CS 200 - Programming I: Arrays

Marc Renault

Department of Computer Sciences
University of Wisconsin – Madison

Fall 2017
TopHat Sec 3 (PM) Join Code: 719946
TopHat Sec 4 (AM) Join Code: 891624

WISCONSIN UNIVERSITY OF WISCONSIN-MADISON
Array Basics
Arrays

A data structure - a particular way of organizing some data.

Organization:
- A container that sequentially groups a fixed number of variable slots of the same type.
- Access items based on an index starting from 0.
- A (1-D) table of data.
# Creating Arrays

**Declaration**

```c
type[] arrayName;
```

- A reference type
- C/C++ style works, but is discouraged:
  ```c
  type arrayName[];
  ```
- E.g.
  ```c
  int[] a; //An int array reference variable
  ```
Creating Arrays

Declaration

type[] arrayName;

- A reference type
- C/C++ style works, but is discouraged:
  
  type arrayName[];

- E.g.
  
  int[] a; //An int array reference variable

Allocating Memory

- new operator: arrayName = new type [length];
- Table of values:
  
  arrayName = new type [] {val0, val1, ... };
- Declaration plus values:
  
  type[] arrayName = {val0, val1, ... };
Creating Arrays

Declaration

```java
type[] arrayName;
```

Allocating Memory

- new operator: `arrayName = new type [length];`
- Table of values:
  ```java
  arrayName = new type [] {val0, val1, ... };
  ```
- Declaration plus values:
  ```java
  type[] arrayName = {val0, val1, ... };
  ```

Some examples:

```java
int [] a = {11, 34, 29, 25, 42, 16, 12, 43, 47, 15};
```

```
   0 1 2 3 4 5 6 7 8 9
  a: 11 34 29 25 42 16 12 43 47 15
```
Creating Arrays

Declaration

type[] arrayName;

Allocating Memory

• new operator: \textit{arrayName} = \textit{new} type [\textit{length}];
• Table of values:
  \textit{arrayName} = \textit{new} type [] \{\textit{val0}, \textit{val1}, \ldots \};
• Declaration plus values:
  type[] arrayName = \{\textit{val0}, \textit{val1}, \ldots \};

Some examples:

\textbf{int} [] a = \{11, 34, 29, 25, 42, 16, 12, 43, 47, 15\};

\begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{array}

\begin{array}{cccccccccc}
11 & 34 & 29 & 25 & 42 & 16 & 12 & 43 & 47 & 15 \\
\end{array}
Creating Arrays

Declaration

\[\text{type[]} \ \text{arrayName};\]

Allocating Memory

- new operator: \(\text{arrayName} = \text{new} \ \text{type} \ [\text{length}]\);
- Table of values:
  \[\text{arrayName} = \text{new} \ \text{type} \ [] \ \{\text{val}0, \text{val}1, \ldots \};\]
- Declaration plus values:
  \[\text{type[]} \ \text{arrayName} = \{\text{val}0, \text{val}1, \ldots \};\]

Some examples:

\[\text{int[]} \ b = \text{new} \ \text{int}[6];\]

\[
\begin{array}{ccccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
\hline
b: & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\]
Creating Arrays

Declaration

type[] arrayName;

Allocating Memory

- new operator: arrayName = new type [length];
- Table of values:
  arrayName = new type [] {val0, val1, ... };
- Declaration plus values:
  type[] arrayName = {val0, val1, ... };

Some examples:

int [] b = new int [6];

```
0 1 2 3 4 5
b: 0 0 0 0 0 0
```
**Default Values**

**Recall: Local Variables**
- Local variables are *NOT* given default values.
- Using an uninitialized variable gives a compilation error.

**Contents of Arrays**
Contents of arrays are given *default values*:
- byte, short, int: 0
- long: 0L
- float: 0.0f
- double: 0.0d
- char: ’\0’ (or in unicode: ’\u0000’)
- boolean: false
- Any reference type: null
Accessing Array Values

Arrays are Mutable

- You can change the values stored in an array after it is created.
- **BUT:** you **CANNOT** change the length of an array after it is created.

Using and Modifying Values
**Accessing Array Values**

Arrays are indexed starting from zero.

Access the value:
```
int i = a[1];
```

Assign a new value:
```
a[2] = 10;
```

**Arrays are Mutable**

- You can change the values stored in an array after it is created.
- **BUT:** you **CANNOT** change the length of an array after it is created.

**Using and Modifying Values**

- Arrays are indexed starting from zero.
Arrays are Mutable

- You can change the values stored in an array after it is created.
- **BUT:** you *CANNOT* change the length of an array after it is created.

Using and Modifying Values

- Arrays are indexed starting from zero.
- Access the value:
  ```
  int i = a[1];
  ```
**Accessing Array Values**

Arrays are indexed starting from zero.

Access the value:
```java
int i = a[1];
```

Assign a new value:
```java
a[2] = 10;
```

**Arrays are Mutable**

- You can change the values stored in an array after it is created.
- **BUT:** you **CANNOT** change the length of an array after it is created.
**Accessing Array Values**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>11</td>
<td>34</td>
<td>10</td>
<td>25</td>
<td>42</td>
<td>16</td>
<td>12</td>
<td>43</td>
<td>47</td>
<td>15</td>
</tr>
</tbody>
</table>

**Arrays are Mutable**

- You can change the values stored in an array after it is created.
- **BUT**: you *CANNOT* change the length of an array after it is created.

**Using and Modifying Values**

- Arrays are indexed starting from zero.
- Access the value:
  ```java
  int i = a[1];
  ```
- Assign a new value:
  ```java
  a[2] = 10;
  ```
TopHat Question 1

What is the output?

```java
int[] a = new int[5];
int[] b = a;
b[0] = 3;
System.out.println(a[0]);
```
Arrays and Memory

Stack

Heap
Arrays and Memory

```java
int[] i = new int[3];
```
Arrays and Memory

```
int[] i = new int[3];
```
Arrays and Memory

```java
int[] i = new int[3];
int[] j = i;
```
Arrays and Memory

```java
int[] i = new int[3];
int[] j = i;
```
Arrays and Memory

```java
int[] i = new int[3];
int[] j = i;
j[1] = 1;
```
Arrays and Memory

```java
int[] i = new int[3];
int[] j = i;
j[1] = 1;
```
Arrays and Memory

```java
int[] i = new int[3];
int[] j = i;
j[1] = 1;
int[] k = {1, 2, 3, 4};
```
Arrays and Memory

```java
int[] i = new int[3];
int[] j = i;
j[1] = 1;
int[] k = {1, 2, 3, 4};
```
Arrays and Memory

```java
int[] i = new int[3];
int[] j = i;
   j[1] = 1;
int[] k = {1, 2, 3, 4};
i = new int[] {1, 2, 3, 4};
```
Arrays and Memory

```
int[] i = new int[3];
int[] j = i;
j[1] = 1;
int[] k = {1, 2, 3, 4};
i = new int[] {1, 2, 3, 4};
```
Arrays and Memory

```java
int[] i = new int[3];
int[] j = i;
j[1] = 1;
int[] k = {1, 2, 3, 4};
i = new int[] {1, 2, 3, 4};
int[] l = {1, 2, 3, 4};
```
Arrays and Memory

int[] i = new int[3];
int[] j = i;
j[1] = 1;
int[] k = {1, 2, 3, 4};
i = new int[] {1, 2, 3, 4};
int[] l = {1, 2, 3, 4};
Arrays and Memory

```java
int[] i = new int[3];
int[] j = i;
j[1] = 1;
int[] k = {1, 2, 3, 4};
i = new int[] {1, 2, 3, 4};
int[] l = {1, 2, 3, 4};
k[2] = 7;
```
Arrays and Memory

```java
int[] i = new int[3];
int[] j = i;
j[1] = 1;
int[] k = {1, 2, 3, 4};
i = new int[] {1, 2, 3, 4};
int[] l = {1, 2, 3, 4};
k[2] = 7;
```
TopHat Question 2

What is the output?

```java
int[] a = {4, 12, -3, 0, 5};
int[] b = a;
int[] c = {4, 12, -3, 0, 5};
System.out.print(a == b);
System.out.print(" ");
System.out.print(b == c);
```
TopHat Question 3

What is the output?

```java
int[] a = {4, 12, -3, 0, 5};
System.out.print(a);
```

- a. [4 12 -3 0 5]
- b. Compile error
- c. I’d like it to print out the array, but this is clearly a trick question and it probably prints out something like [I@4f8e5cde
TopHat Question 3

What is the output?

```java
int[] a = {4, 12, -3, 0, 5};
System.out.println(a);
```

Default toString()

- Default String representation of an object.
TopHat Question 3

What is the output?

```java
int[] a = {4, 12, -3, 0, 5};
System.out.print(a);
```

Default toString()

- Default String representation of an object.
- Hash code: A function that converts the object data into a signed 32-bit integer.
Loops and Arrays
Iterating through Arrays

Length of an Array

- Arrays have final member variable `length` called `length`.
- `int i = a.length; // i is 10`

Iterating an Array

- TopHat Question 4: What loop would generally be the most suited for iterating through an array?
**Example: Printing an Array**

```java
public class PrintArrayEx {

    public static void main(String[] args) {
        int[] a = {4, 12, -3, 0, 5};
        for (int i = 0; i < a.length; ++i) {
            System.out.print(a[i] + " ");
        }
    }
}
```
**Example: Printing an Array**

```java
public class PrintArrayEx {

    public static void main(String[] args) {
        int[] a = {4, 12, -3, 0, 5};
        for(int i = 0; i < a.length; ++i) {
            System.out.print(a[i] + " ");
        }

        // Pretty Print
        System.out.print("[");
        for(int i = 0; i < a.length; ++i) {
            System.out.print(a[i]);
            if(i < a.length - 1)
                System.out.print(" ");
        }
        System.out.print("]");
    }
}
```
What is the output?

```java
int [] a = {1, 2, 3, 4, 5};
int b = 0;
for (int i = 0; i < a.length; ++i) {
    b += a[i];
}
System.out.print(b/a.length);
```
**Sorting Exercise**

Given an array of integers, sort the array from smallest to largest.
More on Arrays
Arrays of Reference Types

String[] sArr = {"Dog", "Cat", "Bear"};
Arrays of Reference Types

String[] sArr = {"Dog", "Cat", "Bear"};
Arrays of Reference Types

String[] sArr = {"Dog", "Cat", "Bear"};
Arrays of Reference Types

String[] sArr = {"Dog", "Cat", "Bear"};
Arrays of Reference Types

String[] sArr = {"Dog", "Cat", "Bear"};
Arrays of Reference Types

String[] sArr = {"Dog", "Cat", "Bear"};
**Arrays of Reference Types**

```java
String[] sArr = {"Dog", "Cat", "Bear"};
String[] tArr = new String[4];
```
Arrays of Reference Types

```java
String[] sArr = {"Dog", "Cat", "Bear"};
String[] tArr = new String[4];
```
Arrays of Reference Types

String[] sArr = {"Dog", "Cat", "Bear"};
String[] tArr = new String[4];
Parallel Arrays

Programming Technique

A way of relating data (especially different types) via the same index across multiple arrays.
**Parallel Arrays**

Programming Technique

A way of relating data (especially different types) via the same index across multiple arrays.

```java
final int NUM_MONTHS
String[] monthName = new String[NUM_MONTHS];
String[] monthAbrv = new String[NUM_MONTHS];
int[] monthDays = new int[NUM_MONTHS];
int i = 0;
monthName[i] = "January"; monthAbrv[i] = "Jan"; monthDays[i++] = 31;
monthName[i] = "February"; monthAbrv[i] = "Feb"; monthDays[i++] = 28;
monthName[i] = "March"; monthAbrv[i] = "Mar"; monthDays[i++] = 31;
monthName[i] = "April"; monthAbrv[i] = "Apr"; monthDays[i++] = 30;
monthName[i] = "May"; monthAbrv[i] = "May"; monthDays[i++] = 31;
monthName[i] = "June"; monthAbrv[i] = "Jun"; monthDays[i++] = 30;
monthName[i] = "July"; monthAbrv[i] = "Jul"; monthDays[i++] = 31;
monthName[i] = "August"; monthAbrv[i] = "Aug"; monthDays[i++] = 31;
monthName[i] = "September"; monthAbrv[i] = "Sep"; monthDays[i++] = 30;
monthName[i] = "October"; monthAbrv[i] = "Oct"; monthDays[i++] = 31;
monthName[i] = "November"; monthAbrv[i] = "Nov"; monthDays[i++] = 30;
monthName[i] = "December"; monthAbrv[i] = "Dec"; monthDays[i++] = 31;
```
Searching Exercise

Write a method that returns the index of a given value within a sorted array of integers (sorted smallest to largest; assuming only 1 instance of any value exists).
SEARCHING EXERCISE

Write a method that returns the index of a given value within a sorted array of integers (sorted smallest to largest; assuming only 1 instance of any value exists).

Without iterating through every single element of the array.
Further Reading

COMP SCI 200: Programming I
zyBook code:
WISCCOMPSCI200Fall2017

● Chapter 7. Arrays
Appendix
References
Image Sources I

https://brand.wisc.edu/web/logos/

http://www.zybooks.com/