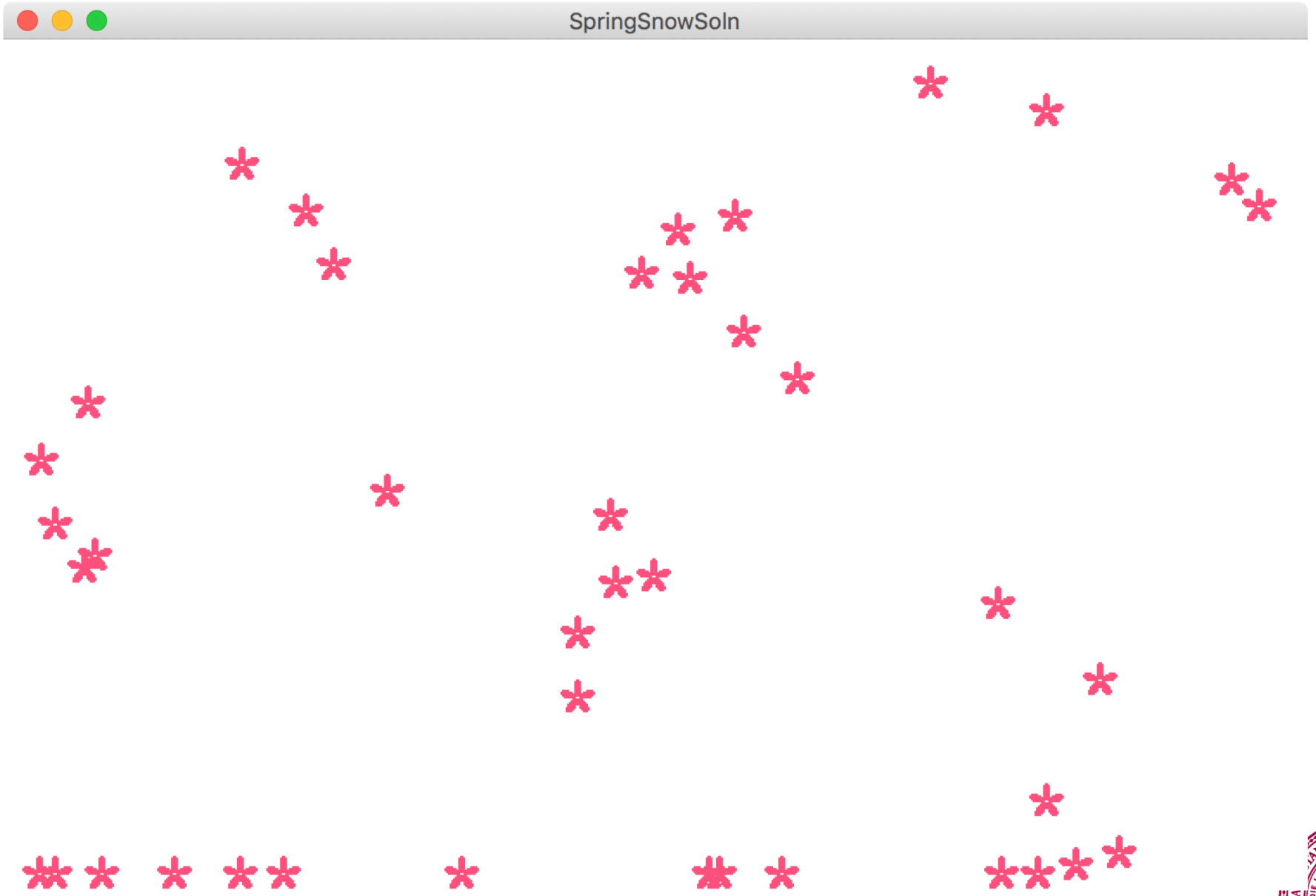
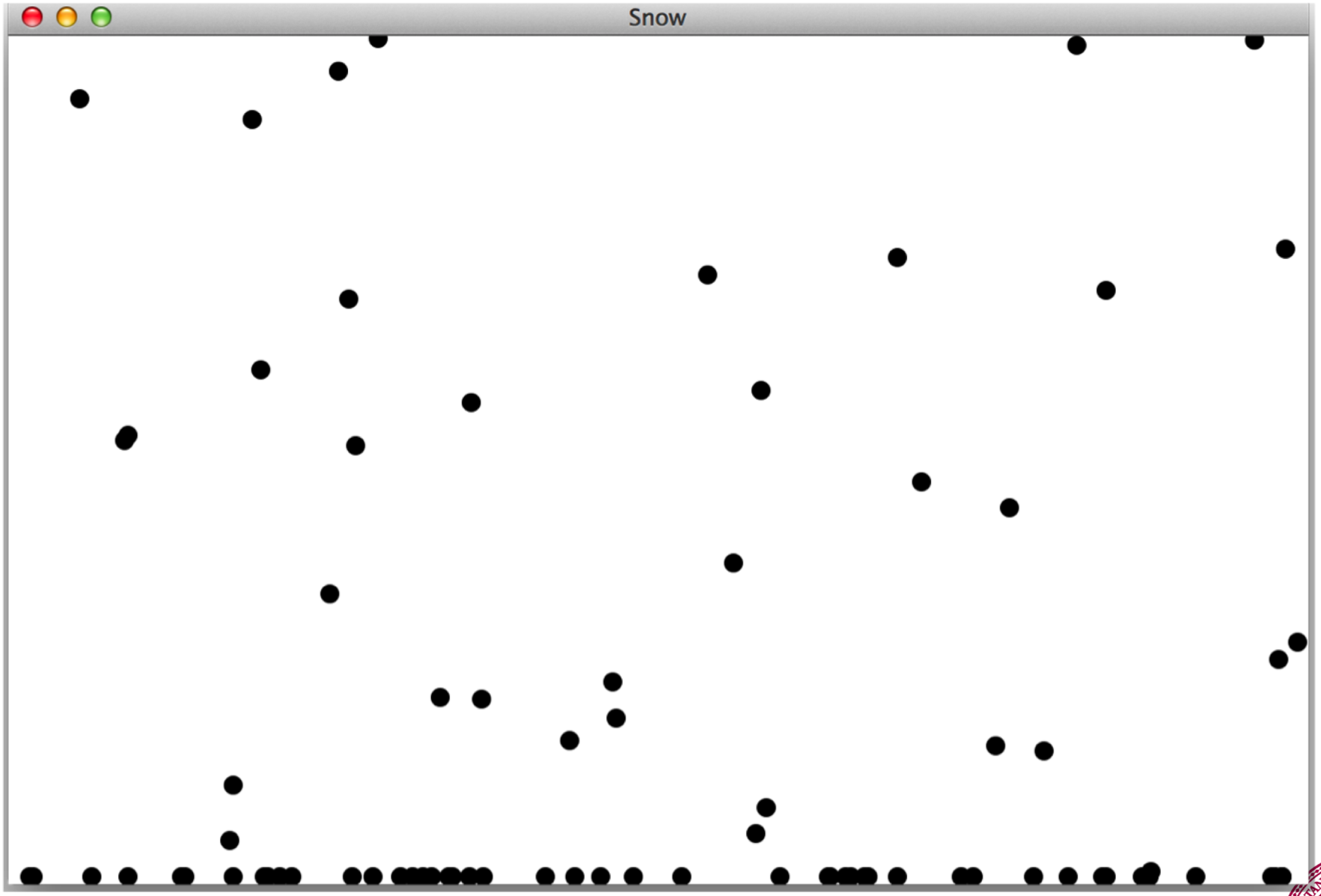
ArrayLists

Chris Piech
CS106A, Stanford University

Why is this hard to write?



Why is this hard to write?

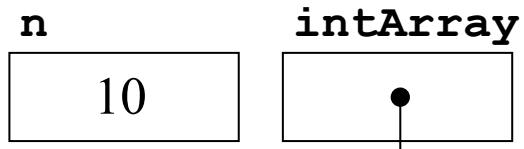


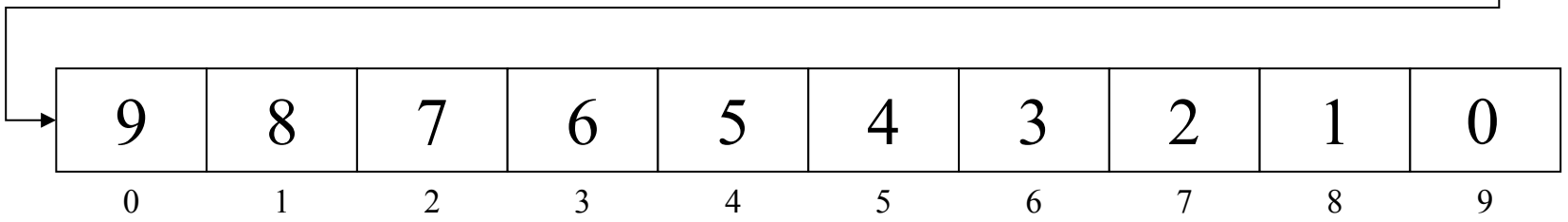
Data Structures

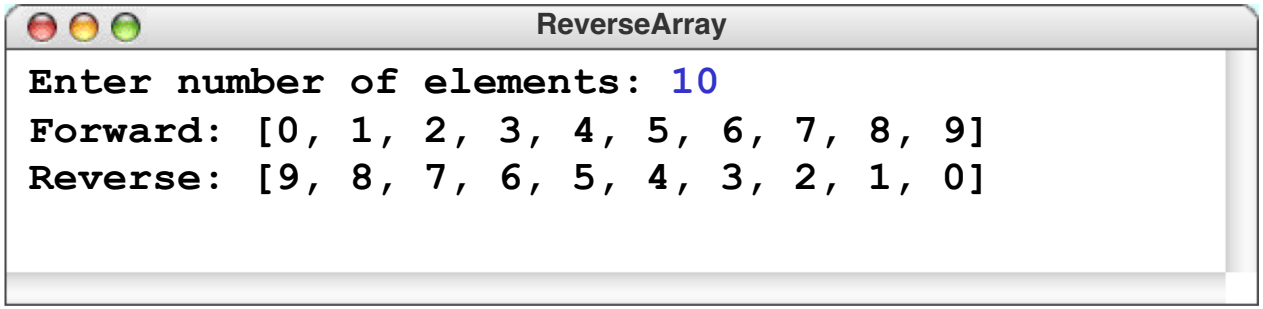
Operation	Strings	Arrays
Make a new one	`String str = "abc";`	`int arr = new int[5];`
Get length?	`str.length()`	`arr.length`
Get element?	`str.charAt(i)`	`arr[i]`
Set element?	*Not allowed*	`arr[i] = 5;`
Loop?	`for(int i = 0; i < str.length(); i++)`	`for(int i = 0; i < arr.length(); i++)`

Review: Reverse Array Program

```
public void run() {
   int n = readInt("Enter number of elements: ");
   int[] intArray = createIndexArray(n);
   println("Forward: " + arrayToString(intArray));
   reverseArray(intArray);
   println("Reverse: " + arrayToString(intArray));
}
```





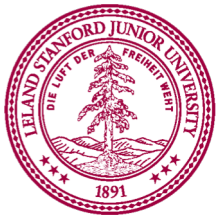


skip simulation

What does this say?

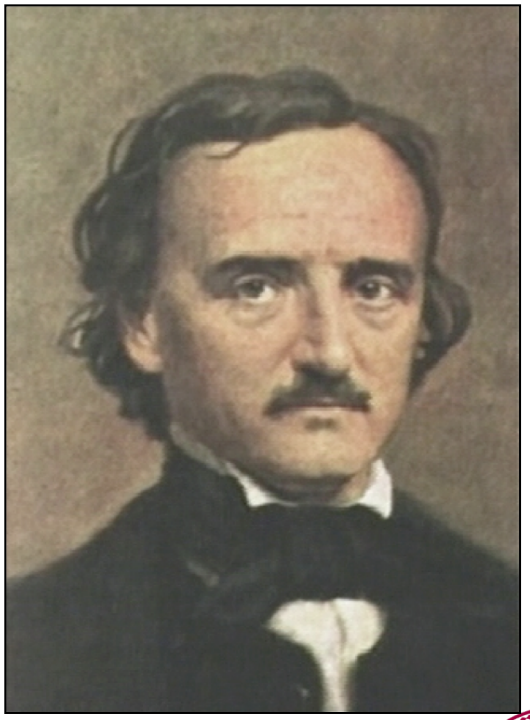
```
53‡‡†305))6*;4826)4‡•)4‡);806*;48†8¶
60))85;1‡(;:‡*8†83(88)5*†;46(;88*96*
?;8)*‡(;485);5*†2:*‡(;4956*2(5*-4)8¶
8*;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡
1;48†85;4)485†528806*81(‡9;48;(88;4(
‡?34;48)4‡;161;:188;‡?;
```

Puzzle in Gold Bug by Edgar Allan Poe



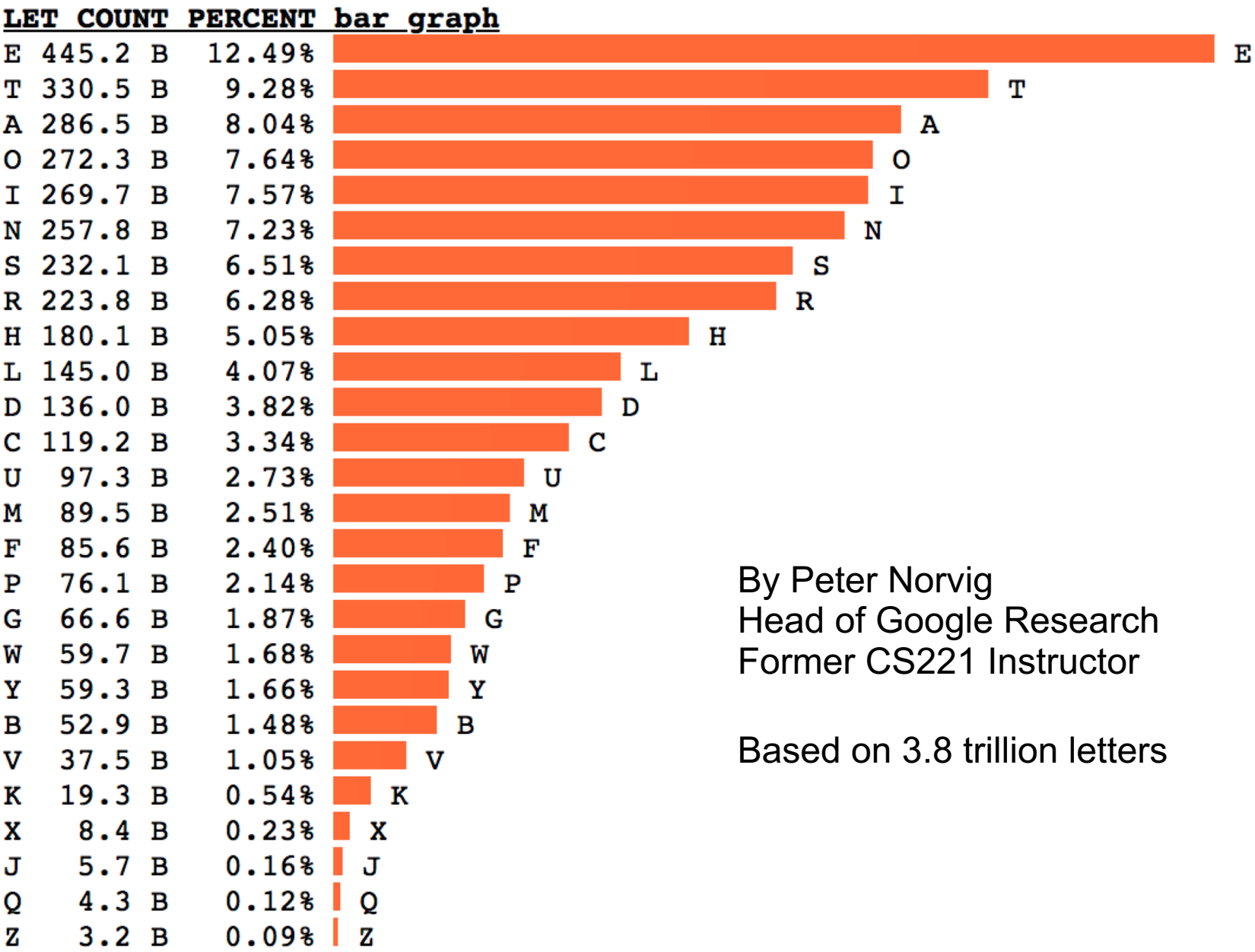
Cryptogram

- A ***cryptogram*** is a puzzle in which a message is encoded by replacing each letter in the original text with some other letter. The substitution pattern remains the same throughout the message. Your job in solving a cryptogram is to figure out this correspondence.
- One of the most famous cryptograms was written by Edgar Allan Poe in his short story "The Gold Bug."
- In this story, Poe describes the technique of assuming that the most common letters in the coded message correspond to the most common letters in English, which are E, T, A, O, I, N, S, H, R, D, L, and U.



Edgar Allan Poe (1809-1849)

Letter Frequency



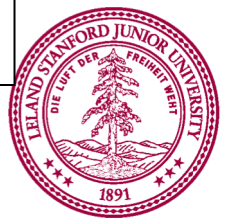
By Peter Norvig
Head of Google Research
Former CS221 Instructor

Based on 3.8 trillion letters

Poe's Cryptographic Puzzle

```
53‡‡†305))6*;4826)4‡•)4‡);806*;48†8¶
60))85;1‡(;:‡*8†83(88)5*†;46(;88*96*
?;8)*‡(;485);5*†2:*‡(;4956*2(5*–4)8¶
8*;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡
1;48†85;4)485†528806*81(‡9;48;(88;4(
‡?34;48)4‡;161;:188;‡?;
```

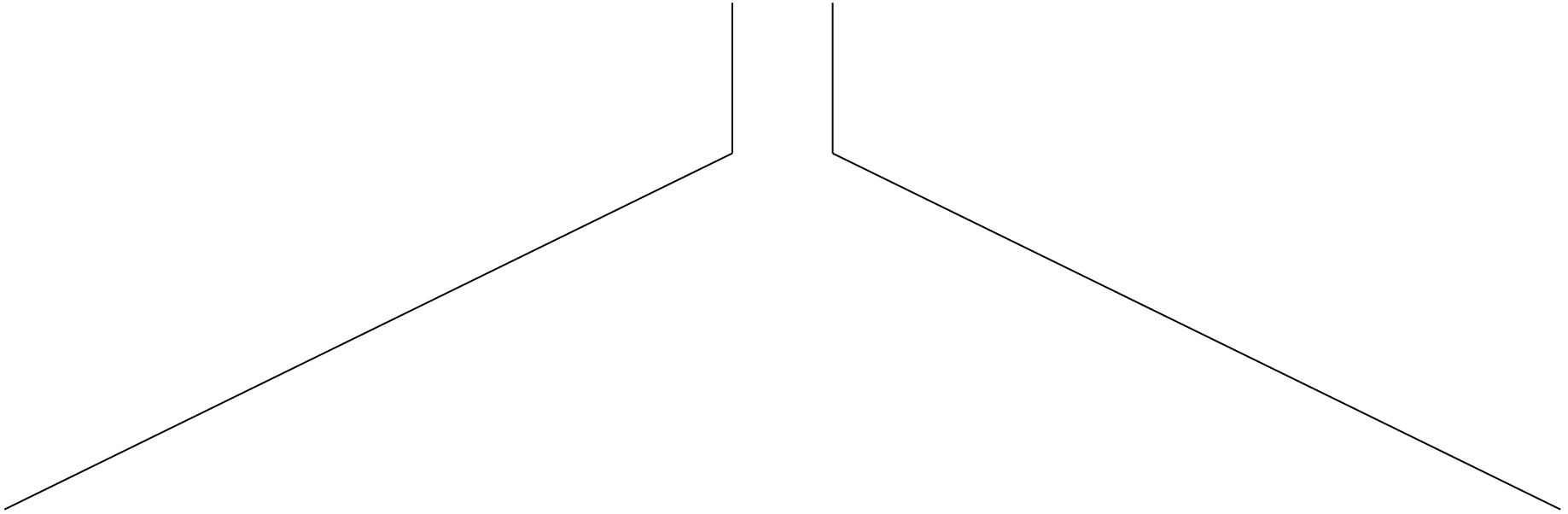
8	33
;	26
4	19
‡	16
)	16
*	13
5	12
6	11
(	10
†	8
1	8
0	6
9	5
2	5
:	4
3	4
?	3
¶	2
–	1
•	1



Implementation Strategy

The basic idea behind the program to count letter frequencies is to use an array with 26 elements to keep track of how many times each letter appears. As the program reads the text, it increments the array element that corresponds to each letter.

TWAS BRILLIG



1	1	0	0	0	0	1	0	2	0	0	2	0	0	0	0	0	1	1	1	0	0	1	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

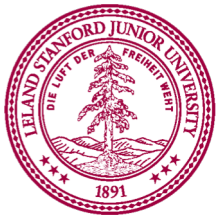
To the code!

Imagine a better world….

ArrayLists

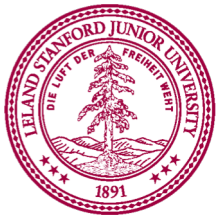
ArrayList

- An ordered, resizable list of information
- Homogeneous
- Can add and remove elements (among other cool functionality)
- Can store any **object** type
- Requires importing **java.util.***;



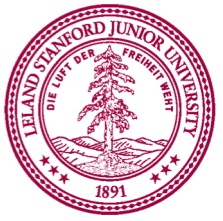
Our First ArrayList

```
ArrayList<String> myArrayList = new ArrayList<String>();
```

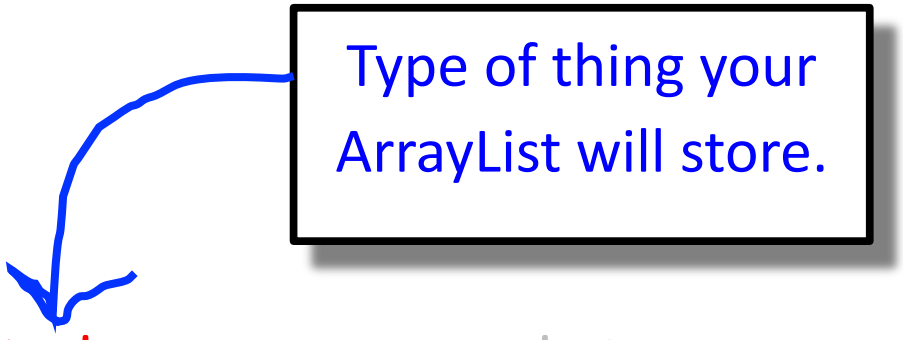



Our First ArrayList

```
ArrayList<String> myArrayList = new ArrayList<String>();
```

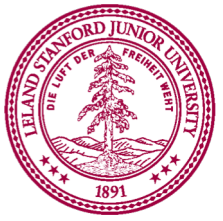
Our First ArrayList




```
ArrayList<String> myArrayList = new ArrayList<String>();
```

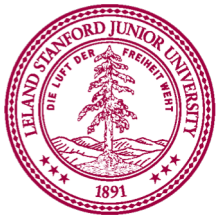
Our First ArrayList

```
ArrayList<String> myArrayList = new ArrayList<String>();
```

Our First ArrayList

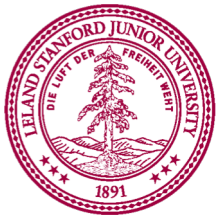
```
ArrayList<String> myArrayList = new ArrayList<String>();
```

Our First ArrayList

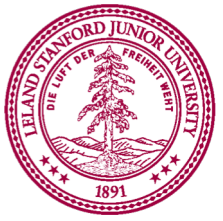
Same type here, but followed by ().

```
ArrayList<String> myArrayList = new ArrayList<String>();
```

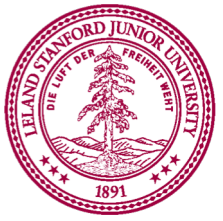
Our First ArrayList

```
ArrayList<String> myArrayList = new ArrayList<String>();
```

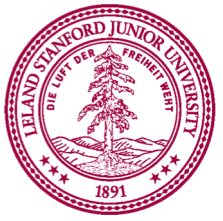
Our First ArrayList

```
ArrayList<String> myArrayList = new ArrayList<String>();

// Adds elements to the back
myArrayList.add("hi");
```

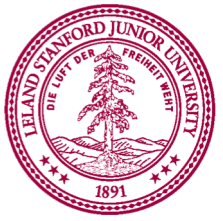
Our First ArrayList

```
ArrayList<String> myArrayList = new ArrayList<String>();

// Adds elements to the back
myArrayList.add("hi");
myArrayList.add("there");
```

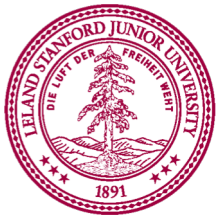
Our First ArrayList

```
ArrayList<String> myArrayList = new ArrayList<String>();

// Adds elements to the back
myArrayList.add("hi");
myArrayList.add("there");

// Access elements by index (starting at 0!)
println(myArrayList.get(0));  // prints "hi"
println(myArrayList.get(1));  // prints "there"
```



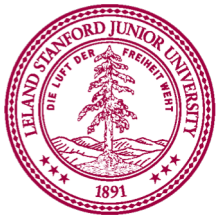
Our First ArrayList

```
ArrayList<String> myArrayList = new ArrayList<String>();

// Adds elements to the back
myArrayList.add("hi");
myArrayList.add("there");

// Access elements by index (starting at 0!)
println(myArrayList.get(0));  // prints "hi"
println(myArrayList.get(1));  // prints "there"

// Wrong type - bad times!  Won't compile
GLabel label = new GLabel("hi there");
myArrayList.add(label);
```

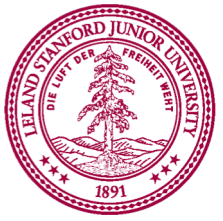
Our First ArrayList

```
ArrayList<String> myArrayList = new ArrayList<String>();

// Adds elements to the back
myArrayList.add("hi");
myArrayList.add("there");

// Access elements by index (starting at 0!)
println(myArrayList.get(0));  // prints "hi"
println(myArrayList.get(1));  // prints "there"

// Wrong type - bad times!  Won't compile
GLabel label = new GLabel("hi there");
myArrayList.add(label);

// Invalid index – crashes!  IndexOutOfBounds Exception
println(myArrayList.get(2));
```



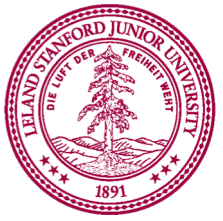
Our First ArrayList

```
ArrayList<String> myArrayList = new ArrayList<String>();

// Adds elements to the back
myArrayList.add("hi");
myArrayList.add("there");

// Access elements by index (starting at 0)
for (int i = 0; i < myArrayList.size(); i++) {
    String str = myArrayList.get(i);
    println(str);
}

// hi
// there
```

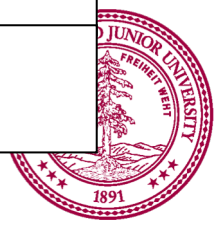
Our First ArrayList

```
ArrayList<String> myArrayList = new ArrayList<String>();

// Adds elements to the back
myArrayList.add("hi");
myArrayList.add("there");

// Access elements by index (starting at 0)
for (int i = 0; i < myArrayList.size(); i++) {
    String str = myArrayList.get(i);
    println(str);
}

// Beautiful way to access each element
for (String str : myArrayList) {
    println(str);
}
```



Methods in the `ArrayList` Class

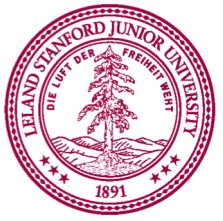
`boolean add(<T> element)` Adds a new element to the end of the **`ArrayList`**; the return value is always **`true`**.
`void add(int index, <T> element)` Inserts a new element into the **`ArrayList`** before the position specified by **`index`**.
`<T> remove(int index)` Removes the element at the specified position and returns that value.
`boolean remove(<T> element)` Removes the first instance of **`element`**, if it appears; returns **`true`** if a match is found.
`void clear()` Removes all elements from the **`ArrayList`**.
`int size()` Returns the number of elements in the **`ArrayList`**.
`<T> get(int index)` Returns the object at the specified index.
`<T> set(int index, <T> value)` Sets the element at the specified index to the new value and returns the old value.
`int indexOf(<T> value)` Returns the index of the first occurrence of the specified value, or −1 if it does not appear.
`boolean contains(<T> value)` Returns **`true`** if the **`ArrayList`** contains the specified value.
`boolean isEmpty()` Returns **`true`** if the **`ArrayList`** contains no elements.



ArrayLists + Primitives =

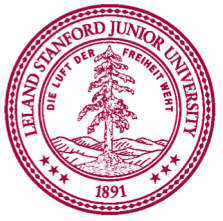
```
// Doesn't compile ☹
ArrayList<int> myArrayList = new ArrayList<int>();
```

ArrayLists can only store **objects!**



ArrayLists + Primitives =

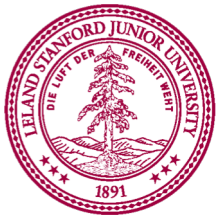
Primitive	"Wrapper" Class
`int`	`Integer`
`double`	`Double`
`boolean`	`Boolean`
`char`	`Character`



ArrayLists + Wrappers =

```
// Just use wrapper class when making an ArrayList
ArrayList<Integer> numList = new ArrayList<Integer>();

numList.add(123);
numList.add(546);

int firstNum = numList.get(0);     // 123
int secondNum = numList.get(1);    // 456
```

Conversion happens automatically!



lets see a simple example.

Data Structures

Operation	Arrays	ArrayListss
Make a new one	`int arr = new int[5];`	`ArrayList<String> list = new ArrayList<String>();`
Get length?	`arr.length`	`list.size();`
Get element?	`arr[i]`	`list.get(i);`
Set element?	`arr[i] = 5;`	`list.set(i, value);` `list.add(value);`
Loop?	`for(int i = 0; i < arr.length(); i++)`	`for(String value : list)`

Winter is over