Building Java Programs

Chapter 7
Lecture 16: Arrays as Parameters, Arrays for Tallying

reading: 4.3, 7.6
Array parameter (declare)

public static type methodName(type[] name) {

• Example:

// Returns the average of the given array of numbers.
public static double average(int[] numbers) {
    int sum = 0;
    for (int i = 0; i < numbers.length; i++) {
        sum += numbers[i];
    }
    return (double) sum / numbers.length;
}

• You don't specify the array's length (but you can examine it).
Array parameter (call)

methodName (arrayName);

- Example:

```java
public class MyProgram {
    public static void main(String[] args) {
        // figure out the average TA IQ
        int[] iq = {126, 84, 149, 167, 95};
        double avg = average(iq);
        System.out.println("Average IQ = " + avg);
    }
    ...
}
```

- Notice that you don't write the [] when passing the array.
Array return (declare)

public static type[] methodName(parameters) {

• Example:

// Returns a new array with two copies of each value.
// Example: [1, 4, 0, 7] -> [1, 1, 4, 4, 0, 0, 7, 7]
public static int[] double(int[] numbers) {
    int[] result = new int[2 * numbers.length];
    for (int i = 0; i < numbers.length; i++) {
        result[2 * i] = numbers[i];
        result[2 * i + 1] = numbers[i];
    }
    return result;
}
Array return (call)

\[
\text{type}[\ ] \text{name} = \text{methodName}(\text{parameters});
\]

- Example:
  
  ```java
  public class MyProgram {
      public static void main(String[] args) {
          int[] iq = {126, 84, 149, 167, 95};
          int[] doubled = double(iq);
          System.out.println(Arrays.toString(doubled));
      }
  }
  ...
  ```

- Output:
  
  `[126, 126, 84, 84, 149, 149, 167, 167, 95, 95]`
Array reversal question

• Write code that reverses the elements of an array.
  • For example, if the array initially stores:
    \[11, 42, -5, 27, 0, 89]\n  • Then after your reversal code, it should store:
    \[89, 0, 27, -5, 42, 11]\n  • The code should work for an array of any size.
  • Hint: think about swapping various elements...
Algorithm idea

- Swap pairs of elements from the edges; work inwards:

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>89</td>
<td>0</td>
<td>27</td>
<td>-5</td>
<td>42</td>
<td>11</td>
</tr>
</tbody>
</table>
public static void main(String[] args) {
    int a = 7;
    int b = 35;

    // swap a with b?
    a = b;
    b = a;
    System.out.println(a + " " + b);
}

• What is wrong with this code? What is its output?

• The red code should be replaced with:

  int temp = a;
  a = b;
  b = temp;
Flawed algorithm

• What's wrong with this code?

```java
text=["int[] numbers = [11, 42, -5, 27, 0, 89];

// reverse the array
for (int i = 0; i < numbers.length; i++) {
  int temp = numbers[i];
  numbers[i] = numbers[numbers.length - 1 - i];
  numbers[numbers.length - 1 - i] = temp;
}
```

• The loop goes too far and un-reverses the array! Fixed version:

```java
text=["for (int i = 0; i < numbers.length / 2; i++) {
  int temp = numbers[i];
  numbers[i] = numbers[numbers.length - 1 - i];
  numbers[numbers.length - 1 - i] = temp;
}
"]```
Array reverse question 2

• Turn your array reversal code into a reverse method.
  • Accept the array of integers to reverse as a parameter.

  ```csharp
  int[] numbers = {11, 42, -5, 27, 0, 89};
  reverse(numbers);
  ```

• How do we write methods that accept arrays as parameters?
• Will we need to return the new array contents after reversal?
  ...
Reference semantics

reading: 7.3
A swap method?

• Does the following swap method work? Why or why not?

```java
public static void main(String[] args) {
    int a = 7;
    int b = 35;

    // swap a with b?
    swap(a, b);
    System.out.println(a + " " + b);
}

public static void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}
```
Value semantics

- **value semantics**: Behavior where values are copied when assigned, passed as parameters, or returned.
  
  - All primitive types in Java use value semantics.
  - When one variable is assigned to another, its value is copied.
  - Modifying the value of one variable does not affect others.

```java
int x = 5;
int y = x; // x = 5, y = 5
y = 17; // x = 5, y = 17
x = 8; // x = 8, y = 17
```
Reference semantics (objects)

- **reference semantics**: Behavior where variables actually store the address of an object in memory.
  - When one variable is assigned to another, the object is *not* copied; both variables refer to the *same object*.
  - Modifying the value of one variable *will* affect others.

```java
int[] a1 = {4, 15, 8};
int[] a2 = a1;       // refer to same array as a1
a2[0] = 7;
System.out.println(Arrays.toString(a1));  // [7, 15, 8]
```

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>8</td>
</tr>
</tbody>
</table>
```

```
a1 ← index 0 1 2
valu e
```

```
a2
```
Arrays and objects use reference semantics. Why?

- **efficiency.** Copying large objects slows down a program.
- **sharing.** It's useful to share an object's data among methods.

```java
drawingPanel panel1 = new drawingPanel(80, 50);
drawingPanel panel2 = panel1;  // same window
panel2.setBackground(Color.CYAN);
```
Objects as parameters

- When an object is passed as a parameter, the object is not copied. The parameter refers to the same object.
  - If the parameter is modified, it will affect the original object.

```java
public static void main(String[] args) {
    DrawingPanel window = new DrawingPanel(80, 50);
    window.setBackground(Color.YELLOW);
    example(window);
}

public static void example(DrawingPanel panel) {
    panel.setBackground(Color.CYAN);
    ...
}
```
Arrays pass by reference

- Arrays are passed as parameters by *reference*.
- Changes made in the method are also seen by the caller.

```java
public static void main(String[] args) {
    int[] iq = {126, 167, 95};
    increase(iq);
    System.out.println(Arrays.toString(iq));
}

public static void increase(int[] a) {
    for (int i = 0; i < a.length; i++) {
        a[i] = a[i] * 2;
    }
}
```

- Output:
  
<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>252</td>
<td>334</td>
<td>190</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>index</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>value</td>
<td>252</td>
<td>334</td>
<td>190</td>
</tr>
</tbody>
</table>
```
Array reverse question 2

- Turn your array reversal code into a reverse method.
  - Accept the array of integers to reverse as a parameter.

```java
int[] numbers = {11, 42, -5, 27, 0, 89};
reverse(numbers);
```

**Solution:**

```java
public static void reverse(int[] numbers) {
    for (int i = 0; i < numbers.length / 2; i++) {
        int temp = numbers[i];
        numbers[i] = numbers[numbers.length - 1 - i];
        numbers[numbers.length - 1 - i] = temp;
    }
}
```
Array parameter questions

• Write a method `swap` that accepts an arrays of integers and two indexes and swaps the elements at those indexes.

```java
int[] a1 = {12, 34, 56};
swap(a1, 1, 2);
System.out.println(Arrays.toString(a1));  // [12, 56, 34]
```

• Write a method `swapAll` that accepts two arrays of integers as parameters and swaps their entire contents.
  • Assume that the two arrays are the same length.

```java
int[] a1 = {12, 34, 56};
int[] a2 = {20, 50, 80};
swapAll(a1, a2);
System.out.println(Arrays.toString(a1));  // [20, 50, 80]
System.out.println(Arrays.toString(a2));  // [12, 34, 56]
```
Array parameter answers

// Swaps the values at the given two indexes.
public static void swap(int[] a, int i, int j) {
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}

// Swaps the entire contents of a1 with those of a2.
public static void swapAll(int[] a1, int[] a2) {
    for (int i = 0; i < a1.length; i++) {
        int temp = a1[i];
        a1[i] = a2[i];
        a2[i] = temp;
    }
}
Array return question

• Write a method `merge` that accepts two arrays of integers and returns a new array containing all elements of the first array followed by all elements of the second.

```java
int[] a1 = {12, 34, 56};
int[] a2 = {7, 8, 9, 10};

int[] a3 = merge(a1, a2);
System.out.println(Arrays.toString(a3));  // [12, 34, 56, 7, 8, 9, 10]
```

• Write a method `merge3` that merges 3 arrays similarly.

```java
int[] a1 = {12, 34, 56};
int[] a2 = {7, 8, 9, 10};
int[] a3 = {444, 222, -1};

int[] a4 = merge3(a1, a2, a3);
System.out.println(Arrays.toString(a4));  // [12, 34, 56, 7, 8, 9, 10, 444, 222, -1]
```
// Returns a new array containing all elements of a1 followed by all elements of a2.
public static int[] merge(int[] a1, int[] a2) {
    int[] result = new int[a1.length + a2.length];
    for (int i = 0; i < a1.length; i++) {
        result[i] = a1[i];
    }
    for (int i = 0; i < a2.length; i++) {
        result[a1.length + i] = a2[i];
    }
    return result;
}
Array return answer 2

// Returns a new array containing all elements of a1,a2,a3.
public static int[] merge3(int[] a1, int[] a2, int[] a3) {
    int[] a4 = new int[a1.length + a2.length + a3.length];
    for (int i = 0; i < a1.length; i++) {
        a4[i] = a1[i];
    }
    for (int i = 0; i < a2.length; i++) {
        a4[a1.length + i] = a2[i];
    }
    for (int i = 0; i < a3.length; i++) {
        a4[a1.length + a2.length + i] = a3[i];
    }
    return a4;
}

// Shorter version that calls merge.
public static int[] merge3(int[] a1, int[] a2, int[] a3) {
    return merge(merge(a1, a2), a3);
}
Value/Reference Semantics

• Variables of primitive types store values directly:
  
  \[
  \begin{array}{c|c|c}
    & age & cats \\
    \hline
    value & 20 & 3 \\
  \end{array}
  \]

• Values are copied from one variable to another:
  
  \[
  \text{cats} = \text{age};
  \]

  \[
  \begin{array}{c|c|c|c}
    & age & cats \\
    \hline
    value & 20 & 20 \\
  \end{array}
  \]

• Variables of object types store references to memory:

  \[
  \begin{array}{c|c|c|c}
    & grades & index \\
    \hline
    \text{value} & 89 & 78 & 93 \\
  \end{array}
  \]

• References are copied from one variable to another:

  \[
  \text{scores} = \text{grades};
  \]

  \[
  \begin{array}{c|c|c|c}
    & scores \\
    \hline
    \text{value} & \\
  \end{array}
  \]
Text processing

reading: 7.2, 4.3
String traversals

• The chars in a String can be accessed using the charAt method.
  • accepts an int index parameter and returns the char at that index

```java
String food = "cookie";
char firstLetter = food.charAt(0);  // 'c'
System.out.println(firstLetter + " is for " + food);
```

• You can use a for loop to print or examine each character.

```java
String major = "CSE";
for (int i = 0; i < major.length(); i++) {
    char c = major.charAt(i);  // C
    System.out.println(c);  // S
    // E
}```
A multi-counter problem

Problem: Write a method `mostFrequentDigit` that returns the digit value that occurs most frequently in a number.

- Example: The number 669260267 contains: one 0, two 2s, four 6es, one 7, and one 9. `mostFrequentDigit(669260267)` returns 6.

- If there is a tie, return the digit with the lower value. `mostFrequentDigit(57135203)` returns 3.
A multi-counter problem

- We could declare 10 counter variables …

  ```
  int counter0, counter1, counter2, counter3, counter4,
  counter5, counter6, counter7, counter8, counter9;
  ```

- But a better solution is to use an array of size 10.
  - The element at index \( i \) will store the counter for digit value \( i \).
  - Example for 669260267:

    | index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
    |-------|---|---|---|---|---|---|---|---|---|---|
    | value | 1 | 0 | 2 | 0 | 0 | 0 | 4 | 1 | 0 | 0 |

- How do we build such an array? And how does it help?
Creating an array of tallies

// assume n = 669260267
int[] counts = new int[10];
while (n > 0) {
    // pluck off a digit and add to proper counter
    int digit = n % 10;
    counts[digit]++;
    n = n / 10;
}

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
public static int mostFrequentDigit(int n) {
    int[] counts = new int[10];
    while (n > 0) {
        int digit = n % 10;  // pluck off a digit and tally it
        counts[digit]++;
        n = n / 10;
    }

    // find the most frequently occurring digit
    int bestIndex = 0;
    for (int i = 1; i < counts.length; i++) {
        if (counts[i] > counts[bestIndex]) {
            bestIndex = i;
        }
    }

    return bestIndex;
}
Section attendance question

- Read a file of section attendance (see next slide):
  yynyyynayayynyyayanyyyaynayyanayyyanyayna
  ayyanyyyyayanaayyanayyyananayaynyayynynyna
  yyayaynyyyayynynnyyyayyanayaynannnnyyyyayayny

- And produce the following output:

  Section 1
  Student points: [30, 27, 29, 24, 19]
  Student grades: [100.0, 90.0, 96.7, 80.0, 63.3]

  Section 2
  Student points: [27, 30, 24, 24, 14]
  Student grades: [90.0, 100.0, 80.0, 80.0, 46.6]

  Section 3
  Student points: [27, 26, 27, 30, 24]
  Student grades: [90.0, 86.7, 90.0, 100.0, 80.0]

- Students earn 5 points for each section attended up to 30.
Section input file

<table>
<thead>
<tr>
<th>student</th>
<th>1234512345123451234512345123451234512345123</th>
</tr>
</thead>
<tbody>
<tr>
<td>week</td>
<td>1 2 3 4 5 6 7 8 9</td>
</tr>
<tr>
<td>section 1</td>
<td>yynyyynayayynyyayahyyyyyaynayyayyayyananyyanyay</td>
</tr>
<tr>
<td>section 2</td>
<td>ayyanyyyyyayanaaayyanayyyananayayaynyayayynnyn</td>
</tr>
<tr>
<td>section 3</td>
<td>yyayaynyyyayyanynnyyyayyanayyanannnnyyyayyyayay</td>
</tr>
</tbody>
</table>

- Each line represents a section.
- A line consists of 9 weeks' worth of data.
  - Each week has 5 characters because there are 5 students.
  - Within each week, each character represents one student.
    - a means the student was absent (+0 points)
    - n means they attended but didn't do the problems (+2 points)
    - y means they attended and did the problems (+5 points)
import java.io.*;
import java.util.*;

public class Sections {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner input = new Scanner(new File("sections.txt"));
        int section = 1;
        while (input.hasNextLine()) {
            String line = input.nextLine(); // process one section
            int[] points = new int[5];
            for (int i = 0; i < line.length(); i++) {
                int student = i % 5;
                int earned = 0;
                if (line.charAt(i) == 'y') { // c == 'y' or 'n' or 'a'
                    earned = 5;
                } else if (line.charAt(i) == 'n') {
                    earned = 2;
                }
                points[student] = Math.min(30, points[student] + earned);
            }
            double[] grades = new double[5];
            for (int i = 0; i < points.length; i++) {
                grades[i] = 100.0 * points[i] / 20.0;
            }

            System.out.println("Section " + section);
            System.out.println("Student points: " + Arrays.toString(points));
            System.out.println("Student grades: " + Arrays.toString(grades));
            System.out.println();
            section++;
        }
    }
}
Data transformations

- In many problems we transform data between forms.
  - Example: digits $\rightarrow$ count of each digit $\rightarrow$ most frequent digit
  - Often each transformation is computed/stored as an array.
  - For structure, a transformation is often put in its own method.

- Sometimes we map between data and array indexes.
  - by position (store the $i^{th}$ value we read at index $i$)
  - tally (if input value is $i$, store it at array index $i$)
  - explicit mapping (count 'J' at index 0, count 'X' at index 1)

- Exercise: Modify our Sections program to use static methods that use arrays as parameters and returns.
// This program reads a file representing which students attended
// which discussion sections and produces output of the students'
// section attendance and scores.

import java.io.*;
import java.util.*;

public class Sections2 {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner input = new Scanner(new File("sections.txt"));
        int section = 1;
        while (input.hasNextLine()) {
            // process one section
            String line = input.nextLine();
            int[] points = countPoints(line);
            double[] grades = computeGrades(points);
            results(section, points, grades);
            section++;
        }
    }

    // Produces all output about a particular section.
    public static void results(int section, int[] points, double[] grades) {
        System.out.println("Section " + section);
        System.out.println("Student scores: "+ Arrays.toString(points));
        System.out.println("Student grades: "+ Arrays.toString(grades));
        System.out.println();
    }

    // Produces all output about a particular section.
    public static void results(int section, int[] points, double[] grades) {
        System.out.println("Section " + section);
        System.out.println("Student scores: "+ Arrays.toString(points));
        System.out.println("Student grades: "+ Arrays.toString(grades));
        System.out.println();
    }

    ...

Copyright 2010 by Pearson Education
...  

// Computes the points earned for each student for a particular section.
public static int[] countPoints(String line) {
    int[] points = new int[5];
    for (int i = 0; i < line.length(); i++) {
        int student = i % 5;
        int earned = 0;
        if (line.charAt(i) == 'y') {    // c == 'y' or c == 'n'
            earned = 3;
        } else if (line.charAt(i) == 'n') {
            earned = 2;
        }
        points[student] = Math.min(20, points[student] + earned);
    }
    return points;
}

// Computes the percentage for each student for a particular section.
public static double[] computeGrades(int[] points) {
    double[] grades = new double[5];
    for (int i = 0; i < points.length; i++) {
        grades[i] = 100.0 * points[i] / 20.0;
    }
    return grades;
}