CS 112 Introduction to Programming

Arrays: Loop Patterns (break)

Yang (Richard) Yang
Computer Science Department
Yale University
308A Watson, Phone: 432-6400
Email: yry@cs.yale.edu

Recap: Exceptions

- Exception: representing a runtime error
- Java allows a method to "throw" an exception, when the method detects a run-time error
- Handling exceptions
  - Declare a throw clause
  - Use try/catch

Recap: CaesarFile using Loop

```java
public static void encode(Scanner scan, int key) {
    for (; scan.hasNextLine(); )
        String text = scan.nextLine();
        String result = Caesar.encode(text, key);
        System.out.println(result);
}
```

Summary: Flow Control Statements

- Loop statements
  - for
  - while
  - do/while
- Choosing the loop control structure depends on the specific situation and personal taste

Summary: for loop

- for loop is easy to read, since it provides visual aid to recognize the four components
  - If complex or empty initialization, increment, or condition, a for loop may not read as well as a while or do/while loop
  - typically used in sequential iteration

Admin.

- Puzzle Day from Friday to Monday
Summary: while, do/while loop

- The choice between do/while and while depends on the logic of the program:
  - First check condition or first execute statement

Recap: Accessing Elements: Example

```java
int[] numbers = new int[10];
numbers[4] = 99;
numbers[6] = 2;
System.out.println(numbers[2-1]);
if (numbers[4] > 10) {
    System.out.println("numbers[4] > 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>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>99</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

BasicArrays.java

Accessing Elements: Example

- What does this code segment do?

```java
int[] numbers = new int[8];
numbers[0] = 1; numbers[1] = 1;
for (int i = 2; i < 8; i++) {
    numbers[i] = numbers[i-1] + numbers[i-2];
}
```

<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>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>13</td>
</tr>
</tbody>
</table>

Quick Array Initialization

- An initializer list can be used to instantiate and initialize an array in one step

```java
<type>[] <name> = {<value>, <value>, ... <value>};
```

- Examples:
  ```java
  int[] numbers = {12, 49, -2, 26, 17, -6};
  char[] letterGrades = {'A', 'B', 'C', 'D', 'F'};
  String[] wordList = {"cs112", "computer", "television"};
  ```

- The values are delimited by braces and separated by commas
- The new operator is not used
- The compiler figures out the size by counting the values in the initializer list
- Elements in array are initialized with the values in the initializer list

Out-of-bounds (OOB)

- Reading or writing any index outside of range will throw an `ArrayIndexOutOfBoundsException`.
- Example (which line(s) will cause OOB):
  ```java
  int[] data = new int[10];
  System.out.println(data[9]);
  System.out.println(data[-1]);
  System.out.println(data[data.length]);
  ```

<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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Out-of-bounds (OOB)

- Reading or writing any index outside of range will throw an `ArrayIndexOutOfBoundsException`.

Example (which line(s) will cause OOB):

1. `int[] data = new int[10];`  // okay
2. `System.out.println(data[9]);`  // okay
3. `System.out.println(data[data.length]);`  // OOB

<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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Array length field

- Java provides a `length` field for each array to store the number of elements.
  - It does not use parentheses like a String's `.length()`
  - `length` holds the number of elements, *not* the largest index

Example:

```java
int[] numbers = new int[8];
System.out.println(numbers.length); // 8
```

What expressions refer to the index of:

- the last element of any array?
- the middle element?

Array as Parameter (Declare)

```java
public static <type> <methodName>(<type>[] <arrayName>) {
}
```

Example:

```java
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;
}
```

Array as Parameter (Call)

```java
<String> <methodName>(<arrayName>);
```

Example:

```java
public class MyProgram {
    public static void main(String[] args) {
        // figure out the average TA IQ
        int[] iq = {129, 109, 149, 167, 125};
        double avg = average(iq);
        System.out.println("Average IQ = " + avg);
    }
    ...
}
```

Notice that you don’t write the [] when passing the array.

Array as Parameter (Call)

Example: Command-Line Arguments

- The signature of the `main` method indicates that it takes an array of `String` 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 an array of three `String` objects into main method:
  ```bash
  > java Calc 3 + 5
  ``
  - The strings "3", "+", "5", are stored at indexes 0-2 of the String array `args`

Exercise: Implement a simple command-line calculator:

```java
<num1> +|-|*|- <num2>
```

See Calc.java
Foundational Programming Concepts

- any program you might want to write
  - objects
  - methods and classes
  - graphics, sound, and image I/O
  - arrays
  - conditionals and loops
  - Math
  - text I/O
  - primitive data types
  - assignment statements

Exercise: Early Exit and boolean method return

- Design a method `hasAnOddDigit`: returns `true` if any digit of an integer `n` is odd, e.g.,
  - `hasAnOddDigit(4822116)` returns `true`
  - `hasAnOddDigit(2448)` returns `false`

- Design questions:
  - How do we loop over each digit from `n`?
    - `digit = n % 10`;
  - If the digit we just saw is odd, can we draw any conclusion?
    - Yes. We found an evidence, return `true`.
  - If the digit is even, can we draw any conclusion?
    - No, unless we have seen all digits.

Exercise: Early Exit and boolean method return

- Designs method `allDigitsOdd`: returns `true` if every digit of an integer is odd.
  - `allDigitsOdd(135319)` returns `true`
  - `allDigitsOdd(9174529)` returns `false`

- Design questions:
  - If the digit we just saw is odd, can we draw any conclusion?
    - Yes. We found a counter evidence, and can return `false`.
  - If the digit is even, can we draw any conclusion?
    - No, unless we have seen all digits.

Roadmap: Program Flow Control

- Deterministic for loop (loop counter)
- cumulative scan loop
- Early exit loops
- Fencepost loops

Boolean Return Answer

```java
public static boolean hasAnOddDigit(int n) {
    while (n != 0) {
        int digit = n % 10;
        if (digit % 2 != 0) {
            return true; // find an example, enough to draw conclusion
        }
        n = n / 10;
    }
    return false; // if we reach here, no evidence of odd, return false
}
```

```java
public static boolean allDigitsOdd(int n) {
    while (n != 0) {
        if (n % 2 == 0) {
            // find a counter example, enough to draw conclusion
            return false;
        }
        n = n / 10;
    }
    return true; // if we reach here, no evidence of odd, return true
}
```
Exercise: MatchDB

- Design a program to query a match-making database
  - database (file) format:
    - line 1: number of candidates
    - each line represents a candidate: name age and then a sequence of words (tags) each describing a character
  - user command
    - list: display each candidate
    - matchAny <a list of tags> // print first match any tag
    - matchAll <a list of tag words> // print first matches all tags

Backup Slides

Manipulating Array:
Array Reversal

- Write code that reverses the elements of an array.
- For example, if the array initially stores:
  [11, 42, -5, 27, 0, 89]
- Then after your reversal code, it should store:
  [89, 0, 27, -5, 42, 11]
- The code should work for an array of any size.

Algorithm Idea

- Swap pairs of elements from the edges; work inwards:
  index 0 1 2 3 4 5
  value 89 0 27 -5 42 11
  ↑ ↑ ↑ ↑ ↑