Lec 20: Abstract Classes

Sam McCauley

April 10, 2026

Admin

- Reminder: midterm 2 a week from today

- Any questions?

Inheritance Example

Inheritance

- Inheritance is not just about extending functionality
- It can also be used for *organizing* code
- Let’s look at a project which could easily be written in a single class. We’ll write it using inheritance instead

Design Plan

- We'll have a `Cipher` class with useful methods for all three ciphers
    - Its encrypt and decrypt methods won't do anything: the message will just map to itself
    - It will also have our `main` method, and a helper method for testing
- `CaesarCipher` and `SubstitutionCipher` inherit from the `Cipher` class, overriding the encrypt and decrypt methods
- `VigenereCipher` will inherit from `CaesarCipher`, overriding the encrypt and decrypt methods
- Let's look at the code.

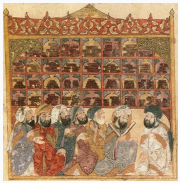
Caesar Cipher

- Key is some number *k* from 1–25
- Encrypt: Move each letter “ahead” by *k* letters, wrapping around if you go past Z
- Decrypt: Move each letter “back” by *k* letters

Vigenère Cipher

- Key is an arbitrary string, let's say the string is "ABD"
- Encrypt: move each letter "ahead" by the index of the corresponding letter in the key. (A has index 0, B has index 1, etc.)
    - In our example, the 1st letter is moved ahead by 0 for A, the next by 1 for B, the next by 3 for D, then we loop around so the fourth letter is moved ahead by 0 for A again
- Decrypt: move each letter "back" by the index of the corresponding letter in the key.

Substitution Cipher



- Key consists of a list of 26 letters denoting how to map each letter to a new letter
- Encrypt: replace each letter with the new letter
- Decrypt: replace each letter with its original

Abstract Classes

Motivation

- We did something a bit silly in this implementation: we implemented the `encrypt` and `decrypt` methods in `Cipher` even though they didn't do anything

- We *had* to do this if we wanted to call `encrypt()` and `decrypt()` on variables of type `Cipher` in the main method

- Idea: why waste time writing this code?
  - Can we write a class where we just "leave out" a method, and say it must be overridden by any subclass?
  - This is an abstract class

Abstract Class

- An abstract class exists *only* to be extended by other classes
- It may be incomplete! It may have methods that *must* be overridden by its subclasses
    - For this reason, cannot *instantiate* an object of abstract class type. Some of its methods are missing!
- An abstract class is declared using the `abstract` keyword. Any “missing” method is declared (with the `abstract` keyword), but the body is left out: a semicolon is used instead of the braces.
- Must use the `abstract` keyword to declare each such method
- Let’s turn `Cipher` into an abstract class. We’ll replace `encrypt()` and `decrypt()` with stubs.