

Topdown v Bottomup

- Programming is changing our world
 - Empowering, liberating, equalizing,...
- Everything is a bit: all 0's and 1's
 - From jpg to mp3 to ...
- It's about problems! It's about details!
 - Should we think about problems to get to the details?
 - Should we master details before grand thinking?
- See [Wikipedia on topdown v bottomup design](#)

Compsci 100, Fall 2009

3.1

Conventions in Compsci 100 projects

- We want you to concentrate on algorithms and data structures
 - Not on rendering fonts, interacting with users
 - This is important! But not what this course is about
- We try to build GUIs or views that facilitate projects
 - You write the brains, we build the skull/manikin
 - Our GUI communicates with your code
 - Requires following conventions in interacting code
- GUI libraries are similar across languages, but...
 - Deeper Java specific details than HashMap

Compsci 100, Fall 2009

3.2

KWIC: Key word in Context

```
Arise, fair sun, and kill the envious moon, Who
I. Yet I should kill thee with much cherishing.
shortly, for one would kill the other. Thou! why,
those twenty could but kill one life. I beg
wherefore, villain, didst thou kill my cousin? That villain
mean, But 'banished' to kill me- 'banished'? O friar,
thou happy. Tybalt would kill thee, But thou slewest
cell there would she kill herself. Then gave I
heaven finds means to kill your joys with love!
```

- Read file, find word and it's context, print
 - Can find all words, but how do we get context?
 - Loop and look: create context line for each occurrence
 - See `ContextModel.java`

Compsci 100, Fall 2009

3.3

Use KWIC example to motivate study

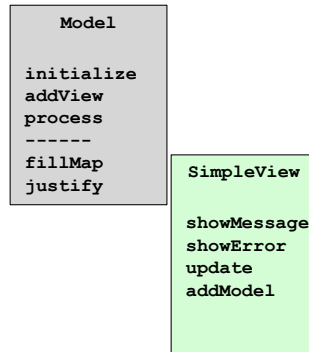
- Dissect and inspect KWIC code to understand conventions
 - Understand Model and View interaction
 - Facilitates doing Markov text-generation assignment
- Review some basic coding idioms and ideas
 - Avoiding recomputing same value, readability, modifiability, ...
- Errors: possible for a method to fail but still work?
 - See KWIC code when same key occurs twice on a line!

Compsci 100, Fall 2009

3.4

MVC Example, key-word-in-context

- **User loads file**
 - Where? Communicate to?
 - What changes in model?
 - What happens in view?
- **User chooses word**
 - Process in Model
 - Alternatives?
 - Generate context, display
 - How to show in any view?



Key Word in Context Explained

- **For every different word, store where it occurs**
 - *love* is the 1st, 3rd, 50th, and 1237th word in the file
- **This data is kept in a map, key is word, value is ??**
 - How do we generate the data in the map?
- **Given a word, how do we find its context? How do we format?**
 - All words are in an array, in order
 - Memory concerns?
 - Original KWIC paper by Parnas as comparison

Code Interlude

- **Examine ContextModel.process**
 - Called when user enters word, parameter is the word
 - If file already read, we don't need map, where is this?
 - Error checking? When and when happens
 - How does Model communicate back to View?
- **Examine ContextModel.justify**
 - What is method doing
 - What is parameter, where was it constructed, issues?
 - What about 'magic numbers', e.g., 30?
 - What about comments, should we add some?

KWIC main program/class

```
public class ContextMain {
    public static void main(String[] args){
        ContextModel model = new ContextModel();
        SimpleViewer view =
            new SimpleViewer("Compsci 100 KWIC", "word>");
        view.setModel(model);
    }
}
```

- **What changes in above, e.g., for Markov assignment?**
 - How can view communicate with *any* model?
 - View doesn't change, model does!
 - Requires using a Java interface to capture commonality

Model View Controller, MVC

- **Gui is the View and often the controller**
 - Separate user-interaction from updates to data
- **User loads file, chooses word, ...**
 - Model notified, computes, updates view
- **Model has all the state and knows when it changes**
 - Communicates changes to views (via controller)
 - Must be initialized, updated, etc.

- **Very common *Design Pattern***
 - Capture common solutions to problems in a context
 - Iterator, Composite, Decorator seen in Compsci 100

Compsci 100, Fall 2009

3.9

Convention Summary

- **Classes start with capital letter and then we have:**
 - They're public, except nested class? Protected means ...
 - camelCaseForMethods and ForClasses
 - Ivars, fields, instance variables, mySize, myMap, ...
 - Constants (public static) are ALL_CAPS

- **Interfaces are IModel, IView, and so on**
 - Not true for standard Java classes, yes for Compsci 100
 - Don't need to label methods as abstract, but can

- **Supply AbstractDefault implements IThing**
 - Constructor, some state, some common behavior: extend!

Compsci 100, Fall 2009

3.10

Eugene (Gene) Myers

- **Lead computer scientist/
software engineer at
Celera Genomics, then at
Berkeley, now at Janelia
Farms Research Institute
(HHMI)**

- **BLAST and WG-Shotgun**



"What really astounds me is the architecture of life. The system is extremely complex. It's like it was designed." ... "There's a huge intelligence there."

Compsci 100, Fall 2009

3.11

Methods, Interfaces, Inheritance

- **A method by any other name would smell as sweet**
 - Method in OO languages, functions, procedures in others
 - Parameters and return value: communication
 - Do objects or methods communicate?: OO v procedural

- **Static : `Math.sqrt`, `Character.isUpperCase`, ...**
 - Don't belong to an object, invoked via class (clue above?)
 - Java API helpful here

- **Interface: implement class with required, related methods**
 - HashMap, TreeMap
 - ArrayList, LinkedList, Vector

Compsci 100, Fall 2009

3.12

Interfaces continued

- **In the beginning**
 - Make code work, don't worry about generalizing
 - But, if you write code using `Map` rather than `TreeMap`
 - Can swap in a new implementation, coded generally!
- **Don't know how to optimize: space or time**
 - Facilitate change: use interface rather than concrete class
 - My DVD connects to my TV, regardless of brand, why?
 - How do you turn on a Nokia cell phone? Motorola? But!
- **Interfaces facilitate code refactoring**
 - Don't add functionality, change speed or memory or ...

Compsci 100, Fall 2009

3.13

What does Object-Oriented mean?

- **Very common method of organizing code**
 - Design classes, which encapsulate state and behavior
 - Some classes can be similar to, but different from their parent class: inheritance
 - Super class, subclass
 - Inherit behavior, use as is or modify and use *or both*
- **Complex to design a hierarchy of classes, but important**
 - More of this in Compsci 108 or on-the-job training
 - We're solving simple problems, not designing re-usable libraries
- **Simple does not mean straight-forward, not Vista!**

Compsci 100, Fall 2009

3.14

Inheritance and Interfaces

- **Interfaces provide method names and parameters**
 - The method signature we can expect and thus use!
 - What can we do to an `ArrayList`? To a `LinkedList`?
 - What can we do to a `Map` or `Set` or `PriorityQueue`?
 - `java.util.Collection` is an interface
- **Abstract classes can implement core, duplicated code**
 - If we can add one object to a [set,map,list], can we add an entire list of objects? `java.util.AbstractCollection`
 - If we can iterate can we remove? Convert to array? Obtain size?

Compsci 100, Fall 2009

3.15

Random Java stuff

- **What happens when you call `new`, when do you?**
 - Creates object, assign reference/pointer somewhere (or?)
 - Two 'labels/variables' share same object, consequences?
 - Why isn't this a concern with `String` objects/labels?
- **What about `int`, `double`, `byte`, `char`, `long`**
 - Related, but different. What's the same, what's different?
 - Range of values for each?
 - Arithmetic with each?
 - Casting vs promotion (pass `int` to `Math.sqrt`?)

Compsci 100, Fall 2009

3.16