LEC 02

CSE 123

Abstract Classes

BEFORE WE START

Talk to your neighbors:
Coffee or tea? Or something else?

Instructor: James Wilcox

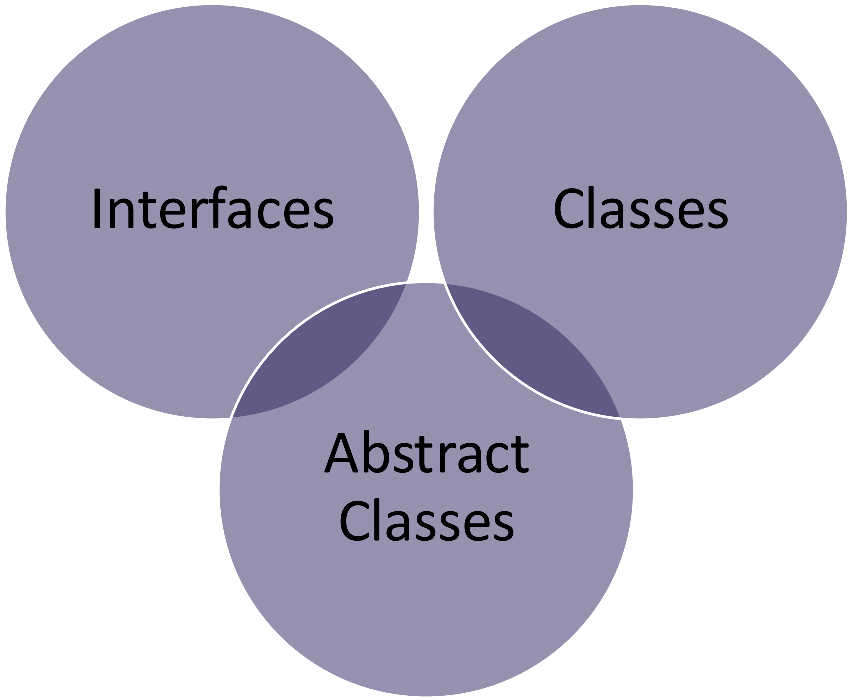
Questions during Class?

Raise hand or send here

sli.do #cse123

Abstract Classes

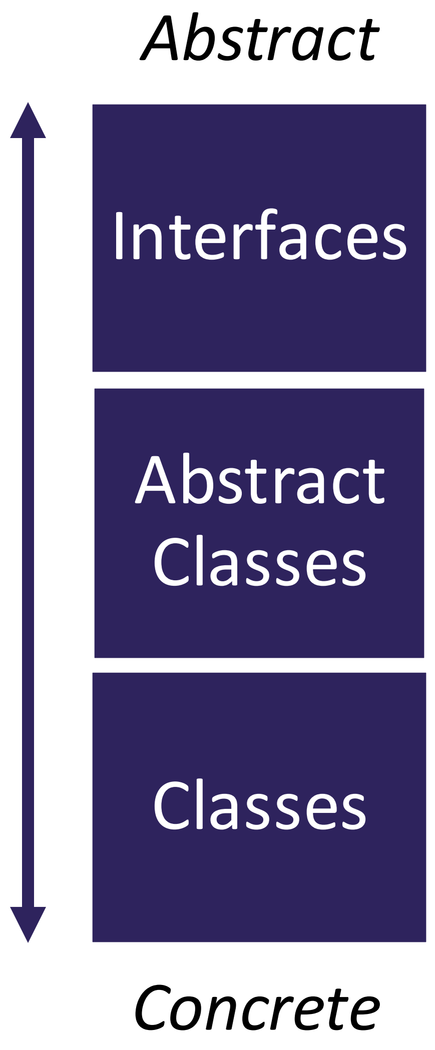
- Mixture of `Interfaces` and `Classes`
  - `Interface` similarities:
    - Can contain (`abstract`) method declarations
    - Can't be instantiated
  - `Class` similarities:
    - Can contain method implementations
    - Can have fields



- Is there identical / nearly similar behavior between classes that shouldn't inherit from one another?

Advanced OOP Summary

- Allow us to define differing levels of abstraction
  - Interfaces = high-level specification
    - What behavior should this type of class have
  - Abstract classes = shared behavior + high-level specification
  - Classes = individual behavior implementation
- Inheritance allows us to share code via “is-a” relationships
  - Reduce redundancy / repeated code & enable polymorphism
    - Still might not be the “best” decision!
  - Interfaces extend other interfaces
  - (abstract) classes extend other (abstract) classes



- You’re now capable of designing some pretty complex systems!

Design in the "real world"

- In this course, we'll always give you expected behavior of the classes you write
  - Often not the case when programming for real
  - Clients don't really know what they want (but programmers don't either)

- My advice:
  - Clarify assumptions before making them (do I really want this functionality?)
  - **There's no one right answer**
    - Weigh the options, make a decision, and provide explanation
    - Iterative development: make mistakes and learn from them
    - Be receptive to feedback and be willing to change your mind

Interface versus Implementation

- Interface: what something *should* do
- Implementation: *how* something is done
- These are different!
- Big theme of CSE 123:

  choose between different implementations of same interface