Programming & Computational Thinking

Abstraction & Pair Programming

Computational Thinking

A bit of *a* computing *buzz-word* right now.

It's not going to be *a* term you're likely to define in *a* job interview, but the concepts that you'll learn *as* we learn to do it, those might be...

So what is it?

A process that generalizes a solution to open ended problems. Open-ended problems encourage full, meaningful answers based on multiple variables, which require using decomposition, data representation, generalization, modeling, and algorithms found in Computational Thinking. ~Wikipedia

Key Concepts

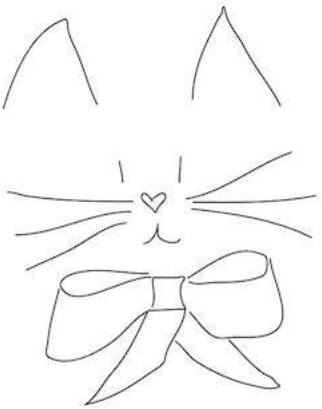
- abstraction
- generalization
- composition & decomposition
- creativity
- data and information
- algorithms

Abstraction

Our first key topic!

What do you think it means?

Meet Bart



Abstract Bart

- My cat sketch is an abstraction.
- It's a cat, but you probably wouldn't know it was Bart and not some other cat.
- The details have been removed.

Can you think of some other examples?

Abstraction in Computing

In computer science, abstraction is a technique for managing complexity of computer systems. It works by establishing a level of complexity on which a person interacts with the system, suppressing the more complex details below the current level.
~Wikipedia

Two areas of focus

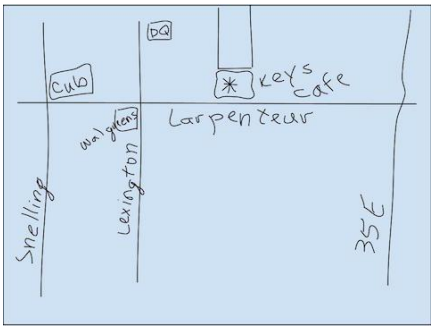
- Data representation
  - Examples: network diagrams, icons on a desktop, dots on a radar
- Systems interactions - interfaces
  - Examples: communication protocols, APIs (application programming interfaces), project development methodologies

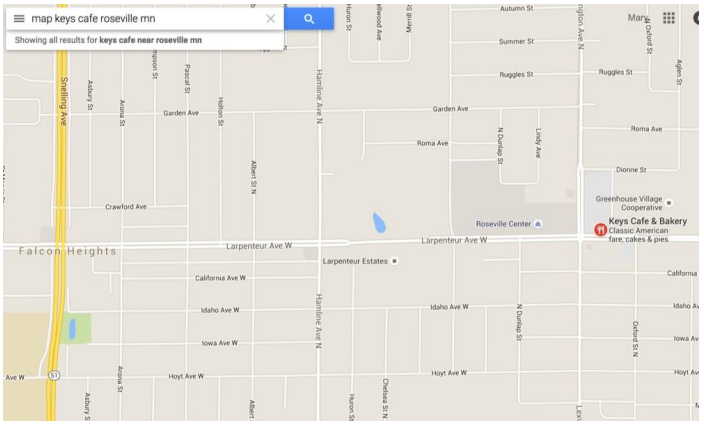
Data Abstraction

Quick exercise! 60 seconds

Draw me a map to your favorite restaurant.

Data Abstraction on Maps





Systems Abstraction

Think about driving a car...

Car Interface

Turn key – start car

Gas Pedal – go

Brake Pedal – stop

Steering Wheel – turn right or left

But what really happens when you do these things? What happens under the hood?

Hiding details is important

Cars have changed a lot over time, but how much has the way we drive them changed?

Abstraction allows you to change the underlying details, *the implementation*, without affecting the way the system is used.

Pair Programming

- An industry practice popular in Agile development.
- Simply put, you take
  - two people
  - give them one computer, keyboard and mouse,
  - and they work together on a single assignment

Why?

- Two heads are better than one.
- There is a sounding board for ideas
- Fewer mistakes or bugs in the final code
- Better, easier to read code
- More people know more of the code
- “Pair pressure” encourages good performance, and less slacking off

Driver & Navigator

There are two roles:

- The Driver's role is tactical, to focus on the mechanics of operating the computer and entering code.
- The Navigator's role is strategic, to think about what needs to be done and plan ahead to get there.

Switching roles

- In the early labs, there are reminders.
- Later, the reminders go away, but you still need to switch, ideally every 10-15 min.
- Do it as it feels natural, but do it.
- It might seem awkward at first, but you'll learn to feel comfortable switching roles at any time.

Be patient & respectful

- No cell phones in lab!
- Work together, not side-by-side.
- Constant communication is key.
- You will learn by explaining your thoughts and rational to someone else.
- Ask questions, challenge your pair.

Take breaks

- It's a long class, step away for a bit.
- Do not work ahead without your pair!
- Stop together and agree on when to return and start up again.
- Be respectful of your partner and do not be late coming back.

Labs - Learn by doing

- I'm not going to lecture on how to code.
- Labs will introduce you to the topics as you put them into practice.
- Take your time and make sure you understand the material as you go.

Be curious. Experiment. Ask "what if".

Lab Advice

- Don't skip stuff!
- One person should log into Snap! to start.
- Save often to the cloud.
- It's OK to save everything in one program.
- To get *a clean* script, make a new sprite.
- At the end of lab, log out, have the other person log in and save to their cloud.

Questions in lab

- Before you ask me a question about the lab, you must first ask two other pairs.
- Be respectful about interrupting. Let them know you have a question and allow them to finish a thought if needed.
- If you have no answer after asking two other pairs, put up the red flag.

Lab check-off

- When you are ready for me to grade your in-class lab, put up the green flag.
- I'll look at at some of your code and look over and collect your lab worksheet.
- See another flag? Review your answers to worksheet with another pair while waiting.