Announcements

• Lab06 tomorrow
• No discussion/quiz Thursday
• Review session Friday
• P4 due Monday
Goodbye lists

You will see lists again:
• CMSC132 and beyond
• Java’s built-in lists at the end of the summer
Hello arrays

Index

length == 5

Element
Arrays

• An array is a contiguous block of memory
  – Each element is the same data-type
  – Elements are numbered from zero

• Can be allocated at runtime

• Each element can be accessed directly
  – No list traversal
Array Syntax

• Declaring arrays:

\[
\text{int}\ [\ ] \ \text{array1}\;
\]

data-type of each element

data-type of array

variable name

• Alternate syntax (discouraged):

\[
\text{int} \ \text{array2}\[\ ];
\]
Array Syntax

• Initializing arrays:

  //Allocate an array with 3 elements.  
  //Elements default to zero.
  array1 = new int[3];
  //Allocate an array with 3 elements. 
  //Elements are specified in {...}.
  array2 = {1, 4, 3};
Array Syntax

• Getting individual elements:
  \[
  \textbf{int } x = \text{myArray}[\text{index}];
  \]

• Setting individual elements:
  \[
  \text{myArray}[\text{index}] = \text{value};
  \]

• Accessing array length:
  \[
  \textbf{int } \text{len } = \text{myArray.length};
  \]

• Histogram example
Arrays in memory

• Arrays are allocated on the heap just like objects
  – Array variables are reference variables
    • Aliasing is possible
  – Length is determined at the time of allocation and is fixed thereafter
  – Contents can be accessed and changed throughout execution

• Array elements can be any data-type
  – Primitives: ints, booleans, doubles, etc.
  – Reference variables: Objects, other arrays
Array reference variables

Stack

Heap

```
0  1  3  -2  4  1
```

```
@
```

a
Array reference variables

Stack

a
b

Heap

0 1 3 -2 3 4 1

@ 1 @
Array reference variables

Stack

- a
- b
- c

Heap

```
0  1  3  -2  3  4  4  1
```

```
0  4  1  10  2  -1
```

obj

data

@
Arrays of references

- Student array example
Iterating over arrays

- Arrays are often used in conjunction with loops:

```java
//Print the contents of array on a single line
for (int i = 0; i < array.length; i++) {
    System.out.print(array[i]+" ");
}
```
Copying arrays

- Often you may need to copy an array
  - Adding additional storage space (extend example)
  - Inserting additional elements (insert example)
- Changes to the original should not affect the copy
  - Aliasing may not be sufficient
- What if the array elements are not primitive?
  - Student array example
Reference copy

original @

copy @

@ @ @

obj1  obj2  obj3
Shallow copy

original

copy

![Diagram showing shallow copy](image-url)
Deep copy

original

@ ➔ @ ➔ @ ➔ obj1 ➔ obj2 ➔ obj3

copy

@ ➔ @ ➔ @ ➔ copy1 ➔ copy2 ➔ copy3
Arrays of arrays (2D)

• Matrix Example
“Ragged” arrays

- Pascal’s triangle example
3D (4D, 5D,...) arrays

Stack

Heap

@ → @

@ → 2  2
@ → 5  0
@ → -3 10

@ → 1  0
@ → 0  2
@ → -1 10