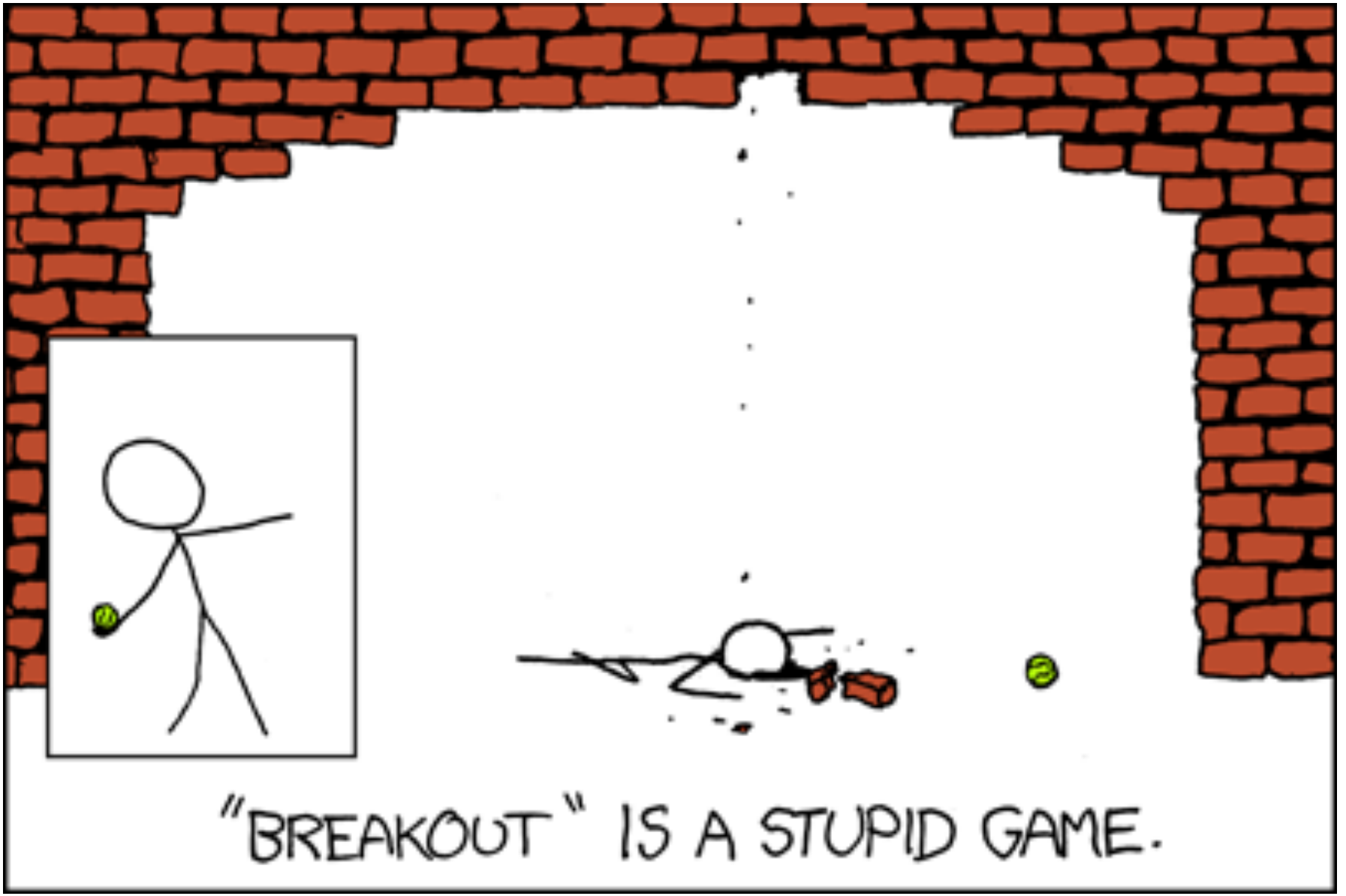
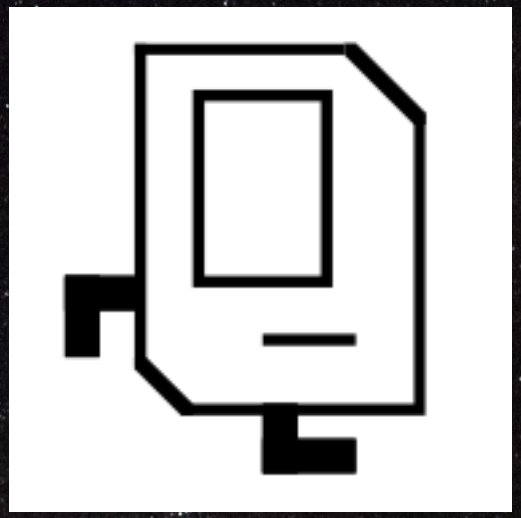


https://xkcd.com/347/

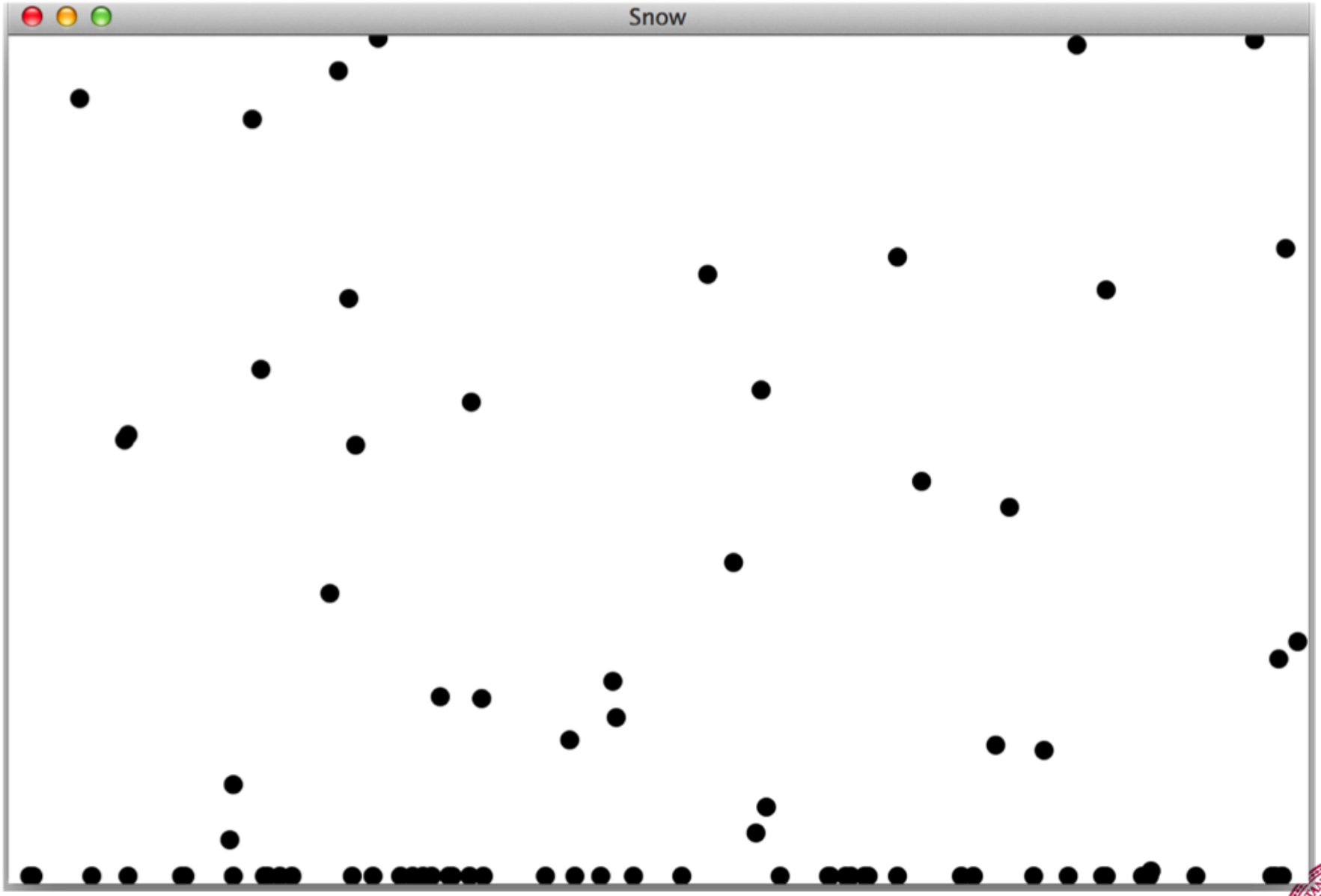


Aka Karel Wars Episode VII

ArrayLists

Chris Piech
CS106A, Stanford University

Why is this hard to write?



Learning Goals

1. Know how to store and retrieve data from an ArrayList.

File Processing

Thanks Keith Schwarz for some great slides to build off!

File concepts in one slide

1. Make a Scanner (lets call it input) to open a file for reading

```
Scanner input = new Scanner(new File("poem.txt"));
```

2. Use scanner.nextLine to get one line from the file

```
input.nextLine(); // returns the next line
```

3. Both the above operations are “dangerous” so we need to use a try/catch loop

```
try{
   // live dangerously
} catch (Exception e){
   // have heath insurance
}
```

4. You can either handle the problem or throw a runtime exception

```
throw new RuntimeException(“AHHHH!”);
```

The classic file reading program.

- The idiomatic "read all the lines of a file" code is shown here:

```
try {
    Scanner input = /*…open the file… */
    while (input.hasNextLine()) {
        String line = input.nextLine();
        /* … process current line … */
    }
    input.close();
} catch (IOException e) {
    throw new RuntimeException(e);
}
```

Understanding this code is about 95% of what we want you to know for files in CS106A

ArrayLists

Thanks Keith Schwarz for some great slides to build off!

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

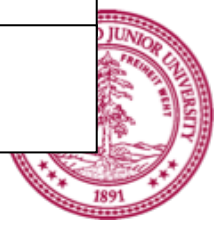
```
ArrayList<String> myArrayList = new ArrayList<String>();

// 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");

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

// hi
// there
```

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!

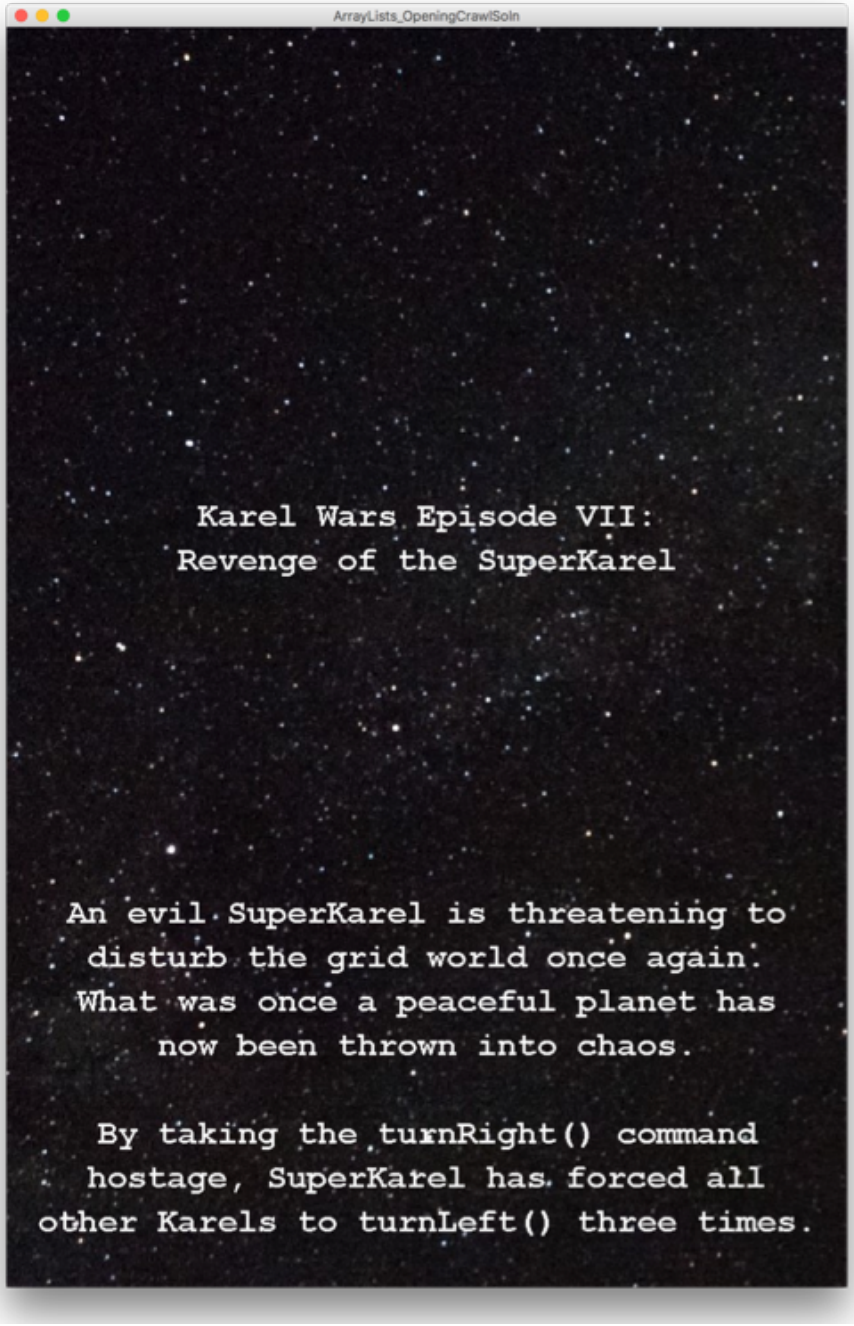
lets see a simple example.

Thanks Keith Schwarz for some great slides to build off!

lets make it snow.

Thanks Keith Schwarz for some great slides to build off!

Demo: File Processing + ArrayLists



Thanks to Nick Troccoli for the awesome example

Summary of Today

- ArrayLists are homogeneous lists of objects.
- You can add, remove, get, find, etc. on ArrayLists.
- Having a variable that is a collection of other variables allows you to solve more problems.

Good luck on the midterm!