Strings
Strings

- A String is an object.
  - Variables can hold a reference.

- A String is a sequence of characters.
  - Immutable (cannot be changed).
  - Methods return new strings; they do not modify the existing string.

- StringBuffer class is mutable.
String Operators

z + (concatenate)
z += (append)
String Methods
(see Eckel, pp. 1063-1065)

- `length()`
- `charAt()`
- `equals()`
- `equalsIgnoreCase()`
- `indexOf`
- `substring()`
- `replace()`
- `toLowerCase()`
- `toUpperCase()`
- `trim()`
String Conversions

“Whenever the compiler expects a String object and it hasn’t got one, it will automatically call the toString() method that’s defined in Object and can be overridden by any Java class.”

– Eckel, p. 493
Interfaces
Interfaces

- An interface provides form, not implementation.
- Looks a lot like a class definition.
- Define someInterface with
  ```java
  interface someInterface { ... }
  ```
- Use with
  ```java
  class someClass implements someInterface { ...
  ```
 Interfaces II

z OK to have:
  y Method signatures
    x Name
    x Argument list
    x Return type
  y Data of primitive type

z Not OK to have:
  y Method bodies (including constructors)
  y Object data
Why Interfaces

z A way to provide the benefits of multiple inheritance
  y But with much less overhead than true multiple inheritance as in C++.
    x e.g. name space collisions, constructors
  y Lets you upcast an object to more than one base type.

z Can substitute for a pure abstract class.
Why Interfaces II

A class can inherit from one superclass and implement one or more interfaces.

```java
class someClass extends someOtherClass implements someInterface, someOtherInterface, aThirdInterface { … }
```

Provides consistency.

See Adventure.java in Chapter 8 for an example.
Arrays
Arrays

- Just one way Java keeps track of collections of objects.
- We will learn more about containers.
Arrays

- A linear collection of primitives or objects.
- What’s in the array?
  - An array of primitives holds the values.
  - An array of objects holds the references (handles).
- Java enforces type checking at compile- and run-time.
Array Features

z Size is fixed once allocated.
  y Use either
    x new class[size]
    x new[size] class

z An array is itself an object.
  y Store its reference in a variable with type “array of particular type of object.”
    x class varName[]
  y Assigning an array copies the reference (handle).

z Arrays are efficient.
Using Arrays

z Create like this:

y Weeble[] b = new Weeble[3];

y Creates a new array (object) which can hold 3 Weebles (objects), and stores the reference to this array in the variable b.

z Access like this:

y b[0] = new Weeble();

y System.out.println(b[0]);
Array of Weebles

A variable which holds a reference to an array of references to Weebles

Some Weebles

An array of references to Weebles

Weeble[] b

[0]
[1]
[2]
Output of Arrays Example (MyArrays.java)

About to declare array a.

a[0]=null [Constructor not invoked]
a[1]=null

About to dynamically allocate Weebles in a.
Weeble constructor
Weeble constructor

a[0]=Weeble@71aa3045
a[1]=Weeble@71a63045
More Output

About to declare array b.
Weeble constructor  [Constructor invoked]
Weeble constructor
Weeble constructor
b[0]=Weeble@715a3045
b[1]=Weeble@71563045
b[2]=Weeble@739e3045
Final Output

About to create c and copy a into it.
c[0]=Weeble@71aa3045  [Same as a]
c[1]=Weeble@71a63045

About to modify c[1].
Weeble constructor
c[0]=Weeble@71aa3045

c[1]=Weeble@73863045  [This changed]
a[0]=Weeble@71aa3045
a[1]=Weeble@73863045  [So did this]
What We Learned

- Declaring an array with
  \[
  \text{Weeble[]} \ a = \text{new Weeble}[2];
  \]
  does not invoke the constructor.

- Declaring an array with
  \[
  \text{Weeble[]} \ b = \{ \text{new Weeble}(), \text{new Weeble}(), \text{new Weeble}() \};
  \]
  does invoke the constructor.

- Assigning an array results in two variables referring to the same array.
Next Time

- Containers
- Casting