Outline

- Problem: How can I represent groups of objects in an array

- Previously considered arrays of primitives

- This can get complicated
  - How many references are there to objects?

- Arrays as parameters
  - Arrays can be used as input and return values from methods
One dimensional Array

° We have introduced one-dimensional array in our previous lectures.
° An array is a collection of variables of the same type, referred by a common name.
° type array-name [] = new type[size];
° int age[] = new int[5]; //declaration
° age[0] = 20; //assignment

Definitions Using Array Literals

° Used when exact size and initial values of an array are known in advance
  • used to initialize the array handle

```java
int[] count = { 5,6,3,10 };
```

Visualize the results of the above command

<table>
<thead>
<tr>
<th>count</th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>10</td>
</tr>
</tbody>
</table>
Contrast Definitions

Elements are primitive values (0s in this case)

Elements are objects

Arrays of Objects

- The elements of an array can be object references
- The following declaration reserves space to store 5 references to String objects
  
  String[] words = new String[5];
- It does NOT create the String objects themselves
- Initially an array of objects holds null references
- Each object stored in an array must be instantiated separately
Arrays of Objects

- The words array when initially declared:

  words

- At this point, the following reference would throw a NullPointerException:

  System.out.println (words[0]);

- After some String objects are created and stored in the array:

  words
Arrays of Objects

- Keep in mind that String objects can be created using literals
- Following declaration creates an array object called verbs
  - Fills it with four String objects created using string literals

```java
String[] verbs = {"play", "work", "eat", "sleep"};
```
These are referenced by verbs[0] through verbs[3].

Definitions Using Array Literals

- Create an array of strings

```java
String[] names = {"Bashful", "Doc", "Sneezy"};
```

- Table:

<table>
<thead>
<tr>
<th>names</th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bashful</td>
<td>Doc</td>
<td>Sneezy</td>
</tr>
</tbody>
</table>
Arrays as Objects

- In Java, an array is an object. If the type of its elements is `anyType`, the type of the array object is `anyType[]`.
- There are two ways to declare an array:

```
anyType [] arrName;
```

or

```
anyType arrName [];
```

The difference becomes significant only when several variables are declared in one statement:

```
int [] a, b; // both a, b are arrays
int a [], b; // a is an array, b is not
```

Arrays as Objects

- As with other objects, the declaration creates only a reference, initially set to `null`.
  - An array must be created before it can be used.
- There are two ways to create an array:

```
arrName = new anyType [ length ];
```

Brackets, not parens!

or

```
arrName = new anyType [ ] { val1, val2, ..., valN };
```
Array's Length

- The length of an array is determined when that array is created.
- The length is either given explicitly or comes from the length of the {...} initialization list.
- The length of an array `arrName` is referred to in the code as `arrName.length`.
- `length` appears like a public field (not a method) in an array object.

Arrays as Parameters

- An entire array can be passed as a parameter to a method.
- The reference to the array is passed,
  - Makes the formal and actual parameters aliases of each other
- Changing an array element within the method changes the original
- An individual array element can be passed to a method as well,
  - The type of the formal parameter is the same as the element type.
  - In this case, the call is 'by value.'
Passing to Methods

° Example:

```java
/**
*   Swaps a[i] and a[j]
*/
public void swap (int a[], int i, int j)
{
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}
```

Returning Arrays from Methods

° As for other objects, an array can be returned from a method.
° The returned array is usually constructed within the method or obtained from calls to other methods.
° The return type of a method that returns an array with someType elements is designated as someType[].
Returning from Methods

° Example:

```java
public double[] solveQuadratic(double a, double b, double c)
{
    double d = b * b - 4 * a * c;
    if (d < 0) return null;
    d = Math.sqrt(d);
    double roots[] = new double[2];
    roots[0] = (-b - d) / (2 * a);
    roots[1] = (-b + d) / (2 * a);
    return roots;
}
```

Or simply:

```java
return new double[] { (-b - d) / (2 * a), (-b + d) / (2 * a) };```

Command-Line Arguments

° The signature of the `main` method indicates that it takes an array of `String` objects 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 three `String` objects into `main`:

> java StateEval pennsylvania texas arizona

° These strings are stored at indexes 0-2 of the array parameter of the `main` method
Summary

° We can now make complicated data structures
  • Objects still the basic units of data storage

° Arrays are fundamental
  • Most data is stored in arrays
  • Allows for easy data access

° Command line arguments are a good example of an array of strings

° More to come: two dimensional arrays