Computer Science I

Chapter 5 Notes

Arrays

- Arrays are objects that help us organize large amounts of information
- You have already seen an array used in the Java programs in this class:

```
public static void main (String [ ] args)
```

args is an array of String references

Arrays

- An *array* is an ordered list of values

The entire array has a single name

Each value has a numeric *index*

	0	1	2	3	4	5	6	7	8	9
`scores`	79	87	94	82	67	98	87	81	74	91

An array of size N is indexed from 0 to N – 1.

This array holds 10 values that are indexed from 0 to 9.

Arrays

- A particular value in an array is referenced using the array name followed by the index in brackets

	0	1	2	3	4	5	6	7	8	9
`scores`	79	87	94	82	67	98	87	81	74	91

For example, the expression

```
scores[2]
```

refers to the value `94`.

Arrays

- An array element can be assigned a value, printed, or used in a calculation :

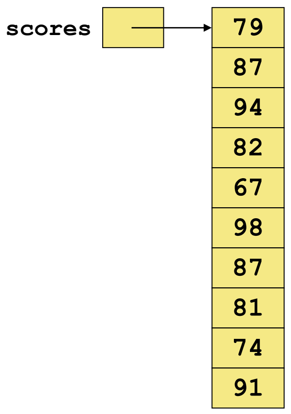
```
scores[2] = 89;

scores[first] = scores[first] + 2;

mean = (scores[0] + scores[1])/2;

System.out.println ("Top = " + scores[5]);
```

Arrays

- The values held in an array are called *array elements.*
- An array stores multiple values of the same type – the *element type.*
- The element type can be a primitive type or an object reference.
- In Java, the array itself is an object that must be instantiated.
- Array index values start at zero.

Arrays

- Another way to depict the `scores` array:



Declaring Arrays

- The `scores` array could be declared as follows:

```
int[] scores = new int[10];
```

- The type of the variable `scores` is `int[]` (an array of integers).
- Note that the array type does not specify its size, but each object of that type has a specific size.
- The right side of the assignment "new int[10]" creates space in memory for 10 array elements.

7-8

Declaring Arrays

- Some other examples of array declarations:

```
float prices[] = new float[500];
boolean[] flags;
flags = new boolean[20];
char[] codes = new char[1750];
```

Note: Java will allow the [ ] to be placed either before the variable name or after. Other programming languages, such as C or C++, require the [ ] to be placed after the variable name.

Bounds Checking

- Once an array is created, it has a fixed size.
- An index used in an array reference must specify a valid element.
- That is, the index value must be in range 0 to N-1 (where N is the number of elements in the array).
- The Java interpreter throws an

  `ArrayIndexOutOfBoundsException`

  if an array index is out of bounds (less than 0 or greater than N – 1).
- This is called automatic *bounds checking.*

Bounds Checking

- For example, if the array codes can hold 100 values, it can be indexed using only the numbers 0 to 99.
- If the value of count is 100, then the following reference will cause an exception to be thrown:

```
System.out.println (codes[count]);
```

- A common error occurs when a loop repeats too many times.

problem

```
for (int index = 0; index <= 100; index++)
    codes[index] = index * 50 + epsilon;
```

Bounds Checking

- Each array object has a public variable called `length` that stores the size of the array.
- It is referenced using the array name. For example,

```
scores.length
```

- Note that `length` holds the number of elements, not the largest index.

Alternate Array Syntax

- In Java, the following two declarations are equivalent:

```
float[] prices;
float prices[];
```

- The instructor prefers the second format because of its compatibility with other languages, such as C and C++.

Initializer Lists

- An *initializer list* can be used to instantiate and fill an array in one step.
- The values are surrounded by braces and separated by commas.
- Examples:

```
int units[ ] = {147, 323, 89, 933, 540};
char[] letterGrades = {'A', 'B', 'C', 'D', 'F'};
```

Initializer Lists

- Note that when an initializer list is used:
  - the `new` operator is not used
  - no array size is specified
- The size of the array is determined by the number of items in the initializer list.
- An initializer list can be used only in the array declaration.

Arrays of Objects

- The elements of an array can be object references.
- The following declaration reserves space to store 5 references to `String` objects:

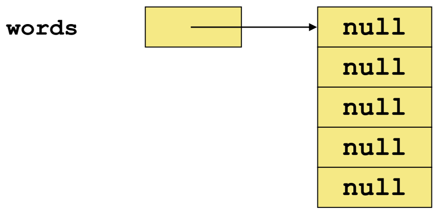
```
String[] words = new String[5];
```

- 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

```
String[] words = new String[5];
```

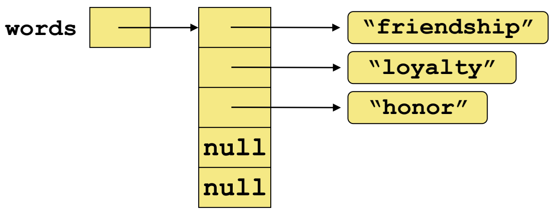
When it is first declared, the `words` array looks like this:



Arrays of Objects

- After the following assignments are executed, `String` objects are created and their references are stored in the array.

```
words[0] = "friendship";
words[1] = "loyalty";
words[2] = "honor";
```



Arrays of Objects

- Keep in mind that `String` objects can be created using literals (a group of characters in quotation marks).
- The following declaration creates an array object called `verbs` and fills it with four `String` objects created using string literals.

```
String verbs[ ] = {"play", "work", "eat", "sleep"};
```

Arrays of Objects

- The array does not contain the actual objects.
- The array contains references (memory addresses) to where the objects are stored.

7-
20