CS 112 Introduction to Programming

Reference Semantics;
2D Arrays;
Array as State

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

Admin

- Exam 1
  - Max: 50
  - Avg/median: 32

- Overall course grade
Recap: Arrays

- Declare an array, e.g.,
  ```java
  int[] numbers = new int[10];
  int[][] grid = new int[3][4];
  char[] letterGrades = {'A', 'B', 'C', 'D', 'F'};
  ```

- Access array elements, e.g.,
  ```java
  numbers[1] = 3;
  numbers[2] = 2;
  numbers[numbers[2]] = 42;
  Grid[1][2] = 3;
  ```

- Pass arrays as parameters, e.g.,
  ```java
  public static double average(int[] numbers)
  ```

Arrays class

- Provides many useful methods that take an array as a parameter, e.g.,
  - `toString(myArray)` // print array

http://download.oracle.com/javase/1.5.0/docs/api/java/util/Arrays.html
Array Return (declare)

```java
public static <type>[] <method>(<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[] stutter(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)

```java
<type>[] <name> = <method>(<parameters>);

    Example:
    public class MyProgram {
        public static void main(String[] args) {
            int[] iq = {126, 84, 149, 167, 95};
            int[] stuttered = stutter(iq);
            System.out.print(Arrays.toString(stuttered));
        }
    }

    Output:
    [126, 126, 84, 84, 149, 149, 167, 167, 95, 95]
```
Array Merge 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]
```

Array Merge

// 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 Merge 3 Question

- 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]
```

Array Merge 3

// 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);
}

Discussion: Which version do you use?
 Complexity Analysis

- V1
  - Creation of array (static complexity)
    - One array of size $N_1 + N_2 + N_3$
  - Copy values (dynamic)
    - $N_1 + N_2 + N_3$ values

- V2
  - Creation of array
    - First size $N_1 + N_2$; second size $(N_1 + N_2) + N_3$
  - Copy values
    - First $N_1 + N_2$
    - Then $N_1 + N_2 + N_3$

 Value vs Reference Semantics
Motivation: Primitive swap

```java
public static void main(String[] args) {
    int a = 1;
    int b = 2;
    System.out.println(a + " " + b); // ?
    swap(a, b);
    System.out.println(a + " " + b); // ?
}

public static void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
    System.out.println(a + " " + b); // ?
}
```

Motivation: Array Reversal

```java
public static void main(String[] args) {
    int[] a = {1, 2};
    System.out.println(Arrays.toString(a));
    reverse(a);
    System.out.println(Arrays.toString(a));
}

public static void reverse(int[] a) {
    for (int i = 0; i < a.length / 2; i++) {
        int temp = a[i];
        a[i] = a[a.length-1-i];
        a[a.length-1-i] = temp;
    }
}
```
Value Semantics and Reference Semantics

- Primitive data types use value semantics: variable stores value
  
  ```
  int x = 5;
  ```

- Non primitive data types (e.g., arrays and objects) use reference semantics: variable stores reference address
  
  ```
  int[] a = {1, 2, 3};
  ```

- See `printVariable()` of `BasicArrays.java`

References

- Why reference variables
  - Efficiency. Copying large objects/arrays slows down a program
  - Sharing. It's useful to share an object/array's data among methods
Value/Reference Semantics and Assignment

- When variable $a$ is assigned to variable $b$: 
  \[ b = a \]
  the content of $a$ is always copied to $b$
  - if a value type, then it is the **value** that is copied
  - if a reference type, then it is the **reference** that is copied, $b$ becomes an **alias** of $a$}

Value/Reference Semantics and Parameter Passing

- Each time a method is called, the *actual argument* in the invocation is copied into the corresponding *formal argument*
  - if a value type, then it is the **value** that is copied
  - if a reference type, then it is the **reference** that is copied
    - The actual argument and the formal argument now refer to the same object
    - Modification in the method will affect the original object.
Example: Value Semantics

- Modifying the value of one value 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
```

Example: Reference Semantics

- Modifying an object/array through one reference variable changes the object/array and hence all references to the same object/array see the changes.

```java
int[] a1 = {1, 2, 3};
int[] a2 = a1;  // same array
a2[1] = 4;
```
Example: Reference Semantics

- Modifying an object/array through one reference variable changes the object/array and hence all references to the same object/array see the changes.

```java
int[] a1 = {1, 2, 3};
int[] a2 = a1;  // same array
a2[1] = 4;
```

Array Passed as Parameter

```java
public static void main(String[] args) {
    int[] iq = {120, 160, 95};
    doubleArray(iq);
    System.out.println(Arrays.toString(iq));
}

static void doubleArray (int[] iqp)
{
    for (int i = 0; i < iqp.length; i++)
        iqp[i] *= 2;
}
```
Array Parameter Question

- Write a method `swap` that accepts an array 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]
```

Array Parameter

```java
// 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;
}
```
Array Parameter Question

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

```java
int[] a1 = {10, 11, 12};
int[] a2 = {20, 21, 22};
swapAll(a1, a2);
System.out.println(Arrays.toString(a1));  // [20, 21, 22]
System.out.println(Arrays.toString(a2));  // [10, 11, 12]
```

Attempt 1

```java
// Does this method swap the entire contents of a1 with those of a2?
public static void swapAll1(int[] a1, int[] a2) {
    int[] temp = a1;
    a1 = a2;
    a2 = temp;
}
```

ArraySwap.java
Why it does not work?

Init.

```java
public static void main(String[] args) {
    int[] a1 = {10, 11, 12};
    int[] a2 = {20, 21, 22};
}
```

Why it does not work?

Invoke

```java
public static void main(String[] args) {
    int[] a1 = {10, 11, 12};
    int[] a2 = {20, 21, 22};
    swapAll (a1, a2);
}
```
Why it does not work?
Invoke

```java
public static void main(String[] args) {
    int[] a1 = {10, 11, 12};
    int[] a2 = {20, 21, 22};
    swapAll(a1, a2);
}
```

```
static void swapAll(int[] a1, int[] a2) {
}
```
Why it does not work?
Swap (a1 = a2)

```java
public static void main(String[] args) {
    int[] a1 = {10, 11, 12};
    int[] a2 = {20, 21, 22};
    swapAll(a1, a2);
}
```

```java
static void swapAll(int[] a1, int[] a2) {
    int[] temp = a1;
    a1 = a2;
}
```

Why it does not work?
Swap (a2 = temp)

```java
public static void main(String[] args) {
    int[] a1 = {10, 11, 12};
    int[] a2 = {20, 21, 22};
    swapAll(a1, a2);
}
```

```java
static void swapAll(int[] a1, int[] a2) {
    int[] temp = a1;
    a1 = a2;
    a2 = temp;
}
```
Why it does not work?
After swapAll

```java
public static void main(String[] args) {
    int[] a1 = {10, 11, 12};
    int[] a2 = {20, 21, 22};
    swapAll(a1, a2);
}
```

Solution

```java
// 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;
    }
}
```
Understanding Two Dimensional Array: An Array of Arrays

```java
int[][] table = new int[3][4];
for (int i = 0; i < 3; i++)
    for (int j = 0; j < 4; j++)
        table[i][j] = i + j;
```

Irregular Two-Dimensional Array

```java
int[][] table = { {1, 2, 3, 4},
                {5, 6, 7},
                {8, 9},
                {0} );
```
**Initializer: Example**

```java
public class Test2DArray
{
    public static void main(String[] args)
    {
        int[][] days = { {1, 2, 3, 4},
                        {5, 6, 7},
                        {8, 9},
                        {0} };
        for (int i = 0; i < days.length; i++)
        {
            for (int j = 0; j < days[i].length; j++)
                System.out.print( days[i][j] );
            System.out.println();
        }
    }
}
```

**Roadmap: Arrays**

- Motivation, declaration, initialization, access
- Reference semantics: arrays as objects
- Example usage of arrays:
  - As counter (please read outside class)
  - As state
Using Array as Counters/Accumulators

- Create an array equal to the size of the number of categories
- Loop over each input
  - map input's value to array index
  - increase the array element at index
- Display result
  - Map each array index back to input to display

Grade Histogram Question

- Given a file of integer exam scores, where a score is between 0 and 100, e.g.,
  
  82
  66
  79
  63
  83

  Write a program that will print a histogram of stars indicating the number of students who earned each unique exam score.

  85:  *****
  86:  ************
  87:  ***
  88:  *
  91:  ****
Using Array as Counters/Accumulators

- Create an array equal to the size of the number of categories
  - Q: how many categories?
    ```java
    int[] counters = new int[101];
    ```

- Loop over each input
  - map input’s value to array index
    ```java
    grade -> counters[grade]
    ```
  - increase the array element at index

- Display result
  - Map each array index back to input to display

Grade Histogram

```java
// Reads a file of test scores and shows a histogram of the score distribution.
import java.io.*;
import java.util.*;
public class Histogram {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner input = new Scanner(new File("midterm.txt"));
        int[] counts = new int[101]; // counters of test scores 0 - 100
        while (input.hasNextInt()) { // read file into counts array
            int grade = input.nextInt();
            counts[grade]++;
            // if grade is 87, then counts[87]++
        }
        for (int i = 0; i < counts.length; i++) { // print star histogram
            if (counts[i] > 0) {
                System.out.printf("\%3d: \", i);
                for (int j = 0; j < counts[i]; j++) {
                    System.out.print("*");
                }
                System.out.println();
            }
        }
    }
}
```
Revision

- How about a bucket for every 5 points:
  - 00-04:
  - 05-09:
  - ..
  - 90-94:
  - 95-99:
  - 100:

Using Array as Counters/Accumulators

- Create an array equal to the size of the number of categories
  - Q: how many categories?
    - int[] counters = new int[100/5+1];

- Loop over each input
  - map input's value to array index
    - grade -> counters[grade/5]
  - increase the array element at index

- Display result
  - Map each array index back to input to display
    - [index * 5, index*5+4]
Revision

- How about the following buckets:
  - 00-59:
  - 60-64:
  - 65-69:
  - 70-74:
  - ...
  - 90-94:
  - 95-99:
  - 100:

Using Array as Counters/Accumulators

- Create an array equal to the size of the number of categories
  - Q: how many categories?
    int[] counters = new int[1+(100-60)/5+1];

- Loop over each input
  - map input’s value to array index
    if (grade < 60)
      index = 0;
    else
      index = (grade-60) / 5 + 1;
  - increase the array element at index

- Display result
  - Map each array index back to input to display
Exercise: Excel Style

- **Buckets specified in an array**
  - `[59, 75, 85, 90] =>
    - 0 – 59 bucket 1
    - 60 – 75 bucket 2
    - 76 – 85 bucket 3
    - 86 – 90 bucket 4
    - 91 and above bucket 5

Letter Frequency Counting

- **Objective:** Count the frequency of letters a to z, ignoring case, in a text file.

  - The inventor of Morse code, Samuel Morse (1791-1872), counted letter frequencies to assign the simpler codes to the more frequently used letters. The counters he obtained:
    - E: 12000
    - T: 9000
    - A: 8000
    - ...
    - X: 400
    - Z: 200
Using Array as Counters/Accumulators

- Create an array equal to the size of the number of categories
  - Q: how many categories?
    ```java
    int[] counters = new int[26];
    ```
- Loop over each input
  - map input's value to array index
    ```java
    ch -> counters[ch-'a']
    ch -> counters[ch-'A']
    ```
  - increase the array element at index
- Display result
  - Map each array index back to input to display
    ```java
    index -> (char)('a'+ index)
    ```

Array Elements as Counters

- Count the number of characters in a line:
  ```java
  int[] counts = new int[26];
  String line = scan.nextLine();
  for (int i = 0; i < line.length(); i++) {
    char ch = line.charAt(i);
    if ('a' <= ch && ch <= 'z') {
      counts[ch-'a']++;
    } else if ('A' <= ch && ch <= 'Z') {
      counts[ch-'A']++;
    }
  }
  ```
Array Index to Display

```java
public static void histogram(int[] counts) {
    int max = max(counts);

    for (int i = 0; i < counts.length; i++) {
        if (counts[i] > 0) {
            System.out.print((char)('a' + i) + " ");
            int h = counts[i];
            for (int j = 0; j < h; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
    // end of histogram
}
```

LetterHistogram.java

Roadmap: Arrays

- Motivation, declaration, initialization, access
- Reference semantics: arrays as objects
- Example usage of arrays:
  - As counter (please read outside class)
  - As state
Coupon-Collector Problem

- Given \( N \) different types of chocolates, and you get one random type each day. Simulate how many days you need in order to have (at least) one of each type.

Coupon-Collector Problem: Pseudo Code

```plaintext
// assume items are numbered 0 to N - 1
repeat when not collected all distinct items
    pick a random item
    if item is new
        record new item
```
// assume items are numbered 0 to N - 1
int distinct = 0;
while distinct < N
    pick a random item
    if item is new
        record new item
        distinct ++;

How?

// assume items are numbered 0 to N - 1
int distinct = 0;
boolean[] has = new boolean[N];
while distinct < N
    pick a random item
    if item is new
        has[item] = true; // record new item
        distinct ++;
public class CouponCollector {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        int cnt = 0;  // number of received collected
        int distinctCnt = 0;  // number of distinct

        boolean[] has = new boolean[N];  // keep state
        while (distinctCnt < N) {
            int val = rand(0, N-1);
            cnt++;
            if (!has[val]) {
                distinctCnt++;
                has[val] = true;  // update state
            }
        }

        // all N distinct cards found
        System.out.println(cnt);
    }
}
Simulate Self-Avoiding Walk

- Model.
  - N-by-N lattice.
  - Start in the middle.
  - Each step randomly moves to a neighboring intersection, if a previously moved intersections, no move.
  - Two possible outcomes: 
    - dead end and escape.

- Applications. Polymers (http://en.wikipedia.org/wiki/Polymer), statistical mechanics, etc.

Self-Avoiding Random Walk

```java
// read in lattice size N as command-line argument.
// read in number of trials T as command-line argument.
// repeat T times:
    // initialize (x, y) to center of N-by-N grid.
    // repeat as long as (x, y) is not escape or trap
        // mark (x, y) as visited.
        // take a random step, updating (x, y).
    // if not escape
        increase #deadEnds
// print fraction of dead ends.
```
% java SelfAvoidingWalks 10 100000
5% dead ends
% java SelfAvoidingWalks 20 100000
32% dead ends
% java SelfAvoidingWalks 30 100000
58% dead ends
...
% java SelfAvoidingWalks 100 100000
99% dead ends

Backup Slides
Array Variables are Reference Variables

```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]
```

Recap: PageRank

[Sergey Brin and Larry Page, 1998]

- Problem: many Web pages may contain the searched key word (e.g., Yale), how to rank the pages when displaying search results?

- Basic PageRank™ idea
  - 10-90 rule
    * 10% of the time surfer types a random page
    * 90% of the time surfer clicks random link on a given page
  - PageRank ranks pages according frequencies surfers visit Web pages
PageRank

Computing PageRank: Setup

- Number pages 0 to N - 1
- Obtain page links among pages
Computing PageRank

- Initialize arbitrary page ranks
- Iterative algorithm
  - Assume current round page rank of page \( p_i \) is \( PR_{cur}(p_i) \)
  - Update next round by distributing the frequencies of the previous round

\[
PR_{new}(x) = 0.1 \left( \frac{1}{N} \right) + 0.9 \sum_{i=1}^{n} \frac{PR_{pre}(p_i)}{C(p_i)}
\]
Input format

5
1*
2 2 3 3 4*
...

Outgoing adjacency List

PageRank: Reading Graph

Scanner input = new Scanner(new File("tiny-web.txt"));

// First read N, the number of pages
int N = Integer.parseInt(input.nextLine());

// An irregular 2D array to keep track of outgoing links
int[][] outgoingLinks = new int[N][];

// read in graph one line at a time
for (int i = 0; i < N; i++) {
    String line = input.nextLine(); // read outgoing links of i
    String[] links = line.split(" ");
    outgoingLinks[i] = new int[links.length];
    for (int j = 0; j < links.length; j++) {
        outgoingLinks[i][j] = Integer.parseInt(links[j]);
    }
}
PageRank: Compute Rank

double[] pr = new double[N];
pr[0] = 1; // initialize to assume start at web page 0
// or Arrays.fill(pr, 1.0 / N);

for (int t = 0; t < 20; t++) {
    double[] newpr = new double[N]; // init newpr
    Arrays.fill(newpr, 0.1 / N);

    // loop over the node to redistribute the frequencies
    for (int i = 0; i < N; i++) { // redistribute node i
        for (int j = 0; j < outgoingLinks[i].length; j++) {
            int to = outgoingLinks[i][j];
            newpr[to] += 0.9 * pr[i] / outgoingLinks[i].length;
        }
    }
    pr = newpr; // swap newpr to be pr
    System.out.printf("pr[%2d] = %s\n", t, Arrays.toString(pr));
}