Basic Ideas

Hunter Jorgensen

Tools To Get Started

- For this course, you will need 3 tools to get you started:
    - A Web Browser (I recommend Firefox or Chrome)
    - A text editor (choose one of the following)
        - VS Code is industry standard for web developers. This is the most popular IDE
        - Sublime/Atom: a very lightweight editor that, with some plugins, can be very powerful
        - WebStorm/IntelliJ EE: a very heavy IDE, but is free with your student email address from the JetBrains
    - Git for source control
        - You are welcome to use the desktop app or the CLI (I recommend using the CLI, but I know it is not as good on Windows)

Git

- Git is a source control program.
- It is used almost universally by technology companies to allow developers to work on similar code bases without running into problem, to see who has made what changes and to undo mistakes or bugs
- Git works by tracking the changes in a code base - this is known as a commit. The current ‘commit’ is a collection of all changes that have come before it
- When you make a code change, your new changes are `untracked` - this means that it is visible on your local computer but not anywhere else.
- When you `add` a file, you are staging it so that you know which pieces of code are going to be part of the change
- When you `commit`, it adds all staged files to a single commit, ready to be pushed to the shared/remote repository
- When you `push` a commit, you are sending to another repository (typically the master remote one; i.e., the one you see on Github.com). This makes official your change and your code is effectively marked as complete

Important Git Commands

Our usage of Git in this course is pretty basic, but it’s helpful to know what you’re typing and why it matters. These are some important commands:

Move all current changes into staging: `git add *`

The * means that you add all files in the current directory and any subdirectories

Put all changes into a commit: `git commit -am “{commit-message}”`

The -a and -m are 2 separate pieces. The -a mean’s all files that have been previously committed or are in staging; the -m means to include the following string as a commit message.

Update your remote (i.e. Github) repository with your changes: `git push` (sometimes `git push origin master`)

This will update your code in Github sot that others can read or use it.

Data Modeling

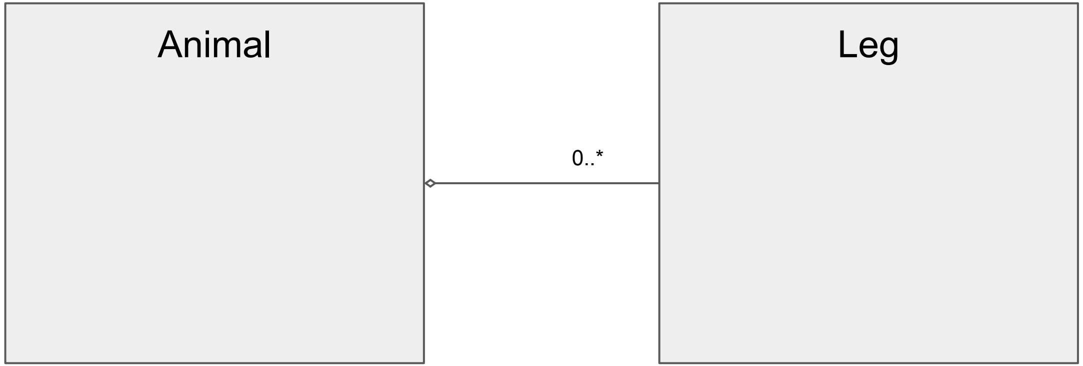
When we represent entities in code, we make assumptions about its properties, state and/or attribute:

Animal

- Number of legs
- What it eats
- Name
- Class/Genus
- Color
- etc.

Data Modeling

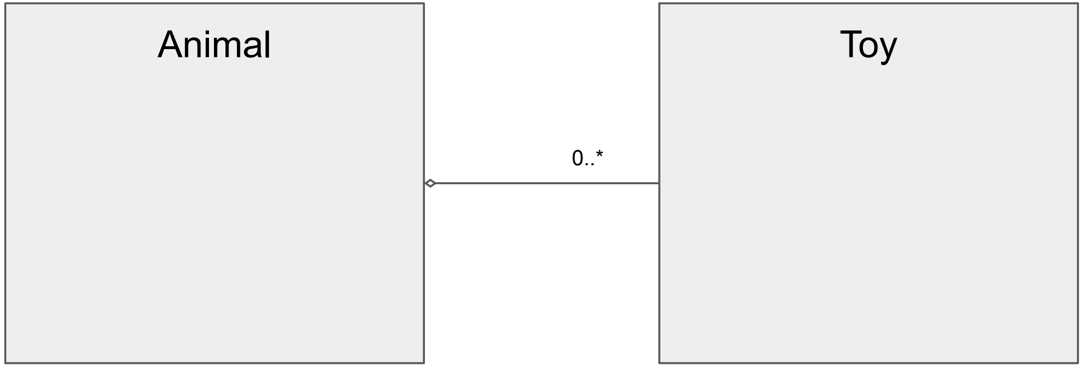
But entities don’t exist in a vacuum: things can be ‘made-up’ of other things, own several things, etc. This can get pretty complicated and change depending on what we’re trying to accomplish. For instance, we can reshow an animal as:



This means an animal can be made up of, or ‘composed’, of 0 to many legs (among other parts)

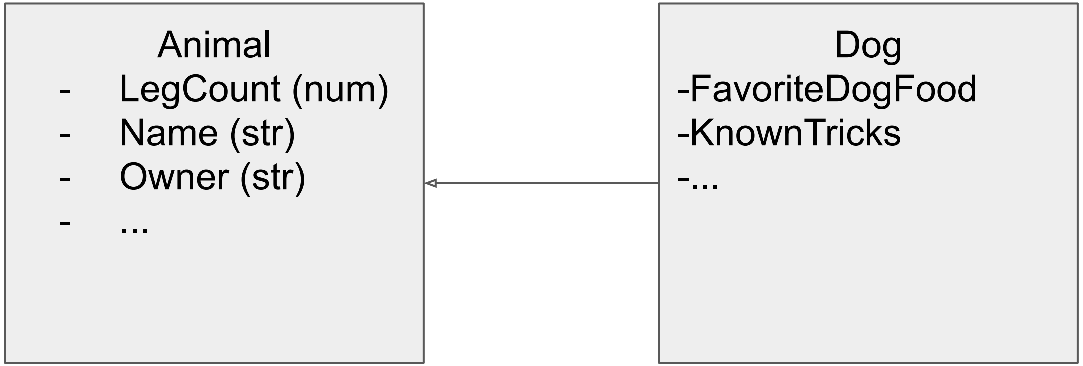
Data Modeling

We can also model what entities ‘possess’ other entities. Here an animal might have 0 to unlimited toys, but the toys are not dependent on the animal. This is a has-a relationship, also known as aggregation:



Data Modeling

Finally, sometimes we can minimize duplicate information by showing how some data is shared across common entities. For instance, all animals have names or a number of legs, but only dogs can know tricks. This is known as inheritance, or generalization:



Command Line Interface or Terminal

Using a terminal can be tricky, but it’s just another way that a computer represents files and directories: instead of visually, like in a standard OS, it lists it out with strings and words.

Using a terminal can be a quick way to move through a system and get the computer to do what you want.

Important Commands for the Terminal

Change Directory: `cd {directory name}`

Go Up a Directory: `cd ..`

Create a new File: `touch {filename}`

Create a new Directory: `mkdir {directory name}`

**Note the distinction between a directory (or folder) and a file!

Print a File to Console: `cat {filename} or less {filename}`