

# Lecture 26: Yet More Lists

---

CS 51G  
Spring 2018  
Kim Bruce

# Announcements

- Exercise 13.8.2
- Quiz:
  - GUI
  - No recursion or for loops
- Recursion lab

# Problems

- Given a list of Strings, myList, held in alphabetic order, and a new String s, find at which index s should be inserted.
- Given an index of where to put an element, s, move other elements to right one slot and then insert s.
- Given a list of Strings, myList, held in alphabetical order, and a new String s, update myList to include s and still be in order

Solutions at:

<http://www.cs.pomona.edu/classes/cs051G/demos/InsertList/insertList.grace>

# Laundry — Again?

- Can use lists and inheritance to make laundry easier.
  - Clothes objects have two lists: solid and framed items.
  - Abstract class can define all methods
  - Pants and T-shirt classes just construct objects

Lists of Lists of ...

# Multi-dimensional Lists

```
def d: List[List[Number]] = emptyList[List[Number]]
```

or

```
def d': List[List[Number]] = list[List[Number]] []
```

```
d.add (list[Number]{2,4,6})
```

```
d.add (list[Number]{1, 3, 5, 7})
```

```
print (d)
```

```
for (d) do {val: List[Number] → print("next is: {val}")}
```

# If that makes your head hurt

```
type NumList = List[[Number]]
def d': List[[NumList]] = emptyList[[NumList]]

d'.add (list[[Number]][2,4,6])
d'.add (list[[Number]][1, 3, 5, 7])

print (d')
for (d') do {val: List[[Number]] → print("next is: {val}")}
```



# Matrix

- Define type and class to represent square matrices
  - Formed as a list of rows
- Example of 5 x 3 matrix:

	1	2	3
1	<code>a.at(1).at(1)</code>	<code>a.at(1).at(2)</code>	<code>a.at(1).at(3)</code>
2	<code>a.at(2).at(1)</code>	<code>a.at(2).at(2)</code>	<code>a.at(2).at(3)</code>
3	<code>a.at(3).at(1)</code>	<code>a.at(3).at(2)</code>	<code>a.at(3).at(3)</code>
4	<code>a.at(4).at(1)</code>	<code>a.at(4).at(2)</code>	<code>a.at(4).at(3)</code>
5	<code>a.at(5).at(1)</code>	<code>a.at(5).at(2)</code>	<code>a.at(5).at(3)</code>

# Defining Matrix

```
type Matrix[[T]] = {  
  at (r: Number, c: Number) -> T  
  at (r: Number, c: Number) put (value:T) -> Done  
}
```

- See class defined at
  - <http://www.cs.pomona.edu/classes/cs051G/demos/Matrix/Matrix.grace>
- Allows you to ignore implementation as list of lists.
- Must provide default value for each slot

# Puzzle

- Suppose define:
  - `def a:Matrix[[Number]] =  
matrixSize[[Number]](5,3) default Value (0)`
  - What is result of executing:
    - `for (i..5) do {row: Number ->  
for (i..3) do { col: Number ->  
a.at(row,col)put(4*(row-1) + col)  
}  
}`

# Tic-Tac-Toe

- See program

Questions?