Java Programming: Arrays
Announcements

HW07 is available on the web
  – Image processing in Java
  – Due next Monday, March 18th at 11:59:59pm

Pennstagram
Java arrays
Java Arrays: Indexing

- An array is a sequentially ordered collection of values that can be indexed in constant time.
- Index elements from 0

- Basic array expression forms
  
  \[ a[i] \]  
  access element of array \( a \) at index \( i \)

  \[ a[i] = e \]  
  assign \( e \) to element of array \( a \) at index \( i \)

  \[ a.length \]  
  get the number of elements in \( a \)
Java Arrays: Dynamic Creation

• Create an array `a` of size `n` with elements of type `C`:
  
  ```java
  C[] a = new C[n];
  ```

• Arrays are objects that live in the heap, values with array type are mutable references:

  ```java
  int[] a = new int[4];
  a[2] = 7;
  ```

Array entries are mutable.

Length is a `final` (immutable) field.
int[] myArray = { 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000};

String[] yourArray = { “foo”, “bar”, “baz” };

Point[] herArray = { new Point(1,3), new Point(5,4) };

herArray = new Point[] { new Point(2,3), new Point(6,5) };
Java Arrays: Aliasing

• Variables of array type are references and can be aliases

```java
int[] a = new int[4];
int[] b = a;
a[2] = 7;
System.out.println(b[2]);
```
Array Iteration
For loops

```
static double sum(double[] a) {
    double total = 0;
    for (int i = 0; i < a.length; i++) {
        total += a[i];
    }
    return total;
}
```

General pattern for computing info about an array
Multi-Dimensional Arrays

A 2-d array is just an array of arrays...

```java
String[][] names = {
    {"Mr. ", "Mrs. ", "Ms. "},
    {"Smith", "Jones"}
};

System.out.println(names[0][0] + names[1][0]);
// --> Mr. Smith
System.out.println(names[0][2] + names[1][1]);
// --> Ms. Jones
```

String[][] just means (String[])[[]]
names[1][1] just means (names[1])[1]

More brackets → more dimensions
int[][] products = new int[5][];
for(int col = 0; col < 5; col++) {
    products[col] = new int[col+1];
    for(int row = 0; row <= col; row++) {
        products[col][row] = col * row;
    }
}
```java
int[][] products = new int[5][];
for(int col = 0; col < 5; col++) {
    products[col] = new int[col+1];
    for(int row = 0; row <= col; row++) {
        products[col][row] = col * row;
    }
}
```

Note: This heap picture is simplified – it omits the class identifiers and length fields for all 6 of the arrays depicted.

(Contrast with the array shown earlier.)

Note also that orientation doesn’t matter.
Demo

ArrayExamples.java
Design Exercise: Resizable Arrays

Arrays that grow without bound.
Suppose you have a friend who runs a comic book store. She would like some help to keep track of her inventory. In each series, each issue has a number and she would like to know how many copies of that issue she has in stock. However, she does not know how many issues of each series there will be.
Step 1: Understand the Problem

- What are the requirements?
Step 2: Define the interface

What does a “resizable array" type need?

• a way to create an array without specifying an initial size
  – new ResArray();

• a way to access elements of the array
  – a.get(3) returns the number of copies of issue #3

• a way to update elements of the array
  – a.set(4,2) records the number of copies of issue #4

• a way to know the index of the most recent issue in stock
  – a.getExtent()

• a way to convert a resizable array to a regular array
  – a.values() returns an int[]
Demo: Steps 3 & 4

ResArray.java
ResArrayTest.java