Arrays

Department of Computer Science

Slides Originally Created by Albert Lionelle
(Albert.Lionelle@colostate.edu), updated by Marcia Moraes
(marcia.moraes@colostate.edu)

Announcements

TODO Reminders:

Readings are due **before** lecture

- Reading 20 (zybooks) – you should have already done that ☺
- Lab 13
- Reading 21 (zyBooks) – you should have already done that ☺
- Lab 14 – optional because of snow day
- Reading 22 (zybooks) – you should have already done that ☺
- RPA 10

Keep practicing your RPAs in a spaced and mixed manner ☺

https://stock.adobe.com/search/images?k=%22happy+friday%22

Friday Help Desk –
12-4pm CSB120

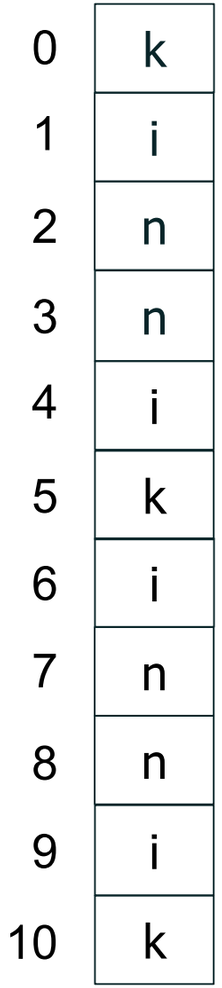
Friday Help Session –
1-2pm Teams

Saturday Help Desk –
12-4pm Teams

Sunday Help Desk –
3-7pm Teams

Recalling The Past

- For every value you want to store
  - You need a variable
- What if you want to store 100 values? 10,000 values?
  - Use ArrayLists for storing object
  - But how about primitive types?
    - Introducing Arrays
    - Reserving memory for storing values, in order from the 0 index
- Sound Familiar - Recall String
  - The String object contains
    - chars in order!
  - It is a character array!



Basic Array

- `char[] palindrome = {‘k’, ‘i’, ‘n’, ‘n’, ‘i’, ‘k’}; // ok shorter version`
  - builds an array with six elements.
  - with those values in the memory locations
- Another way:
  - `char[] palindrome =  new char[6];  // declare the size of the array (more common)`
  - `palindrome[0] = ‘k’;`
  - `palindrome[1] = ‘i’;`
  - `palindrome[2] = ‘n’;`
  - `palindrome[3] = ‘n’;`
  - `palindrome[4] = ‘i’;`
  - `palindrome[5] = ‘k’;`

Arrays are Mutable

- Elements in the array can be changed / reset!

```
char[] palindrome = {‘k’, ‘i’, ‘n’, ‘n’, ‘i’, ‘k’};

palindrome[0] = ‘c’;

palindrome[2] = ‘i’;

System.out.println(Arrays.toString(palindrome)); // prints [c,i,i,n,i,k]
```

Arrays can be any type!

- `int[] values = new int[100];`
  - default values are 0
  - true for numeric primitives
- `String[] names = new String[10];`
  - default values are null
  - true for all objects
- Format is:
  - `TYPE[] name = new TYPE[size];`

```
String[] rhps = new String[10];

rhps[0] = “brad”;
rhps[1] = “janet”;
rhps[2] = “magenta”;
rhps[3] = “columbia”;
rhps[4] = “riff-raff”;
rhps[5] = “eddie”;
rhps[6] = “scott”;
rhps[7] = “frankie”;
rhps[5] = “rocky”;
System.out.println(Arrays.toString(rhps));
```

[brad, janet, magenta, columbia, riff-raff, rocky, scott, frankie, null, null]

Array Length

- Array size allocated to size 10
  - 0..9 indices valid
- rhps[20]
  - throws IndexOutOfBoundsException!
- How to check for that?
  - .length
  - notice no parentesis, command, not a method
- rhps.length
  - returns 10
- which means
  - we always know the 1st index (0)
  - and the last (rhps.length-1)
  - no matter the size of the array

```
String[] rhps = new String[10];

rhps[0] = “brad”;
rhps[1] = “janet”;
rhps[2] = “magenta”;
rhps[3] = “columbia”;
rhps[4] = “riff-raff”;
rhps[5] = “eddie”;
rhps[6] = “scott”;
rhps[7] = “frankie”;

rhps[20] = “rocky”; // error!
```

Loops and Arrays

- What is going to be printed?

```
String[] tran = new String[10];
for(int i = 0; i < tran.length; i++) {
    tran[i] = String.format(“Tran %d”, i);
}

for(int i = 0; i < tran.length; i++) {
    System.out.println(tran[i]);
}
```

```
Tran 0
Tran 1
Tran 2
Tran 3
Tran 4
Tran 5
Tran 6
Tran 7
Tran 8
Tran 9
```

Loops and Arrays

- What is going to be printed?

```
String[] seats = new String[4];

seats[0] = "Amy";

seats[2] = "Rory";

        System.out.println(Arrays.toString(seats));

for(int i = 0; i < seats.length; i++) {

            System.out.printf("%s is in seat %d%n", seats[i] != null ? seats[i] : "No one", i+1);

}
```

```
[Amy, null, Rory, null]
Amy is in seat 1
No one is in seat 2
Rory is in seat 3
No one is in seat 4
```

Arrays versus ArrayLists

- ArrayLists are lists that use Array as the underlining structure
- Arrays are just how you declare a group of objects in order
- ArrayList is an individual object someone wrote

feature	array	ArrayList
can contain primitives	X	only with wrapper classes and boxing/unboxing
fixed size	X	
variable size		X
how to access elements	direct access	through methods (.get, .set, .add)
speed	faster	slower due to method overhead

When use arrays over ArrayLists?

- When your size is fixed, arrays are much faster to use!

- When you need to keep order on sparsely populated datasets (that are often fixed sizes)
  - [value, null, null, null, value, null, value]

- They are used about equally, just depends on what you are doing.

When use arrays over ArrayLists?

```
long[] values = new long[1000000];
Instant start = java.time.Instant.now();
for(int i = 0; i < values.length; i++) {
    values[i] = i * 10l;
}
Instant end = java.time.Instant.now();
System.out.println("Array Loop Done: " + java.time.Duration.between(start, end).toMillis());
List<Long> valList = new ArrayList<>(); // remember polymorphism use List
start = java.time.Instant.now();
for(int i = 0; i < 1000000; i++) {
     valList.add(i*10l);
}
end = java.time.Instant.now();
System.out.println("List Loop Done: " + java.time.Duration.between(start, end).toMillis());
//note this is not really the best way to determine the time between algorithms just an example
```

```
Array Loop Done: 6
List Loop Done: 61
```

Worksheet

- Do the arrays worksheet.

- In class code and worksheet code - https://github.com/CSU-CompSci-CS163-4/Handouts/tree/main/ClassExamples/10Arrays