The Java Language

Mark A. Austin

University of Maryland

austin@umd.edu
ENCE 688P, Fall Semester 2020

September 28, 2020

Overview

1. Quick Review
2. Basic Stuff
   - Primitive Data Types, IEEE 754 Floating Point Standard
   - Three types of Java Variable
   - Arithmetic Operations
   - Control Statements
3. Packages and Import Statements
4. Methods
   - Polymorphism and Class Methods
5. Working with Arrays
   - One- and Multi-dimensional Arrays; Ragged Arrays

Part 3

Working with Arrays

Definition of an Array

Definition

An array is simply a sequence of numbered items of the same type.

In Java, permissible types include:

- Primitive data types, and
- Instances of a class.

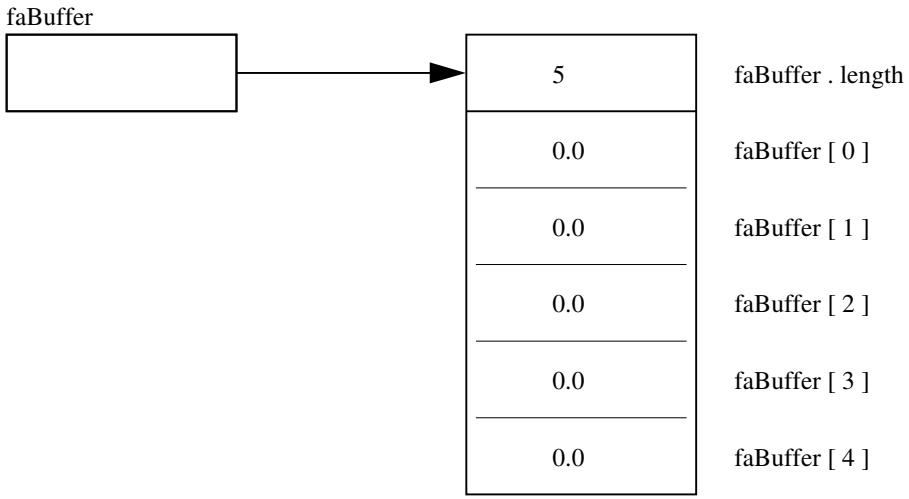
In either case, individual items in the array are referenced by their position number in the array.

One-Dimensional Arrays

Example 1. Declaration for Array of Floating Point Numbers

```
float[] faBuffer = new float [5];
```

Layout of Memory



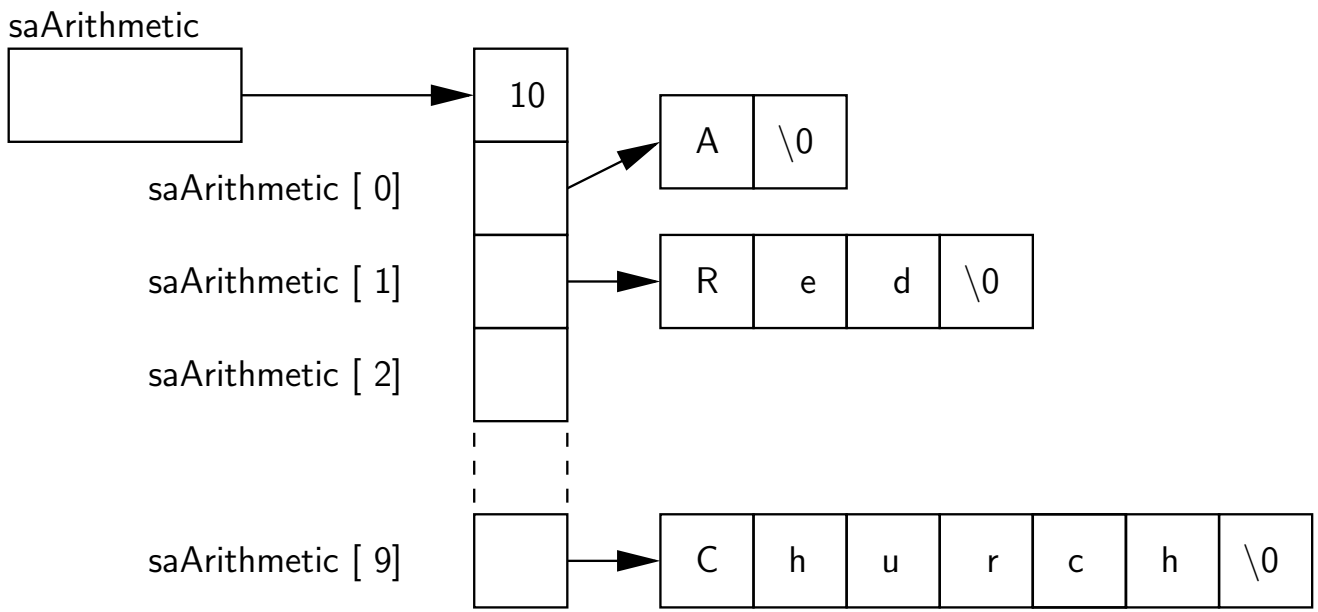
The first and last elements in the array are `faBuffer[0]` and `faBuffer[4]`. By default, all of the array elements will be initialized to zero!

One-Dimensional Arrays

Example 2. Declaration for Array of Character Strings

```
String [] saArithmetic = { "A",  "Red", "Indian", "Thought",
      "He", "Might",  "Eat", "Toffee",      "In", "Church" };
```

Abbreviated Layout of Memory



Multi-Dimensional Arrays

Multidimensional arrays are considered as arrays of arrays and are created by putting as many pairs of [] as of dimensions in your array.

Example 3. 4x4 matrix of doubles

```
double daaMat[][] = new double[4][4]; // This is a 4x4 matrix
```

Querying Dimensionality

You can query the different dimensions with the following syntax

```
array.length;          // Length of the first dimension.
array[0].length;       // Length of the second dimension.
array[0][0].length;    // Length of the third dimension.
.... etc ...
```

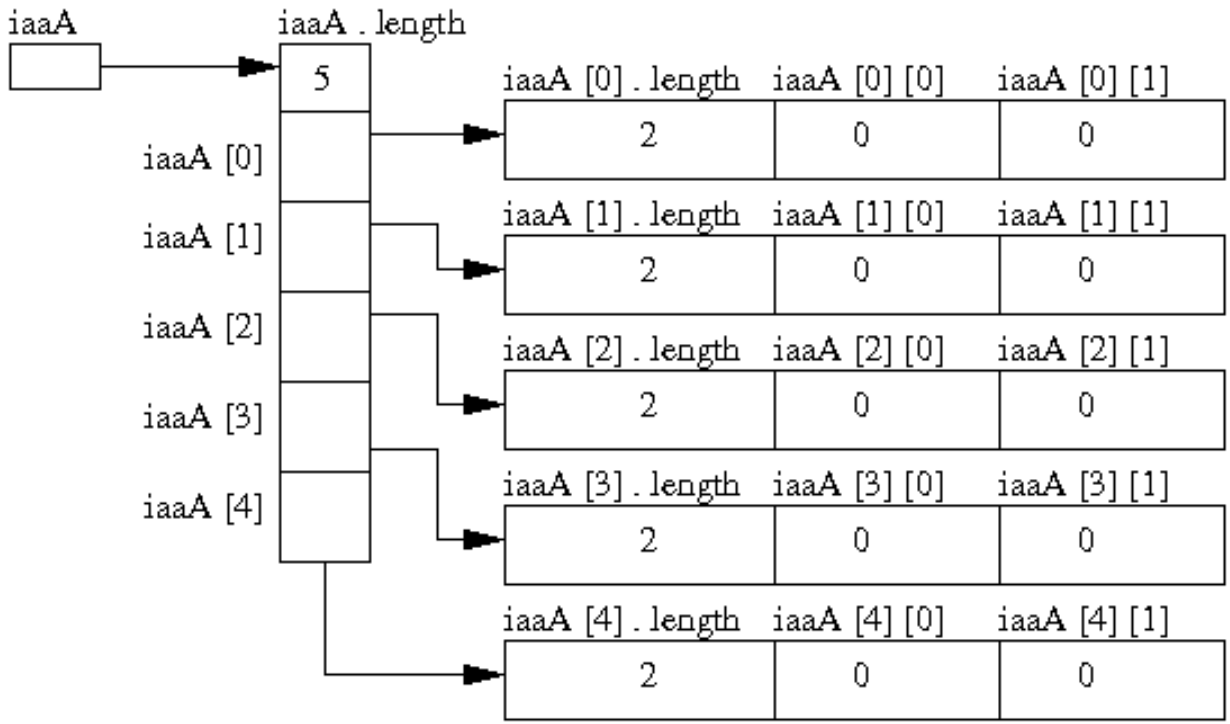
Multi-Dimensional Arrays

Example 4. Two-Dimensional Array of Ints

Array Declaration

```
int [][] iaaA = new int [5][2];
```

Layout of Memory



Ragged Arrays

Skyline Matrix Storage (Big savings in required memory):

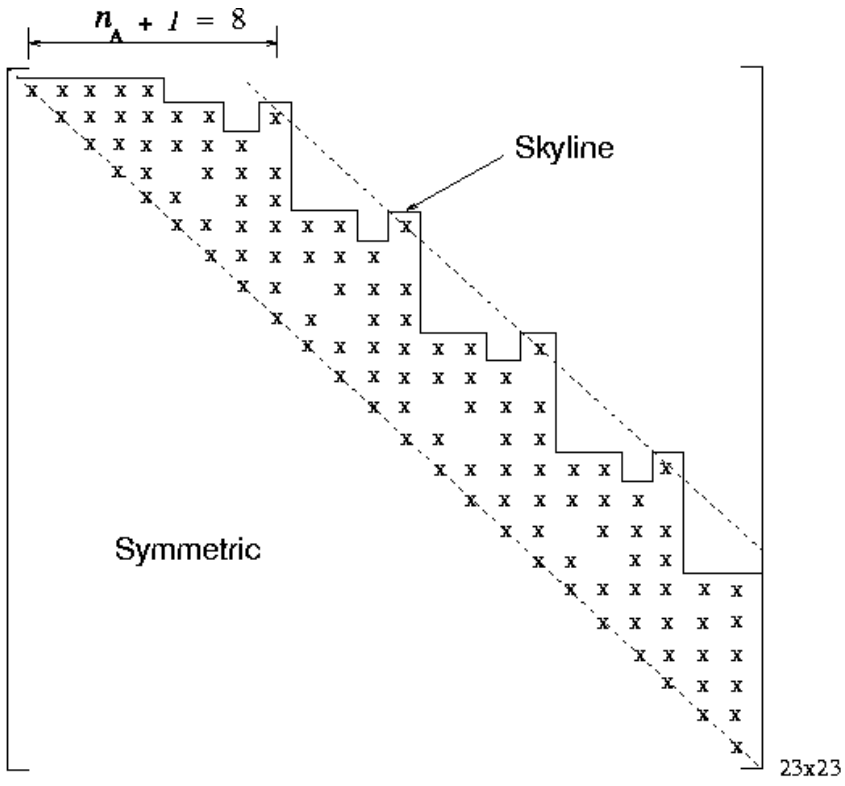


Figure 4.4: The structure of the system matrix $\mathbf{A}^t$ of our physics-based fish model, the skyline of $\mathbf{A}^t$ and its half-bandwidth n_A. Each "x" represents a non-zero entries.

Ragged Arrays

Strategy 1. Compiler Determines Layout of Memory:

```
1
2   System.out.println("Test ragged arrays with variable row length");
3   System.out.println("Method 1: Compiler determines details");
4
5   int [][] iaaB = {{1,2},{3,4,5},{6,7,8,9},{10}};
6
7   System.out.println("");
8   System.out.println("No of rows      = " + iaaB.length );
9   System.out.println("Length of row 1 = " + iaaB[0].length );
10  System.out.println("Length of row 2 = " + iaaB[1].length );
11  System.out.println("Length of row 3 = " + iaaB[2].length );
12  System.out.println("Length of row 4 = " + iaaB[3].length );
13
14  System.out.println("Array: iaaB");
15  System.out.println("-----------");
16
17  for(int i = 0; i < iaaB.length; i=i+1) {
18      for(int j = 0; j < iaaB[i].length; j=j+1)
19          System.out.printf(" %3d ", iaaB[i][j] );
20      System.out.printf("\n" );
21  }
```

Ragged Arrays

Strategy 1. Output:

```
Test ragged arrays with variable row length
-------------------------------------------
Method 1: Compiler determines details

No of rows      = 4
Length of row 1 = 2
Length of row 2 = 3
Length of row 3 = 4
Length of row 4 = 1
Array: iaaB
-----------
   1    2
   3    4    5
   6    7    8    9
  10
```

Ragged Arrays

Strategy 2." Manual Assembly of Ragged Arrays:

```
1   System.out.println("Method 2: Manual assembly of the array structure");
2
3   int[][] iaaC = new int[4][];     // Create number of rows...
4   iaaC[0] = new int[2];            // Create memory for row 1.
5   iaaC[1] = new int[3];            // Create memory for row 2.
6   iaaC[2] = new int[4];            // Create memory for row 3.
7   iaaC[3] = new int[1];            // Create memory for row 4.
8
9   iaaC[0][0] =  1; iaaC[0][1] = 2;
10  iaaC[1][0] =  3; iaaC[1][1] = 4; iaaC[1][2] = 5;
11  iaaC[2][0] =  6; iaaC[2][1] = 7; iaaC[2][2] = 8; iaaC[2][3] = 9;
12  iaaC[3][0] = 10;
13
14  System.out.println("Array: iaaC");
15  System.out.println("-----------");
16  for(int i = 0; i < iaaC.length; i=i+1) {
17      for(int j = 0; j < iaaC[i].length; j=j+1)
18          System.out.printf(" %3d ", iaaC[i][j] );
19      System.out.printf("\n" );
20  }
```

Ragged Arrays

Strategy 2. Output:

```
Method 2: Manual assembly of the array structure

Array: iaaC
-----------
   1    2
   3    4    5
   6    7    8    9
  10
```