Lecture 7 — Objects and Arrays

CITS2005 Object Oriented Programming

Department of Computer Science and Software Engineering
University of Western Australia

Contents

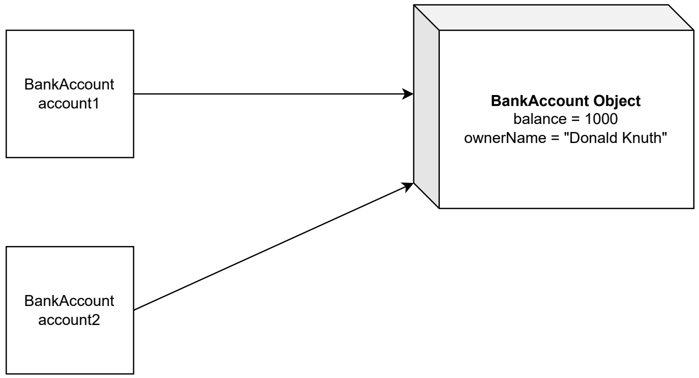
- See Chapters 4 and 5 of the textbook
- Objects and garbage collection
- Arrays
- Multidimensional arrays
- For loops with arrays

How Objects are Stored

```java
public class BankExample3 {
    public static void main(String[] args) {
        BankAccount account1, account2;
        account1 = new BankAccount();
        account2 = account1; // Same object
        account1.ownerName = "Donald Knuth";
        account1.balance = 1000;
    }
}
```

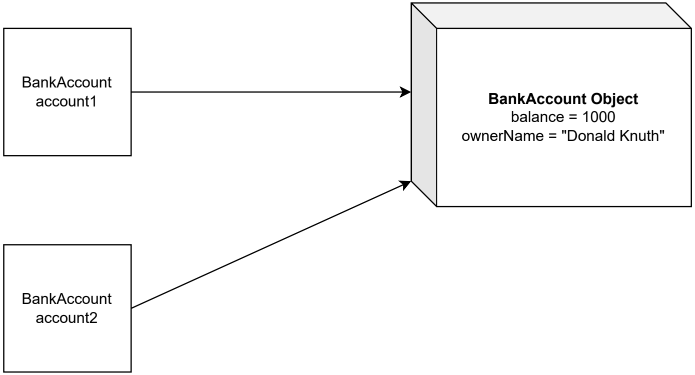
- Consider this example from the previous lecture
- How are `account1` and `account2` stored?

How Objects are Stored



- Non-primitive variables store references
- Variables storing primitive types (e.g., `int`, `char`) store them directly for efficiency
- This why the default values for a variable storing an object is `null`, but it is 0 or `false` for primitive types

Garbage Collection



- Variables disappear when their *scope* ends
- What happens to an object when no more variables reference it?
- *Garbage collection*

Garbage Collection

- In languages like C and C++, you need to manage memory yourself
- In Java, objects get automatically deleted and their memory is reclaimed
- The *garbage collector* runs in the background checking for objects with no references
- If you used `new` to create something, the garbage collector will keep track of it
- Note that this means primitive types are not garbage collected
- The are stored directly "inside" the variables
- Once the variable is out of scope, their memory is reclaimed

Arrays

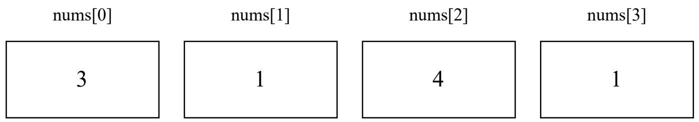
- Almost all objects in Java are defined by a `class`
- Arrays are the exception
- They are a special type that is an object, but not defined by a class
- They group a collection of variables together
- `int[] example = new int[10];`
- In general: `type[] array-name = new type[size]`
- Warning: our textbook sometimes uses `int example[] = new int[10]` syntax!
- Since they are implemented as special objects in Java, they are garbage collected and passed by reference

Indexing Arrays

```
public class Nums {
    public static void main(String[] args) {
        int[] nums = new int[4];
        nums[0] = 3;
        nums[1] = 1;
        nums[2] = 4;
        nums[3] = 1;
        for (int num : nums) // For each element in nums
            System.out.println(num);
    }
}
```

- The elements of an array can be accessed using square brackets: `array-name[index]`
- An array of length n is indexed using by $0, 1, \ldots, n-1$
- This is called *zero indexing*

How Objects are Stored



- A diagram for the previous program's array

Arrays and args

```
public class HelloCITS3 {
    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++) {
            System.out.println("args[" + i + "] is " + args[i]);
        }
    }
}
```

- This is from Lecture 1
- `args` is an array of `Strings`
- Notice that the elements are indexed from `0` to `args.length-1`
- Indexing an array out of this range will cause an error

Histogram

- Lets try to put together what we have learned into a Histogram program
- Use `Scanner` to read in numbers from 1 to 10
- Create a frequency histogram of the numbers entered
- An array is used to store the histogram
- Print them to the terminal
- Note that we will need to use a new method of `Scanner` called `hasNextInt()`
- We will also see `System.out.print()` (not `println()`)

Histogram

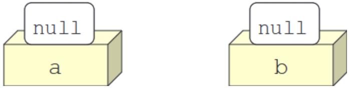
```java
import java.util.Scanner;

public class Histogram {
    public static void main(String[] args) {
        int[] histogram = new int[11]; // Create the array. Note that the size is 11, not 10
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter numbers between 1 and 10.");
        while (sc.hasNextInt()) { // Loop until the user enters something that is not an integer
            int n = sc.nextInt();
            if (n < 1 || n > 10) {
                System.out.println("Invalid number: " + n + ". Enter a number between 1 and 10.");
                continue; // Skip invalid numbers
            }
            histogram[n]++; // Increment the counter for the number
        }
        for (int i = 1; i < histogram.length; i++) { // Goes from 1 to 10
            System.out.print(i + ": ");
            for (int j = 0; j < histogram[i]; j++)
                System.out.print("*"); // .print() is used instead of .println() to print on the same line
            System.out.println(); // End the line
        }
    }
}
```

Arrays and Objects

- Arrays are stored by *reference* like objects
- They are a kind of object (e.g., `new`)
- This means two array variables can point to the same array

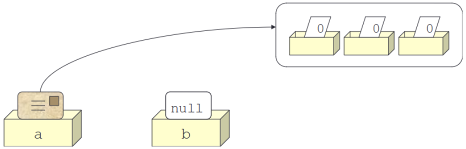
Arrays and Objects

- `int[] a, b;`



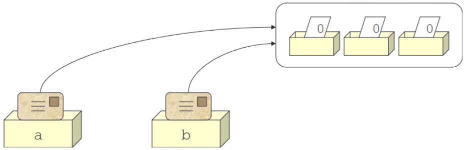
Arrays and Objects

- `int[] a, b;`
- `a = new int[3];`



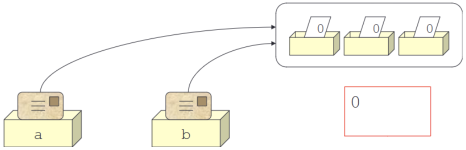
Arrays and Objects

- `int[] a, b;`
- `a = new int[3];`
- `b = a;`



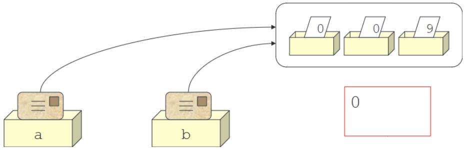
Arrays and Objects

- `int[] a, b;`
- `a = new int[3];`
- `b = a;`
- `System.out.println(b[2]);`



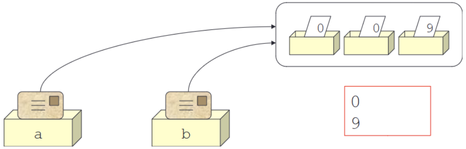
Arrays and Objects

- `int[] a, b;`
- `a = new int[3];`
- `b = a;`
- `System.out.println(b[2]);`
- `a[2] = 9;`



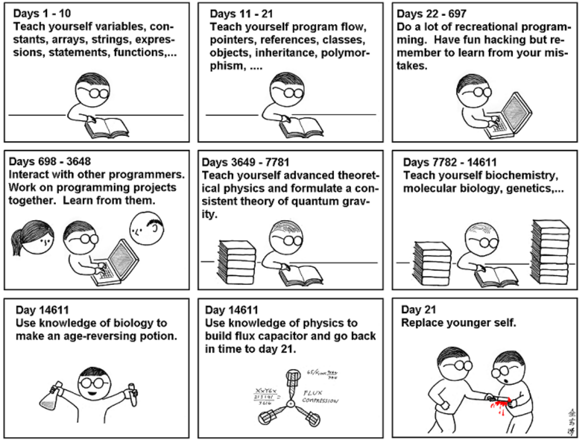
Arrays and Objects

- `int[] a, b;`
- `a = new int[3];`
- `b = a;`
- `System.out.println(b[2]);`
- `a[2] = 9;`
- `System.out.println(b[2]);`



```
0
9
```

Mid-lecture break


Days 1 - 10
Teach yourself variables, constants, arrays, strings, expressions, statements, functions,...
Days 11 - 21
Teach yourself program flow, pointers, references, classes, objects, inheritance, polymorphism, ...
Days 22 - 697
Do a lot of recreational programming. Have fun hacking but remember to learn from your mistakes.
Days 698 - 3648
Interact with other programmers. Work on programming projects together. Learn from them.
Days 3649 - 7781
Teach yourself advanced theoretical physics and formulate a consistent theory of quantum gravity.
Days 7782 - 14611
Teach yourself biochemistry, molecular biology, genetics,...
Day 14611
Use knowledge of biology to make an age-reversing potion.
Day 14611
Use knowledge of physics to build flux capacitor and go back in time to day 21.
FLUX COMPRESSION
Day 21
Replace younger self.

As far as I know, this is the easiest way to "Teach Yourself C++ in 21 Days".

Multidimensional Arrays

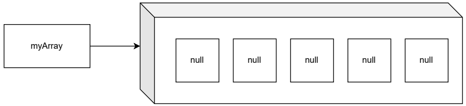
- Java allows multidimensional arrays
- `int[][] table = new int[3][4];`
- This is a 2D array—analogous to a spreadsheet with 3 rows and 4 columns
- `type[][] name = new type[x][y];`

Multidimensional Arrays

- More dimensions are possible
- `String[][][] threeDee = new String[5][5][5];`
- Arrays can be initialised with regular dimensions: `new type[x][y]`
- Or they can be initialised with irregular dimensions by leaving out a suffix of dimensions: `new type[x][]`
- These are called “ragged” or “irregular” arrays

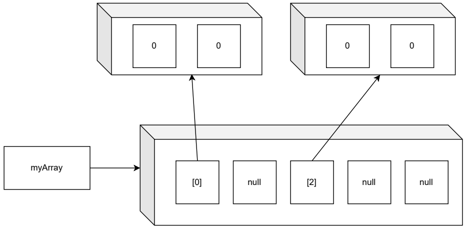
Multidimensional Arrays

- Multidimensional arrays are actually arrays of arrays
- `int[][] myArray = new int[5][];`



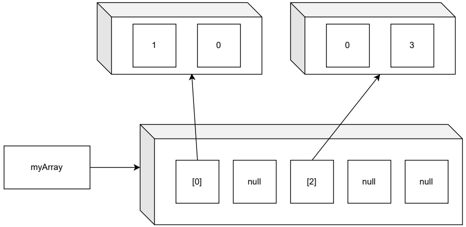
Multidimensional Arrays

- Multidimensional arrays are actually arrays of arrays
- `int[][] myArray = new int[5][];`
- `myArray[0] = new int[2];`
- `myArray[2] = new int[2];`



Multidimensional Arrays

- Multidimensional arrays are actually arrays of arrays
- `int[][] myArray = new int[5][];`
- `myArray[0] = new int[2];`
- `myArray[2] = new int[2];`
- `myArray[0][0] = 1;`
- `myArray[2][1] = 3;`



Multidimensional Arrays

```
public class Arrays2d {
    public static void main(String[] args) {
        int [][] table = new int [3][4];
        System.out.println(table.length);
        System.out.println(table[0].length);
        System.out.println(table[1][2]) ; // Numeric types are initialized to 0
        String [] names;
        names = new String[3];
        System.out.println(names[0] == null); // Strings are initialized to null
        names[0] = "Alice";
        System.out.println(names[0]);
        char [][][] grid = new char [2][][];
        System.out.println(grid[0] == null); // Each array is initialized to null
        grid[0] = new char[4][3];
        grid[1] = new char[2][1];
        System.out.println(grid[0][0][0]) ; // chars are initialized to \u0000 ((int) 0)
    }
}
```

Array Initialisation

- Arrays can be initialised with specific values using a special syntax
- `String[] names = {"Donald", "Alan"};`
- `int[][] myArray = {{1,2},{3,4,5}};`

Arrays and `for`

- Arrays and `for` loops go well together
- A `for` loop can be used to iterate through the indexes of an array
- `for (int index = 0; index < myArray.length; ++index)`
- The built-in `.length` member helps
- Arrays also work naturally with `for-each` loops
- `for (int number : myArray)`
- Nested loops can process with multidimensional arrays

Arrays and `for`

```java
public class ArrayFor {
    public static void main(String[] args) {
        int[][] multiples = new int[5][5];
        // for loops go well with arrays
        for (int i = 0; i < multiples.length; i++) {
            for (int j = 0; j < multiples[i].length; j++) { // Notice the use of .length
                multiples[i][j] = (i + 1) * (j + 1);
            }
        }
        // So do for-each loops
        for (int[] row : multiples) {
            for (int num : row) {
                System.out.print(num + "\t");
            }
            System.out.println();
        }
    }
}
```