersAt[hour];
    }
    return total;
}
```

Note the use of `total` to accumulate the sum

Reporting Stats

- number of speeders detected at different times of the day
- percent speeders in each hour

A Simpler Version

A Simple Histogram

- Loop is similar to loop of Triangle
- Operation: to draw bars corresponding to hours

```java
public void drawHistogram() {
    double barHeight;
    double totalSpeeders = speedersSeen();
    for (int hour = 0; hour < speedersAt.length; hour++) {
        barHeight = (speedersAt[hour]/totalSpeeders)*graphHeight;
        new FilledRect(graphLeft + hour*barWidth,
                      graphBottom - barHeight,
                      barWidth-1,
                      barHeight,
                      canvas);
    }
}
```
Assume

- graphHeight is height of area in which graph is to be drawn
- graphLeft is x coordinate of bottom edge of graph
- graphBottom is y coordinate of bottom edge of graph
- barWidth is width of a single bar

Simple Histogram Output

Output of drawHistogram likely to look like this

- At any hour, the number of speeders on average is 1/24th of the total number of speeders
- Bars on average are 1/24th of the available vertical space

Finding the Largest Value in an Array

- Begin with the first value and then look hour by hour for a new maximum
- Variable max is equal to the largest number in the array so far. If a new, larger number is in the array, then max changes

  ```java
  private int maxSpeeders() {
    int max = speedersAt[0];
    for (int hour = 1; hour < speedersAt.length; hour++) {
      if (speedersAt[hour] > max) {
        max = speedersAt[hour];
      }
    }
    return max;
  }
  ```

Review

- Arrays are collections of primitive values or objects
- Learned how to
  - Declare them
  - Create them
  - Refer to items in them
  - Process all items in them in some way (move, hide)
  - Gather information from them (sum, max)

Collections With Variable Sizes

- A new application: timing and scoring of a cross-country race

```
<table>
<thead>
<tr>
<th>Place</th>
<th>Int No.</th>
<th>Finish Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>127</td>
<td>12:00</td>
</tr>
<tr>
<td>2</td>
<td>128</td>
<td>12:01</td>
</tr>
<tr>
<td>3</td>
<td>129</td>
<td>12:02</td>
</tr>
<tr>
<td>4</td>
<td>130</td>
<td>12:03</td>
</tr>
<tr>
<td>5</td>
<td>131</td>
<td>12:04</td>
</tr>
<tr>
<td>6</td>
<td>132</td>
<td>12:05</td>
</tr>
<tr>
<td>7</td>
<td>133</td>
<td>12:06</td>
</tr>
<tr>
<td>8</td>
<td>134</td>
<td>12:07</td>
</tr>
<tr>
<td>9</td>
<td>135</td>
<td>12:08</td>
</tr>
<tr>
<td>10</td>
<td>136</td>
<td>12:09</td>
</tr>
<tr>
<td>11</td>
<td>137</td>
<td>12:10</td>
</tr>
<tr>
<td>12</td>
<td>138</td>
<td>12:11</td>
</tr>
<tr>
<td>13</td>
<td>139</td>
<td>12:12</td>
</tr>
</tbody>
</table>
```

Team Score

- Add placements of a team's four fastest racers.
- Last digit of runner's bib indicates team
- Team 1's score = 1+2+4+6 = 13
Program Organization

- **RaceController**
  - Extension of WindowController
  - User interface to enable officials to enter timing data
- **RaceStatistics**
  - Uses array to keep track of data entered
  - Methods to compute team score, etc.

Parallel Arrays vs. Arrays of Objects

Need to keep track of pairs of bib numbers and times
- Two separate arrays
  - Arrays are “parallel arrays,” one number from one associated with one from other
  ```java
  private int[] bibNumber;
  private String[] elapsedTime;
  ```
- Single array of racer information
  - Assumes definition of a RacerInfo class
  ```java
  private RacerInfo[] racer;
  ```

RacerInfo Class

```java
public class RacerInfo {
  private int bibNumber;
  private String time;
  public RacerInfo(int number, String finishingTime) {
    bibNumber = number;
    time = finishingTime;
  }
  public int getBib() {
    return bibNumber;
  }
  public String getTime() {
    return time;
  }
  public int getTeam() {
    return bibNumber % 10;
  }
}
```

Keeping Track of Size

- must specify size to construct racer array.
- often perfect size info unknown, but can give upper limit
- use upper limit as size
- separately keep track of actual number of items in array

```java
private static final int TEAMSIZE = 100;
private static final int TEAMSINMEET = 3;
private RacerInfo[] racer = new RacerInfo(TEAMSIZE*TEAMSINMEET);
private int racerCount;
```

Adding Array Entries

- Check that there’s room left
- Add new item to the end
- Update count of items

```java
public void addRacer(int bib, String time) {
  if (racerCount < racer.length) {
    racer[racerCount] = new RacerInfo(bib, time);
    racerCount++;
  }
}
```
Iterating Through Collection of Variable Size

- Similar to earlier for loops
- Use array size variable to determine when to stop

Ex. To create a string of race results for printing

```java
public String individualResults() {
    String results = "";
    for (int place = 0; place < racerCount; place++) {
        results = results + (place+1) + ". " +
                    "Racer" + racer[place].getBib() + " " +
                    "in",
    }
    return results;
}
```

Finding an Element

- Use a for loop
- Keep going as long as not found and items left to consider

```java
public int getPlacement(int bib) {
    int result = -1
    for (int place = 0;
        place < racerCount && result == -1
        place++) {
        if (racer[place].getBib() == bib) {
            result = place+1;
        }
    }
    return result;
}
```

Alternate Version

```java
public int getPlacement(int bib) {
    for (int place = 0; place < racerCount; place++) {
        if (racer[place].getBib() == bib) {
            return place+1;
        }
    }
    return -1;
}
```

Computing a Team’s Score

A combination of
- Finding an element
- Adding to a running total

```java
public int teamScore(int teamNo) {
    int racersCounted = 0;
    int score = 0;
    for (int place = 0;
        place < racerCount && racersCounted < 4;
        place++) {
        if (racer[place].getTeam() == teamNo) {
            racersCounted++;
            score = score + (place + 1);
        }
    }
    if (racersCounted < 4) {
        score = -1;
    }
    return score;
}
```

Ordered Arrays

- Previous examples assumed
  - Array items ordered by elapsed time
  - Items supplied to array in correct order

What if we want to add or delete items and guarantee that correct order is maintained?

Adding to an Ordered Array

The racer that should be associated with index 4 is missing
Runner with bib 200 was omitted from the array

Adding the New Item

Need to
- Find appropriate index for the new item
- Shift existing items out of the way
- Insert new item
- Update the count

Shifting Array Entries

Shifting Racer Entries

To make room for runner 200 at index 4
racer[10] = racer[9];
racer[9] = racer[8];
racer[8] = racer[7];
racer[7] = racer[6];
racer[6] = racer[5];
racer[5] = racer[4];

Note that each line is of the form
racer[position] = racer[position-1]

Loop to Shift Array Entries

for ( int position = racerCount; position > insertionPos; position-- ) {
    racer[position] = racer[position-1];
}

Why does the loop go backward?

Putting It Together

To insert at a specific index
public void addRacerAtPosition( int bib, String time, int insertionPos ) {
    if ( insertionPos > racer.length ) return;
    for ( int position = racerCount; position > insertionPos; position-- ) {
        racers[position] = racers[position-1];
    }
    racers[insertionPos] = new RacerInfo( bib, time );
    racerCount++;
}
Removing from an Array

To shift entries 7, 8, 9 left (and delete 6)

\[
\begin{align*}
\text{racer}[6] &= \text{racer}[7]; \\
\text{racer}[7] &= \text{racer}[8]; \\
\text{racer}[8] &= \text{racer}[9]; \\
\text{racer}[9] &= \text{null};
\end{align*}
\]

Putting it all Together

```java
public void removeRacerAtPosition( int position ) {
  if ( position < racerCount ) {
    racerCount--; 
    for ( int place = position; place < racerCount; place++ ) {
      racer[place] = racer[place + 1];
    }
    racer[racerCount] = null;
  }
}
```