- Why do we not serve answers on silver platters
- C
- D
- Uncle Sam Wants You
- Software Systems
- Developing Software Systems: HtDS from HtDP

It would be easy for me to have TAs
serve answers to all your questions
on Piazza and during office hours—just
like your F I II or OOD instructor might
have done. It would make my life
peaceful. It would make their lives even
easier. And grading might just be a breeze.

Q: Why do you think I am not doing this?

Q: Why do you think the Sw Dev staff does this?

C.

We will use streams ofJSON as a language for communicating between game servers and client players.

Are there any remaining questions about echoing a number of JSON expressions from STDIN to STDOUT?

D. The project will need GUIs.

to volunteer for code walks,
as panelists and/or presenters.

Software Systems come with four attributes:

- a start-up phase (set up all components)
- a steady-state phase (run & provide services)
- a shut-down phase (except for “eternal severes”)
- a configuration (a system or its components need to adjust)

plus graceful failure

How to Scale the Design Ideas from F I, F II, and OOD

For every task, formulate one method/function.
Every function/method is dedicated to one task.
(Then compose such functions/methods.)

If your data definitions is “hierarchical” (one refers to another) for every task design one function per data definition and have them call each other according to their references.

1. figure out the data that your units of code work on
2. articulate the purpose of every unit of code in your words
3. work through examples; keep the context in mind
4. lay out what you have; that’s what your code may use
5. *now code*
6. turn the examples into unit tests
7. [[if the code is ‘generative’ (ex. ‘while’) does it terminate?]]

decompose tasks

For every task, formulate one method/function.
Every function/method is dedicated to one task.
(Then compose such functions/methods.)

follow the data

If your data definitions is “hierarchical” (one refers to another) for every task design one function per data definition and have them call each other according to their references.

most basic design process

1. figure out the data that your units of code work on
2. articulate the purpose of every unit of code in your words
3. work through examples; keep the context in mind
4. lay out what you have; that’s what your code may use
5. *now code*
6. turn the examples into unit tests
7. [[if the code is ‘generative’ (ex. ‘while’) does it terminate?]]

For every task, formulate one method/function.
Every function/method is dedicated to one task.
(Then compose such functions/methods.)

Read some number of XML objects from a stream until it is closed, determine the average of their numeric size attributes, and write the average to an output stream.

every unit of code has one distinct purpose

function, method, class, module, package

interpretation

1. figure out the data that your units of code work on
2. articulate the purpose of every unit of code in your words
3. work through examples; keep the context in mind
4. lay out what you have; that's what your code may use
5. *now code*
6. turn the examples into unit tests
7. [[if the code is 'generative' (ex. 'while') does it terminate?]]

If your data definitions is “hierarchical” (one refers to another) for every task design one function per data definition and have them call each other according to their references.

A JSON array is a comma-separated sequence of JSON “thingies”. How do you compute the corresponding JSON array of numeric attributes of each JSON “thing”.