MVC: Model, View, Controller

A model is the state and brains of a system

In a game it's all the pieces and where they are
 In a spreadsheet it's the data and the formulae

- * The view is how we look at the model
 - □ Spread sheet has graphs, charts, cells, text, ...
 - **Game has board, number of opponents, hit-points, ...**

* When the model changes, the views reflect the changes

- □ The model tells the views how/if it has changed
- □ Model sends information to views OR
- View asks model for information

Com	nSci	100F	
COM	pour	TOOL	

10.1

MVC: interfaces and inheritance

A model might have multiple views

- □ Tell all the views "I've changed"
- □ Who manages the views? This requires state: store views
- □ Why can't we keep this state in an interface?

See IModel and AbstractModel

- **One specifies behavior, the other provides default**
- Don't rewrite code if we don't have to, maintaining views will be the same for all models

See IView and SimpleView

□ No default/shared view state/behavior: text and GUI

	100E

Does SimpleViewer know Model?

- What does the SimpleViewer know about its model?
 - □ If we look at code, is there any application-specific logic?
 - □ What if we wanted to play a game, start a new game?
- * Control in MVC with SimpleViewer and IModel
 - Loading a file calls initialize()
 - □ Entering text calls process()
 - □ Model calls view with messages, errors, and complete update

* This isn't complete general, but it's pretty generic

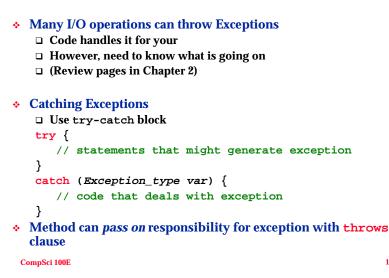
□ For this input, here's the output

Pixmap Assignment

- Traditional "Last" CompSci 6 Assignment
 - Lots has been done for you
 - □ Mainly an exercise in working with 2 D info
- Not really MVC
 - **Doesn't hurt to keep that model in mind, though**
- Lots of GUI stuff
 - **Graphical User Interface is not reall focus of this course**
 - □ Just use what has been given
 - **D** Become familiar with it by reading code, seeing results

* Feel free to experiment

Java Exceptions



Stack: What problems does it solve?

- * Stacks are used to avoid recursion, a stack can replace the implicit/actual stack of functions called recursively
- * Stacks are used to evaluate arithmetic expressions, to implement compilers, to implement interpreters
 - **□** The Java Virtual Machine (JVM) is a stack-based machine
 - **D** Postscript is a stack-based language
 - Stacks are used to evaluate arithmetic expressions in many languages
- * Small set of operations: LIFO or last in is first out access
 - Operations: push, pop, top, create, clear, size
 - □ More in postscript, e.g., swap, dup, rotate, ...

```
CompSci 100E
```

10.5

10.7

Simple stack example

CompSci 100E

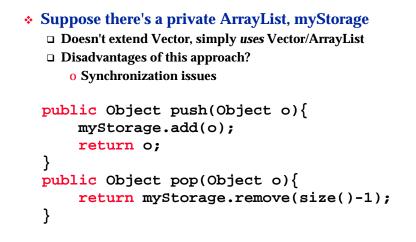
```
$ Stack is part of java.util.Collections hierarchy
    It's an OO abomination, extends Vector (like ArrayList)
        o Should be implemented using Vector
        o Doesn't model "is-a" inheritance
        What does pop do? What does push do?
    Stack s = new Stack();
    s.push("panda");
    s.push(tgoi=zšypop();
    s.push(tgoi=zšypop();
    s.push("brown");
    System.out.println("size = " + s.size());
    System.out.println(s.peek());
    Objec
    System.out.println(s.peek());
```

Implementation is very simple

- Extends Vector, so simply wraps Vector/ArrayList methods in better names
 - □ push==add, pop==remove
 - □ Note: code below for ArrayList, Vector is actually used.

```
public Object push(Object o){
    add(o);
    return o;
}
public Object pop(Object o){
    return remove(size()-1);
}
```

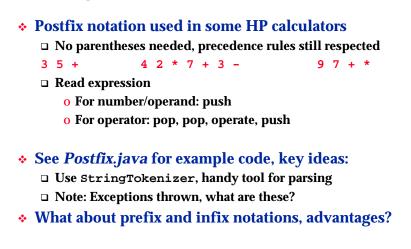
Uses rather than "is-a"



CompSci 100E

10.9

Postfix, prefix, and infix notation



CompSci 100E

Exceptions

* Exceptions are *raised* or *thrown* in exceptional cases

- □ Bad indexes, null pointers, illegal arguments, ...
- □ File not found, URL malformed, ...

* Runtime exceptions aren't meant to be handled or *caught*

- **D** Bad index in array, don't try to handle this in code
- □ Null pointer stops your program, don't code that way!
- * Other exceptions must be caught or rethrown
 - □ See FileNotFoundException and IOException in Scanner class implementation
- RuntimeException extends Exception, catch not required

Prefix notation in action

 Scheme/LISP and other functional languages tend to use a prefix notation

(define (square x) (* x x))

Postfix notation in action

- * Practical example of use of stack abstraction
- Put operator after operands in expression
 - □ Use stack to evaluate
 - o operand: push onto stack
 - o operator: pop operands push result
- PostScript is a stack language mostly used for printing
 drawing an "X" with two equivalent sets of code

%!	81	
200 200 moveto		
100 100 rlineto	100 -100 200 300 100 100 200 200	
200 300 moveto	moveto rlineto moveto rlineto	
100 -100 rlineto	stroke showpage	
stroke showpage		

CompSci 100E