Software Development

- software challenge

psychologist George Miller: http://www.musanim.com/miller1956/

The magical number 7

“the seven seas, seven deadly sins, seven ages of man, seven primary colors, seven notes of the musical scale, seven objects in the span of attention, and the seven digits in the span of immediate memory?”

- abstraction tools

Problem Decomposition:
“Modules” = “smaller pieces”

- Self-contained program part: encapsulation or “packaging”
  - logical (groups smaller related pieces)
  - physical (separate file)

- program design/build/test process is more manageable
  separate compilation unit is faster, teams

- “off -the-shelf ” components

- information hiding (abstraction), encapsulation (protection)

Problem Decomposition
What “Modules” to Build ?

- **Focus on tasks** (procedures) Top-down Design
  What tasks need to be done?
  (What functions to build?, When to call each one…flow?)
  (What data is required by the function(s) to do the task?)
  (see crazy8’s)
- Focus on data Abstract Data Types (ADT)
  set of values
  (What are the values expected?)
  operations on the values
  (What basic operations would need to be performed on the values?)

Software Development

Problem decomposition techniques using abstraction mechanisms.

What does the word abstraction mean?

Have you ever abstracted before?

What is a high level language?

Software Development

Problem decomposition techniques using abstraction mechanisms.

Procedural (functional) abstraction

building one function at a time with the larger problem first

(top-down design)

calling functions not written yet (what task, what data)

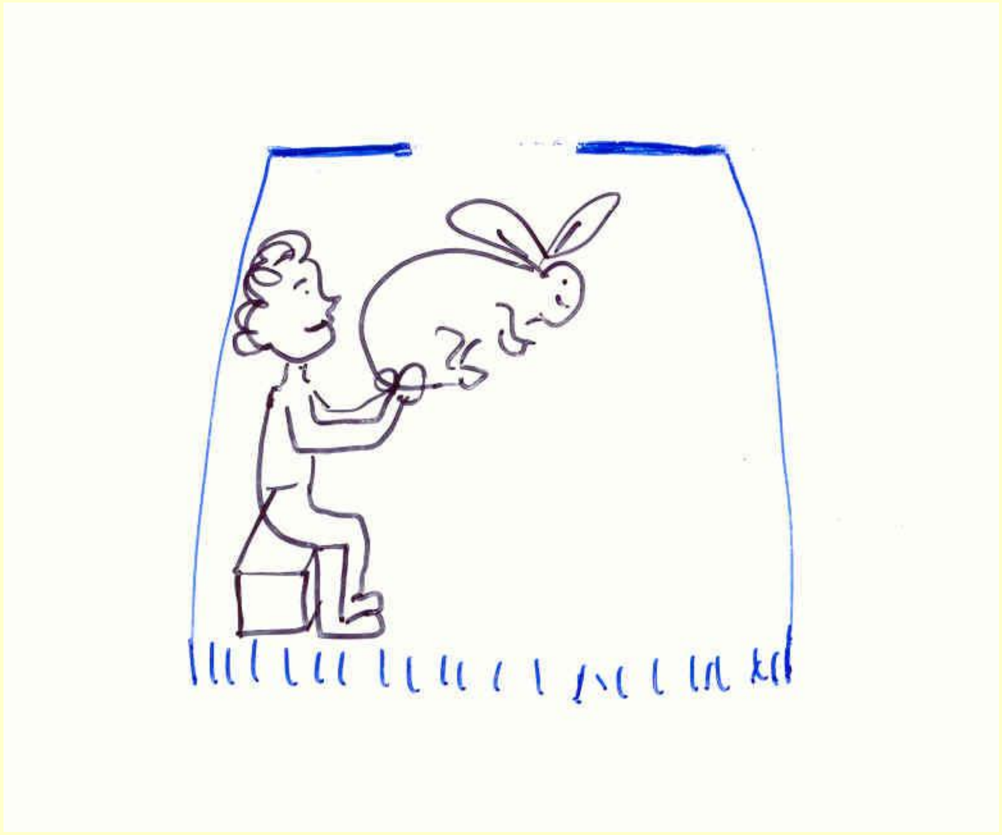
(specify them carefully but don't write code)

algorithmic flow

Interface (Specification)
(focus on "What" -- the contract)
procedural abstraction

Implementation (focus on "How")

code: for data storage, logic to make it happen



NEED: Separation between the “interface” and the “implementation”

why? manage complexity for HUMANS
provide abstraction of some details
(information hiding)
security

Interface and Implementation

Abstraction barrier

(Specification)
(interface)
What is the task?

What data is needed?

What data is produced or changed?
Contractual agreement.
Javadoc
(documentation)
programmer perspective
user perspective
Public

Implementation
(code)

How is the task actually coded?
java function/class/file/..
the “magic” happens here!!
How is the input data used?
specific java data types

program deliverable to meet contract.
programmer perspective
Private
May be > 1 implementation for the <u>same</u> interface

Crazy 8's

Top-down design

Begin at larger problem and "chunk" it into functions calls.

Code ONE function at a time while "pretending" to have helper functions and calling them when appropriate.

-specify the functions to be called (WHAT will they do?)

-show flow of function calls (When are they called?)

Stepwise refine: same process at the next (lower) level of functions.

Craps Problem

Top-down design

First roll of 7 or 11, you win outright.
First roll of 2,3, or 12, you lose outright.
First roll of 4..10, that becomes your point number
and you keep rolling until you either roll a 7 or you roll your point number again. You win if you roll your point number before you roll a 7. If you roll a 7 before the point number, you lose.