ARRAYS AND ARRAYLISTS

Chapter 7

Array

- Sequence of values of the same type
  - Primitive types
  - Objects
- Create an Array
  - double[] values = new double[10]
  - int[] values = {2,4,5,6,7,8,9}
  - BankAccount[] accounts=new BankAccount[10]

Definitions

- Length of array
  - Number of declared elements
  - Used or unused
- Element type
  - Type of the array
- Index
  - Access to the array
  - Integer

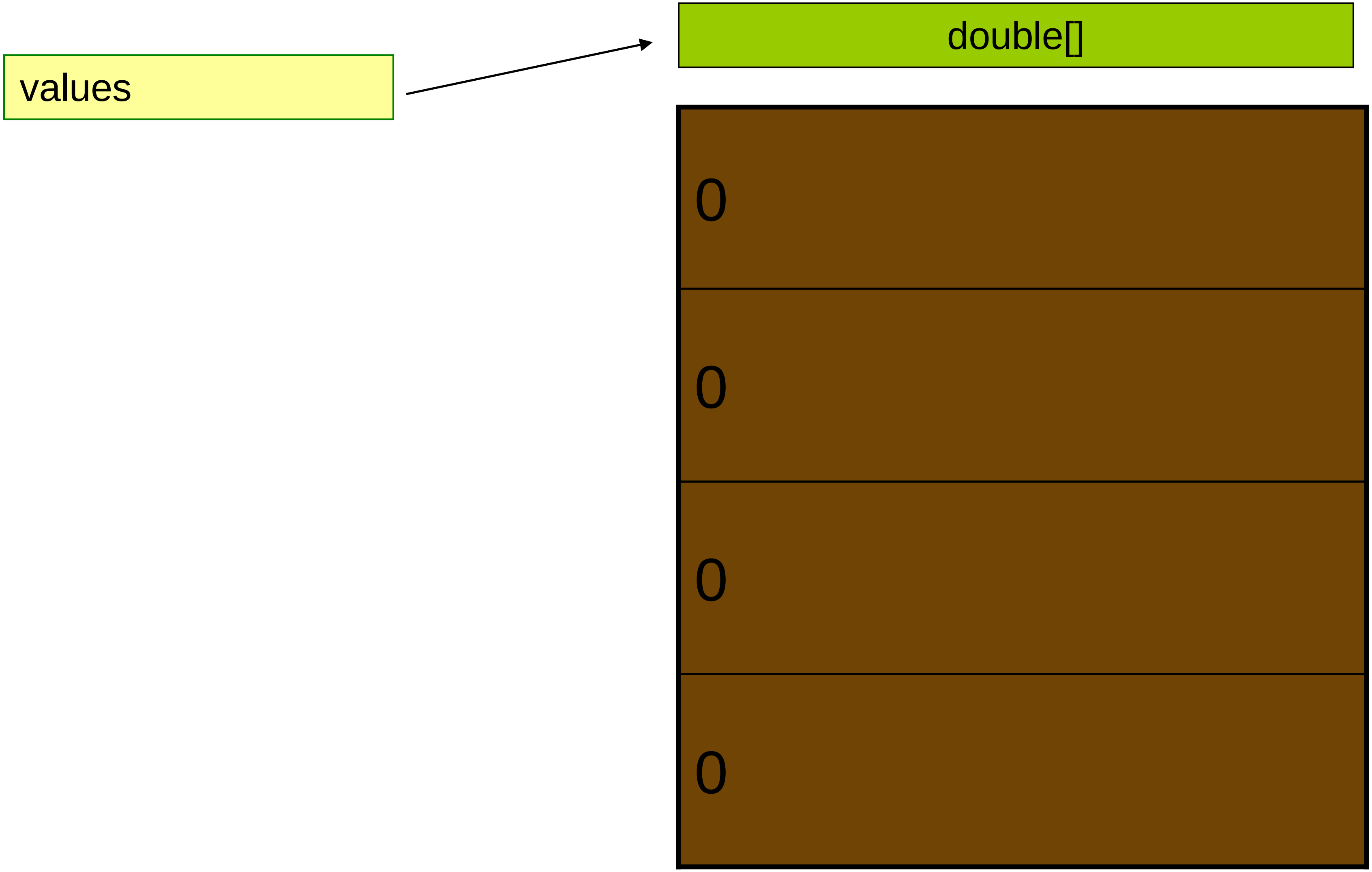
Differences Between Java and Visual Logic

- Visual Logic
  - Do not have to define type of array
  - Use ( ) to surround index number
- Java
  - Have to define type of array
  - Must use the new operator when creating the array
  - Use [ ] to surround index number

Default Initialization of Array Elements

- Array of numbers (int or double) = 0
- Array of boolean = false
- Array of objects = null

Arrays and Memory



values

double[]

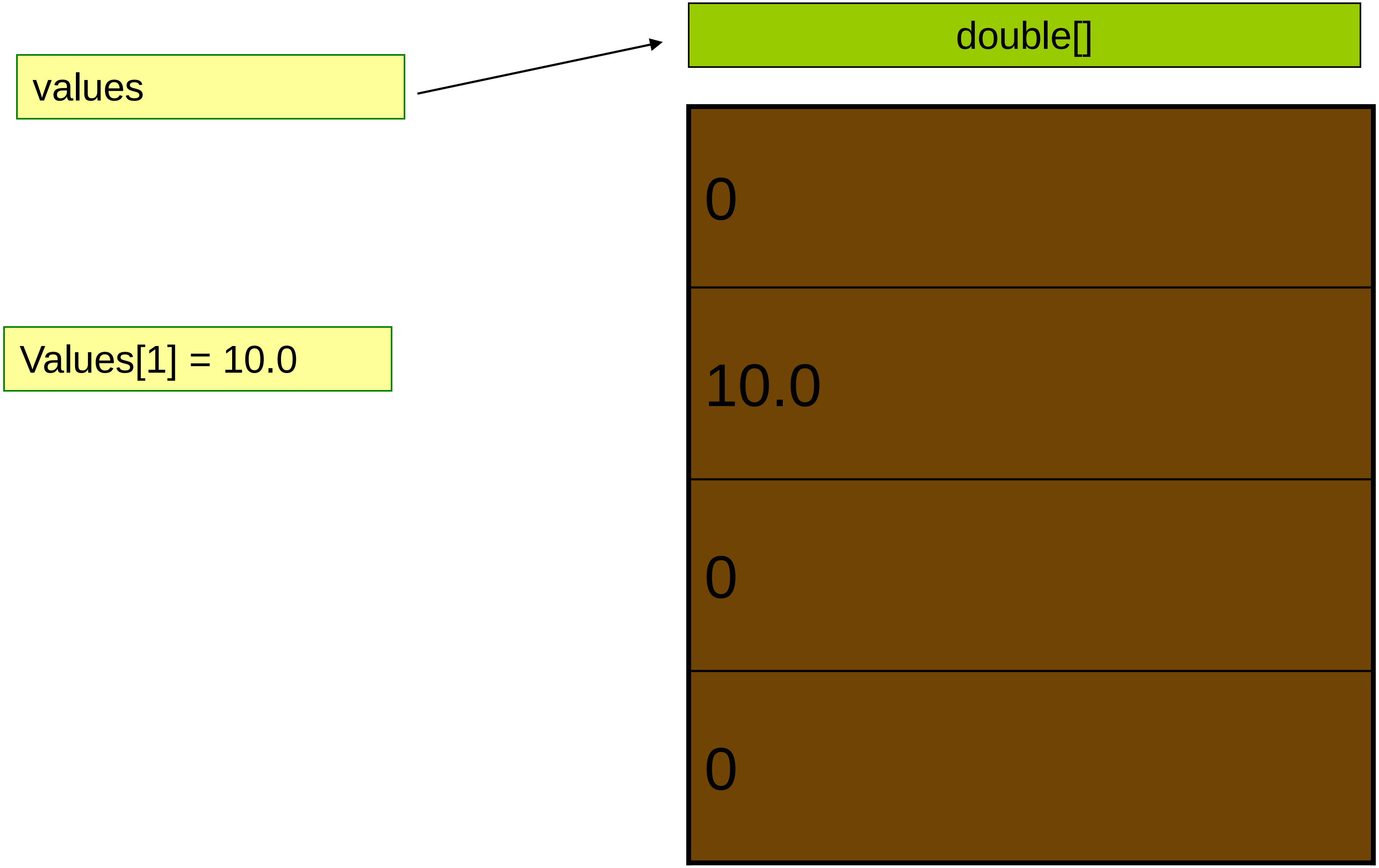
0

0

0

0

Add Value



values

double[]

Values[1] = 10.0

0

10.0

0

0

More Definitions

- Index values – range from 0 to length-1
- Bounds error
  - Accessing a non existent elements
  - Program terminate
- values.length() – method to get the length of the array named values
- Parallel arrays
  - 2 or more arrays used to describe one thing

Parallel Arrays

- Student name
- Student age
- Student gpa

- String[] name = new String[10]
- int [] age = new int[10]
- double [] gpa = new double[10]

- Avoid change to array of object

Major Problem With Array

- Length is fixed
- Array can develop “holes in delete” or “add”
  - Won’t know if array is full

Array List

- Allows you to collect objects just like arrays.
- Can grow and shrink as needed
- Has methods for inserting and deleting objects.
- Will not work on primitive types

ArrayList / Generic Class

- ArrayList<String> names = new ArrayList<String>();
- Notice the type of objects are in <>.
- These are called generics.
- Generics are used when you want anytype in its place.
- Will study later. Maybe next semester.

How To Use Array Lists

names.add(“Kathy”);	Add elements to end
System.out.println(names)	Prints [Kathy]
names.add(1,”Bob”)	Inserts Bob before Kathy
names.remove(0)	removes first element - Bob
names.set(0,”Bill”)	removes Kathy puts Bill in Kathy’s place
String name = names.get(0)	gets the first element
String name = namew.get(names.size()-1)	gets last element

Wrapper Classes

- The object class for a corresponding primitive type
- Can convert from primitive to wrapper
- Can store Wrapper in ArrayList
- Convert int to Integer
- Use Array List of type Integer

Primitive	Wrapper
byte	Byte
boolean	Boolean
char	Character
double	Double
float	Float
int	Integer
long	Long
short	Short

Converting From Primitive to Wrapper

- Converting from primitive to Wrapper Class is called “auto-boxing”
  - Double d = 29.95
- Converting from Wrapper Class to primitive is called “auto-unboxing”
  - double dd = d;
- Can still do arithmetic
  - Double dPlus = d +1;
  - d was unboxed. One was added. The result was boxed and placed in dPlus.

Enhanced for Loop

- Shortcut
- Traverses all elements of a collection

```
double [] values = ..............;
double sum = 0;
for (double element : values)
{
   sum = sum+ element;
}
```

- Loop variable contains an element not index.

Partially Filled Array

- arrayName.length() gives number of elements
- Does not give how many are used
- Keep a companion value to track how many elements are used.

Removing an Element

- Remove the 4th element of eight
- Array List
  - Use the remove method
  - Necessary shifts will take place 5th will move to 4th, and previous 6th to 5th etc.
  - You do nothing
- Array
  - You have to do all the necessary shifts

Inserting An Element

- Array List
  - If order doesn’t matter simply use
    - arrayListName.add(element)
  - If order does matter use
    - arrayListName.add(position, element)
- Array
  - If order doesn’t matter
    - use index of next available opening
  - If order does matter
    - must shift to create opening

Copying an Array

- An array variable stores a reference to the array.
- Copying yields a second reference to the same array.
- to create a true copy use copyOf

Copying and Growing an Array

```
int[] value = new int[10];
int valueSize = 0;
while (in.hasNextDouble())
{
  if (valuesSize == values.length)
        values =
        Arrays.copyOf(values,2*values.length);
  value[valueSize] = in.nextDouble();
  valuesSize++;
}
```

Multiple-Dimensional Arrays

- 2 Dimensions
  - String[][] board = new String[rows, columns]
  - rows and columns = some values
- 3 Dimensions
  - String[][][] board = new String[2][3][4]