CS 106
INTRODUCTION TO DATA STRUCTURES

SPRING 2020

PROF. SARA MATHIESON

HAVERFORD COLLEGE

ADMIN

- **By lab on Friday you should at least be through Section 1 of Lab 1**

Tip: include a toString *method for your class for quick testing*

- **Lab 2 (continuation of Lab 1) is now posted. Before starting:**
  - tag your Lab 1 repo
  - do the pre-lab exercises

FEB 6 OUTLINE

- Recap exceptions and arrays
- Arrays with loops (including foreach)
- Are Strings arrays? (+ StringBuilder)
- ArrayLists and Generics

FEB 6 OUTLINE

- **Recap exceptions and arrays**
- **Arrays with loops (including foreach)**
- **Are Strings arrays? (+ StringBuilder)**
- **ArrayLists and Generics**

What's wrong with returning a default value?

How an internet mapping glitch turned a random Kansas farm into a digital hell

https://splinternews.com/how-an-internet-mapping-glitch-turned-a-random-kansas-f-1793856052

Slide: Sorelle Friedler

EXCEPTION HIERARCHY

Just like all objects extend **Object**, all exceptions extend **Exception** (which extends **Object**).

You can make your own specialized **Exceptions**!

```
public class MyException extends Exception {
   public MyException() {
      super("error message here");
   }
}
```

Slide: Sorelle Friedler

CODING EXERCISE (OUTSIDE OF CLASS)

1. Make a new **`NotMyNameException`** class

2. Make a function **`myNameIs`** that takes a **`String`** as input and throws an exception if the given **`String`** is not your name

3. Call your **`myNameIs`** function from the `main` function in a try / catch block

4. Print out a message and the stack trace (**`e.printStackTrace()`**) associated with the exception in the catch block in main

Slide: Sorelle Friedler

HANDOUT 5, PAGE 2

```
int[] arr = new int[3];
arr[1] = 7;
System.out.println(Arrays.toString(arr));

int[] brr = arr;
brr[2] = 10;
System.out.println(Arrays.toString(arr));
```

arr and **brr** point to the same object in memory, so the output is:
[0, 7, 0]
[0, 7, 10]

Note: **Arrays.toString(..)** is a *static* method in the **Arrays** class in Java, it returns a **String** representation of the array by calling **toString** on the individual elements

HANDOUT 5, PAGE 2

```
String[] colors = {"red", "blue",
   "green"};

System.out.println(colors.length);

colors.append("yellow");

System.out.println(colors.length);
```

HANDOUT 5, PAGE 2

```
String[] colors = {"red", "blue",
   "green"};
System.out.println(colors.length);
colors.append("yellow");
System.out.println(colors.length);
```

We cannot change the size of an array!

HANDOUT 5, PAGE 2

```
String greeting = "Hello!”;
System.out.println(greeting[0]);
greeting[5] = "o”;
System.out.println(greeting);
```

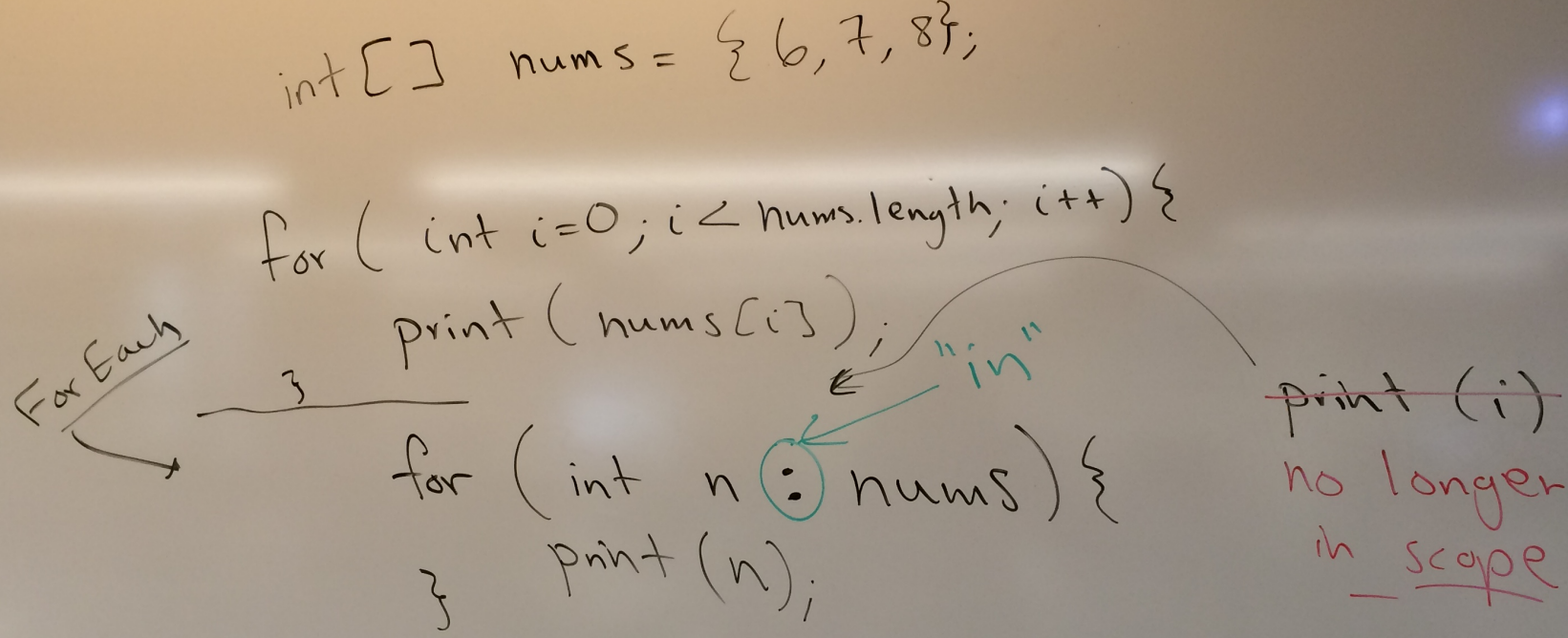
HANDOUT 5, PAGE 2

```
String greeting = "Hello!";
System.out.println(greeting[0]);
greeting[5] = "e";
System.out.println(greeting);
```

Strings are immutable!

FEB 6 OUTLINE

- Recap exceptions and arrays
- **Arrays with loops (including foreach)**
- Are Strings arrays? (+ StringBuilder)
- ArrayLists and Generics


int[] nums = {6, 7, 8};
for (int i=0; i < nums.length; i++) {
print (nums[i]);
}
For Each
for (int n : nums) {
print (n);
}
"in"
print (i)
no longer
in scope

ARRAY CODING EXERCISE

- **First create a new Java Project called ArrayPractice**
- **Inside the src folder of the project, create a new class Main**
- **Create a main method within your Main class**

1. **Create an array of Strings to hold 5 Strings.**
2. **Put 5 Strings into your array (your choice!)**
3. **Add a 6th String to your array:**
   a. create a new String array with longer length
   b. copy all the old strings over using a loop
   c. put the 6th string into the new array
4. **Use Arrays.toString(<arr>) to print out both arrays**

ARRAY CODING EXERCISE

Example Solution
(not commented)

```
import java.util.Arrays;

public class Main {

    public static void main(String[] args) {

        // way 1 of initializing an array
        String[] myWords = {"Feb", "Mar", "April", "May", "June"};

        // way 2
        String[] myWords2 = new String[5];
        myWords2[0] = "Feb";
        //...

        String[] myWordsLong = new String[6];

        for (int i=0; i < myWords.length; i++) {
            myWordsLong[i] = myWords[i];
        }

        myWordsLong[5] = "July";

        System.out.println(Arrays.toString(myWords));
        System.out.println(Arrays.toString(myWordsLong));

    }
```

FEB 6 OUTLINE

- Recap exceptions and arrays
- Arrays with loops (including foreach)
- **Are Strings arrays? (+ StringBuilder)**

  No, the are immutable! See last board slide for StringBuilder example
- ArrayLists and Generics

FEB 6 OUTLINE

- Recap exceptions and arrays
- Arrays with loops (including foreach)
- Are Strings arrays? (+ StringBuilder)
- **ArrayLists and Generics**

WHAT IF WE COPIED THE ARRAY EVERY TIME?

Unfortunately I forgot to take a picture of this board, but convince yourself that if you copied over the array every time you wanted to add an element, it would be quadratic in the size of the array

=> Very slow :-(

ArrayList<String> al = new ArrayList<String>();

whatever I want (generic)

put # here

call the constructor.

operations = 2n — linear

al.add("dog");
al.add("cat");
al.add("fish");
al.add("bird"); ← automatic resize (make double prev size)

// 1 operation

10 times

copy => 10 steps

int[]

type

HEAP

garbage collected Strings are 10 bytes

al

84 (address)

size 10 array

"dog" "cat" "fish"

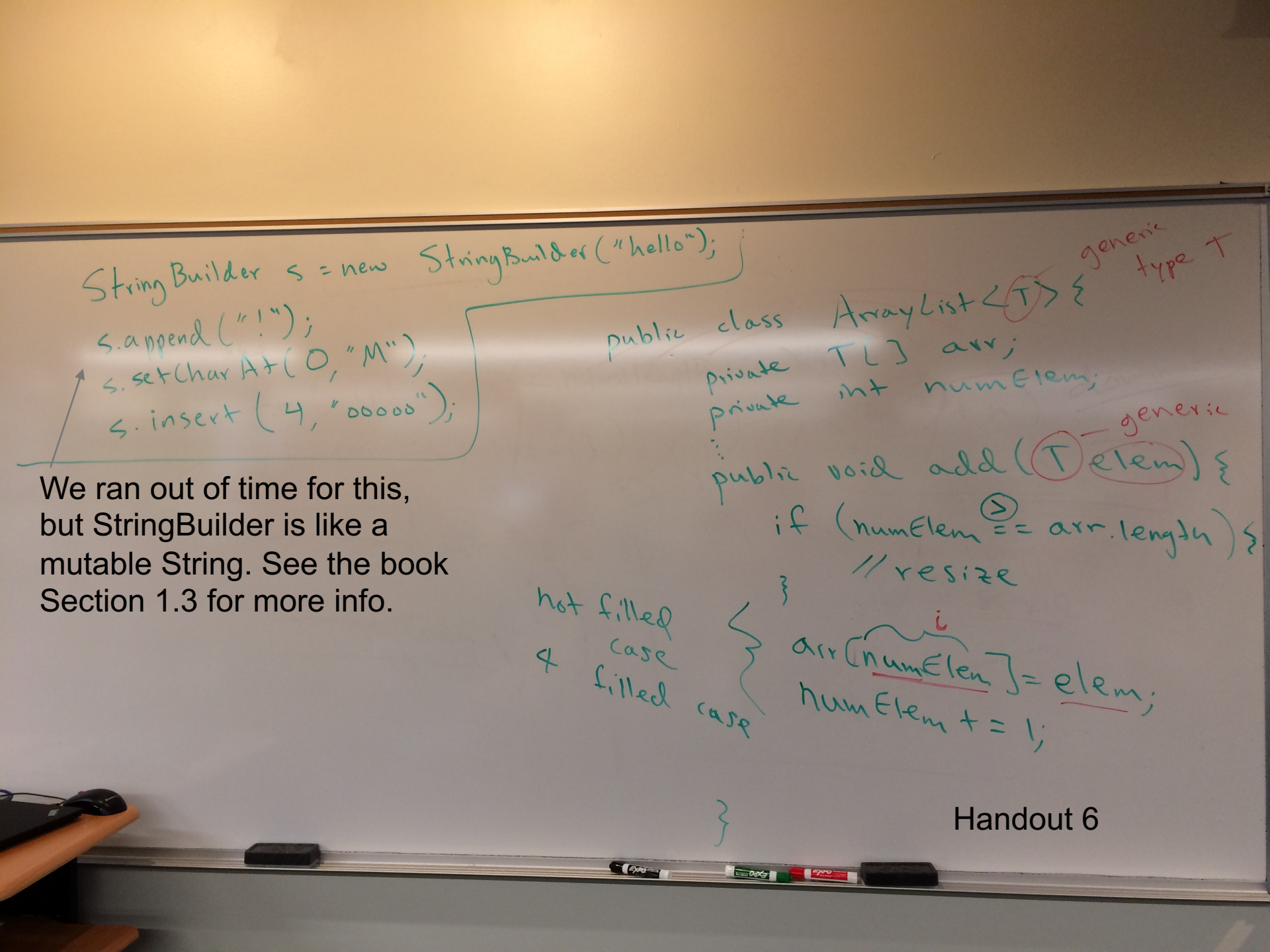
5th element

change reference

← 20 → "bird"

84 + 5·10 = 134

Arrays: constant time to get/set


StringBuilder s = new StringBuilder("hello");
s.append("!");
s.setCharAt(0, "M");
s.insert(4, "00000");
public class ArrayList<T> {
generic type T
private T[] arr;
private int numElem;
public void add(T elem) {
generic
if (numElem == arr.length) {
// resize
}
not filled case & filled case
arr[numElem] = elem;
numElem += 1;
}
We ran out of time for this, but StringBuilder is like a mutable String. See the book Section 1.3 for more info.
Handout 6

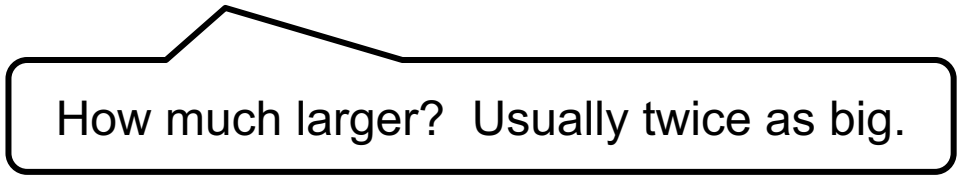
ARRAYLISTS

ArrayLists are based on arrays, but do the dynamic growth and removal for you!

remove method: moves all of the items > i over one

add method: checks to see if there's enough room to add the new item and, if not, makes a new larger array and copies everything over

Note: both of these methods have the potential to be slow!



Want more info? See book section 7.2.

ARRAYLISTS: INITIALIZATION

```
import java.util.ArrayList;
ArrayList<String> myList = new ArrayList<String>();
```

Imports the library for the ArrayList class.

The type that the underlying array will be declared to have. These are known as generics in Java.

No need to give a specific length!

Source code for the class:

```
public class ArrayList<T> {
      private T[] dataArray;
…
```

This is how classes with generic types are created and then used.

https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html

Slide: Sorelle Friedler

CODING EXERCISE (OUTSIDE CLASS)

Within a Main class and main method…

1.Create an array of Strings to hold 5 Strings.

2.Put 5 Strings into your array (your choice!)

3.Create an ArrayList

4.Copy the 5 Strings from the array into the ArrayList

5.Remove all the Strings from the ArrayList

Slide: Sorelle Friedler