Arrays
Chris Piech
CS106A, Stanford University
What does this say?

Puzzle in Gold Bug by Edgar Allan Poe
### Changing Variable Types

<table>
<thead>
<tr>
<th>int to double?</th>
<th>int to String?</th>
</tr>
</thead>
<tbody>
<tr>
<td>int x = 5;</td>
<td>int x = 5;</td>
</tr>
<tr>
<td>double xDb = x;</td>
<td>String xStr = &quot;&quot; + x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String to int?</th>
<th>String to double?</th>
</tr>
</thead>
<tbody>
<tr>
<td>String xStr = &quot;5&quot;;</td>
<td>String xStr = &quot;5.6&quot;;</td>
</tr>
<tr>
<td>int x = Integer.parseInt(x);</td>
<td>double x = Double.parseDouble(xStr);</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Casting double to int</th>
<th>GObject to GRect</th>
</tr>
</thead>
<tbody>
<tr>
<td>double x = 5.2;</td>
<td>GObject obj = getElementAt(5, 2);</td>
</tr>
<tr>
<td>int y = (int)x;</td>
<td>GRect objRect = (GRect)obj;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int to char</th>
</tr>
</thead>
<tbody>
<tr>
<td>int diff = 'C'-'A';</td>
</tr>
<tr>
<td>char next = (char)'a' + diff;</td>
</tr>
</tbody>
</table>
Changing Variable Types

```
0: ๐
1: ๑
2: ๒
3: ๓
4: ๔
5: ๕
6: ๖
7: ๗
8: ๘
9: ๙

Enter arabic number: ๒๒
Western arabic translation: 42
Enter arabic number: ๓๓๓๓๓
Western arabic translation: 999000
Enter arabic number: |
```
Number Translation

Brāhmī

Indien (Gwālior)

Sanskrit-Devanāgarī (Indien)

Arabe occidental (Gobār)

Arabe oriental

Xle siècle (Apices)

XVe siècle

XVe siècle (Dürer)
Where are we?

- Karel the Robot
- Java
- Console Programs
- Graphics Programs
- Text Processing
- **Data Structures**
- Defining our own Variable Types
- GUIs
A new variable type that is an object that represents an ordered, homogeneous list of data.

- Arrays have many *elements* that you can access using *indices*.
You can create arrays of any variable type. For example:

```java
double[] results = new double[5];
String[] names = new String[3];
boolean[] switches = new boolean[4];
GRect[] rects = new GRect[5];
```

- Java initializes each element of a new array to its default value, which is 0 for int and double, '\0' for char, false for boolean, and null for objects.
You can create arrays of any variable type. For example:

```java
char[] oldSchoolString = new char[5];
```

- Java initializes each element of a new array to its default value, which is 0 for int and double, ‘\0’ for char, false for boolean, and null for objects.
## Data Structures

<table>
<thead>
<tr>
<th>Operation</th>
<th>Strings</th>
<th>Arrays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make a new one</td>
<td>String str = “abc”;</td>
<td></td>
</tr>
<tr>
<td>Get length?</td>
<td>str.length()</td>
<td></td>
</tr>
<tr>
<td>Get element?</td>
<td>str.charAt(i)</td>
<td></td>
</tr>
<tr>
<td>Set element?</td>
<td>Not allowed</td>
<td></td>
</tr>
<tr>
<td>Loop?</td>
<td>for(int i = 0; i &lt; str.length(); i++)</td>
<td></td>
</tr>
</tbody>
</table>
## Data Structures

<table>
<thead>
<tr>
<th>Operation</th>
<th>Strings</th>
<th>Arrays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make a new one</td>
<td>String str = “abc”;</td>
<td>int arr = new int[5];</td>
</tr>
<tr>
<td>Get length?</td>
<td>str.length()</td>
<td>arr.length</td>
</tr>
<tr>
<td>Get element?</td>
<td>str.charAt(i)</td>
<td>arr[i]</td>
</tr>
<tr>
<td>Set element?</td>
<td>Not allowed</td>
<td>arr[i] = 5;</td>
</tr>
<tr>
<td>Loop?</td>
<td>for(int i = 0; i &lt; str.length(); i++)</td>
<td>for(int i = 0; i &lt; arr.length(); i++)</td>
</tr>
</tbody>
</table>
Creating Arrays

```java
type[] name = new type[length];

int[] numbers = new int[5];
```

Java automatically initializes elements to 0.
name[index]    // get element at index

- Like Strings, indices go from 0 to the array's length - 1.

```java
for (int i = 0; i < 7; i++) {
    println(numbers[i]);
}
println(numbers[9]);    // exception
println(numbers[-1]);   // exception
```

<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>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Putting Data In An Array

```plaintext
name[index] = value;  // set element at index
```
Setting values

name[index] = value;  // set element at index

• Like Strings, indices go from 0 to the array's length - 1.

```java
int[] numbers = new int[7];
for (int i = 0; i < 7; i++) {
    numbers[i] = i;
}
numbers[8] = 2;    // exception
numbers[-1] = 5;  // exception
```

<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>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Q: What are the contents of numbers after executing this code?

```java
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;
int x = numbers[1];
numbers[x] = 2;
numbers[numbers[4]] = 9;
```

// 0 1 2 3 4 5 6 7
A. {0, 3, 0, 2, 7, 0, 5, 9}
B. {0, 3, 0, 0, 7, 0, 5, 0}
C. {3, 3, 5, 2, 7, 4, 5, 0}
D. {0, 3, 0, 2, 7, 6, 4, 4}
Similar to a String, you can get the length of an array by saying

```
myArray.length
```

Note that there are no parentheses at the end!

**Practice:**

- What is the index of the *last element* of an array in terms of its length?
- What is the index of the *middle element* of an array in terms of its length?
Just like with Strings, we can use an array’s length, along with its indices, to perform cool operations.
Just like with Strings, we can use an array’s length, along with its indices, to perform cool operations. For instance, we can efficiently initialize arrays.

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

<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>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>
Just like with Strings, we can use an array’s length, along with its indices, to perform cool operations. For instance, we can read in numbers from the user:

```java
int length = readInt("# of numbers? ");
int[] numbers = new int[length];
for (int i = 0; i < numbers.length; i++) {
    numbers[i] = readInt("Elem " + i + ": ");
}
```
Just like with Strings, we can use an array’s length, along with its indices, to perform cool operations. Try it out! *sum up* all of an array’s elements.

```java
// assume that the user has created int[] numbers
int sum = 0;
for (int i = 0; i < numbers.length; i++) {
    sum += numbers[i];
}
println(sum);
```
Sometimes, we want to hardcode the elements of an array.

```java
int numbers = new int[7];
numbers[0] = 5;
numbers[1] = 32;
numbers[3] = 12;
...

// This is tedious!
```
Sometimes, we want to hardcode the elements of an array. Luckily, Java has a special syntax for initializing arrays to hardcoded numbers.

```java
type[] name = { elements };
```

```java
// Java infers the array length
int[] numbers = {5, 32, 12, 2, 1, -1, 9};
```
Limitations of Arrays

- An array’s length is **fixed**. You cannot resize an existing array:

  ```java
  int[] a = new int[4];
  a.length = 10; // error
  ```

- You cannot compare arrays with `==` or `equals`:

  ```java
  int[] a1 = {42, -7, 1, 15};
  int[] a2 = {42, -7, 1, 15};
  if (a1 == a2) { ... } // false!
  if (a1.equals(a2)) { ... } // false!
  ```

- An array does not know how to print itself:

  ```java
  println(a1); // [I@98f8c4]
  ```
The class Arrays in package java.util has useful methods for manipulating arrays:

<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrays.binarySearch(array, value)</td>
<td>returns the index of the given value in a sorted array (or &lt; 0 if not found)</td>
</tr>
<tr>
<td>Arrays.copyOf(array, length)</td>
<td>returns a new copy of array of given length</td>
</tr>
<tr>
<td>Arrays.equals(array1, array2)</td>
<td>returns true if the two arrays contain same elements in the same order</td>
</tr>
<tr>
<td>Arrays.fill(array, value);</td>
<td>sets every element to the given value</td>
</tr>
<tr>
<td>Arrays.sort(array);</td>
<td>arranges the elements into sorted order</td>
</tr>
</tbody>
</table>
| Arrays.toString(array)             | returns a string representing the array, such as 
                                  | "[10, 30, -25, 17]"                                                         |
Arrays.toString accepts an array as a parameter and returns a string representation of its elements.

```java
int[] e = {0, 2, 4, 6, 8};
println("e is " + Arrays.toString(e));
```

Output:
e is [0, 14, 4, 6, 8]
Arrays as Parameters

- Arrays are just another variable type, so methods can take arrays as parameters and return an array.

```java
private int sumArray(int[] numbers) {
    ...
}

private int[] makeSpecialArray(...) {
    ...
    return myArray;
}
```
• Arrays are just another variable type, so methods can take arrays as parameters and return an array.
• However, arrays are **objects**, so per *A Variable Origin Story*, an array variable box actually stores its *location*.
• This means changes to an array passed as a parameter affect the original array!
public void run() {
    int[] numbers = new int[7];
    fillArray(numbers);
    println(Arrays.toString(numbers));
}

private void fillArray(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        arr[i] = 2 * i;
    }
}
Let’s write a method called `swapElements` that swaps two elements of an array. How can we do this?

What parameters should it take (if any)? What should it return (if anything)?

```java
private ??? swapElements(????) {
    ...
}
```
public void run() {
    int[] array = new int[5];
    ...
    swapElements(array[0], array[1]);
    ...
}

private void swapElements(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}
public void run() {
    int[] array = new int[5];
 ...
    swapElements(array[0], array[1]);
...
}

private void swapElements(int x, int y) {
    int temp = x;
    x = y;
    x = temp;
}

Ints are primitives, so they are passed by value! Their variable boxes store their actual values. So changes to the parameter do not affect the original.
public void run() {
    int[] array = new int[5];
    ...
    swapElements(array, 0, 1);
    ...
}

private void swapElements(int[] arr, int pos1, int pos2) {
    int temp = arr[pos1];
    arr[pos1] = arr[pos2];
    arr[pos2] = temp;
}
public void run() {
    int[] array = new int[5];
    swapElements(array, 0, 1);
}

private void swapElements(int[] arr, int pos1, int pos2) {
    int temp = arr[pos1];
    arr[pos1] = arr[pos2];
    arr[pos2] = temp;
}

Arrays are objects, so they are passed by reference! Their variable boxes store their location. So changes to the parameter do affect the original.
public void run() {
    int n = readInt("Enter number of elements: ");
    int[] intArray = createIndexArray(n);
    println("Forward: " + arrayToString(intArray));
    reverseArray(intArray);
    println("Reverse: " + arrayToString(intArray));
}

private int[] createIndexArray(int n) {
    int[] array = new int[n];
    for (int i = 0; i < n; i++) {
        array[i] = i;
    }
    return array;
}

private String arrayToString(int[] array) {
    String str = "";
    for (int i = 0; i < array.length; i++) {
        if (i > 0) str += ", ";
        str += array[i];
    }
    return "[" + str + "]";
}

private void reverseArray(int[] array) {
    for (int i = 0; i < array.length / 2; i++) {
        swapElements(array, i, array.length - i - 1);
    }
}

private void swapElements(int[] array, int p1, int p2) {
    int temp = array[p1];
    array[p1] = array[p2];
    array[p2] = temp;
}

Example: Reverse Array Program

Enter number of elements: 10
Forward: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Reverse: [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
A cryptogram is a puzzle in which a message is encoded by replacing each letter in the original text with some other letter. The substitution pattern remains the same throughout the message. Your job in solving a cryptogram is to figure out this correspondence.

One of the most famous cryptograms was written by Edgar Allan Poe in his short story “The Gold Bug.”

In this story, Poe describes the technique of assuming that the most common letters in the coded message correspond to the most common letters in English, which are E, T, A, O, I, N, S, H, R, D, L, and U.
### Letter Frequency

<table>
<thead>
<tr>
<th>LETTER</th>
<th>COUNT</th>
<th>PERCENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>445.2</td>
<td>12.49%</td>
</tr>
<tr>
<td>T</td>
<td>330.5</td>
<td>9.28%</td>
</tr>
<tr>
<td>A</td>
<td>286.5</td>
<td>8.04%</td>
</tr>
<tr>
<td>O</td>
<td>272.3</td>
<td>7.64%</td>
</tr>
<tr>
<td>I</td>
<td>269.7</td>
<td>7.57%</td>
</tr>
<tr>
<td>N</td>
<td>257.8</td>
<td>7.23%</td>
</tr>
<tr>
<td>S</td>
<td>232.1</td>
<td>6.51%</td>
</tr>
<tr>
<td>R</td>
<td>223.8</td>
<td>6.28%</td>
</tr>
<tr>
<td>H</td>
<td>180.1</td>
<td>5.05%</td>
</tr>
<tr>
<td>L</td>
<td>145.0</td>
<td>4.07%</td>
</tr>
<tr>
<td>D</td>
<td>136.0</td>
<td>3.82%</td>
</tr>
<tr>
<td>C</td>
<td>119.2</td>
<td>3.34%</td>
</tr>
<tr>
<td>U</td>
<td>97.3</td>
<td>2.73%</td>
</tr>
<tr>
<td>M</td>
<td>89.5</td>
<td>2.51%</td>
</tr>
<tr>
<td>F</td>
<td>85.6</td>
<td>2.40%</td>
</tr>
<tr>
<td>P</td>
<td>76.1</td>
<td>2.14%</td>
</tr>
<tr>
<td>G</td>
<td>66.6</td>
<td>1.87%</td>
</tr>
<tr>
<td>W</td>
<td>59.7</td>
<td>1.68%</td>
</tr>
<tr>
<td>Y</td>
<td>59.3</td>
<td>1.66%</td>
</tr>
<tr>
<td>B</td>
<td>52.9</td>
<td>1.48%</td>
</tr>
<tr>
<td>V</td>
<td>37.5</td>
<td>1.05%</td>
</tr>
<tr>
<td>K</td>
<td>19.3</td>
<td>0.54%</td>
</tr>
<tr>
<td>X</td>
<td>8.4</td>
<td>0.23%</td>
</tr>
<tr>
<td>J</td>
<td>5.7</td>
<td>0.16%</td>
</tr>
<tr>
<td>Q</td>
<td>4.3</td>
<td>0.12%</td>
</tr>
<tr>
<td>Z</td>
<td>3.2</td>
<td>0.09%</td>
</tr>
</tbody>
</table>

By Peter Norvig  
Head of Google Research  
Former CS221 Instructor  

Based on 3.8 trillion letters
Poe's Cryptographic Puzzle

53‡‡305) 6*; 4826) 4‡•) 4‡) 806*; 48†8¶60) 85; 1‡(; ‡*8‡83(88) 5*†; 46(88*96*?
; 8)*‡(; 485); 5*‡2:*‡(; 4956*2(5*4) 8¶8*; 4069285);) 6‡8) 4‡‡;1(9; 48081; 8:8‡
1; 48†85; 4) 485‡528806*81(9; 48;(88;4(‡?34; 48) 4‡;161;188‡?;8

ACFOOCDOASS6NTHEBESPOPSHOEOTSHONTHEBTY
6DS$8R4FRTYONEDEGRERSANDTHETRHN6MN
UTSH$ORTHEASTANDBAYOHRTM36NBRANCHSBY
$NH0GRBASTSHODRSH$HTOHTHORRETRYFH
THRBHAT$HRADBSROBOMFONHTHTKR
OUCHTHESHOTTE6TYERTOUT

8 33
; 26
4 19
‡ 16
) 16
* 13
5 12
6 11
( 10
† 8
1 8
0 6
9 5
2 5
: 4
3 4
? 3
¶ 2
- 1
• 1
The basic idea behind the program to count letter frequencies is to use an array with 26 elements to keep track of how many times each letter appears. As the program reads the text, it increments the array element that corresponds to each letter.

TWAS BRILLIG

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
```

---

Piech, CS106A, Stanford University
To the code!