CSCI 136:
Data Structures
and
Advanced Programming

Lecture 5

Abstraction

Instructor: Dan Barowy

Williams

Topics

- Practice Quiz
- Abstraction
- WordSeq

Practice Quiz

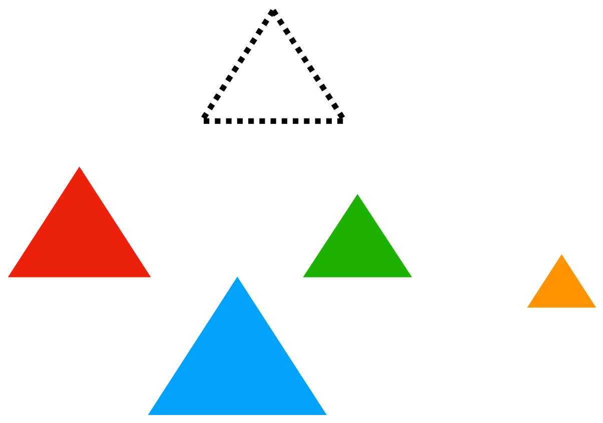
Your to-dos

1. Lab 1, **due Tuesday 2/15 by 10pm**.
2. Read **before Wed**: Bailey, Ch 2.
   Suggestion: read *actively*.

Classes

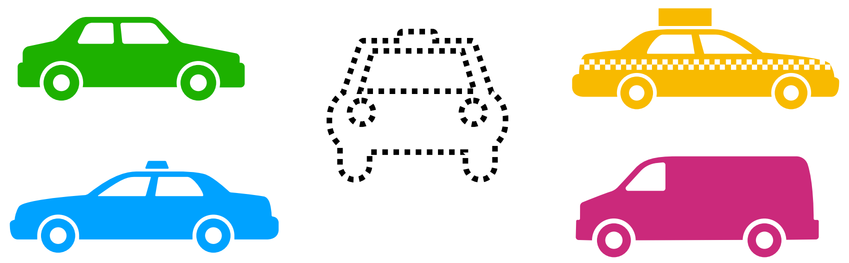
Classes are prototypes.

Objects are copies ("instances").



"Car" is a prototype.

There are many instances of cars.

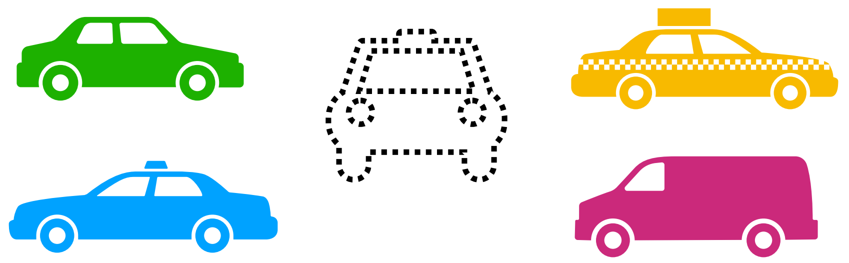


All cars have the same interface.

(wheels, doors, steering wheel, etc.)

"Car" is a prototype.

There are many instances of cars.



But most cars vary in the details

(wheels, doors, steering wheel, etc.)

Methods are functions that are tied to either:

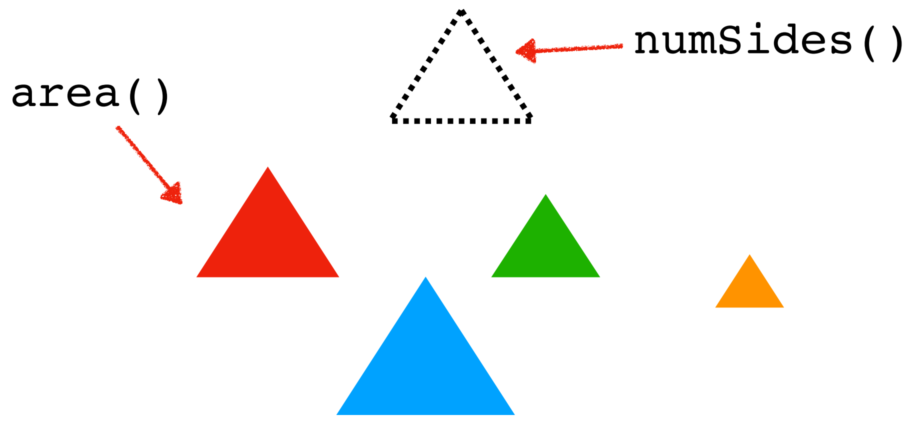
1. a class, or
2. an instance of a class (an object).

static method

instance method

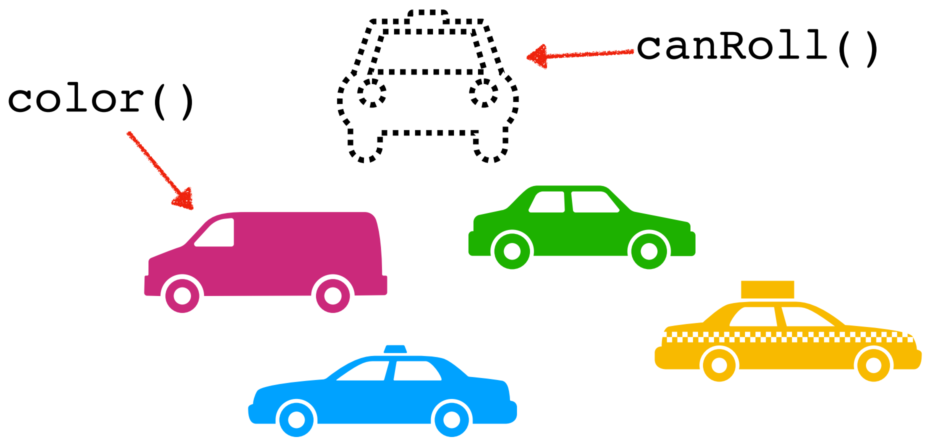
static methods are “attached” to class.

instance methods are “attached” to object.



static methods are “attached” to class.

instance methods are “attached” to object.



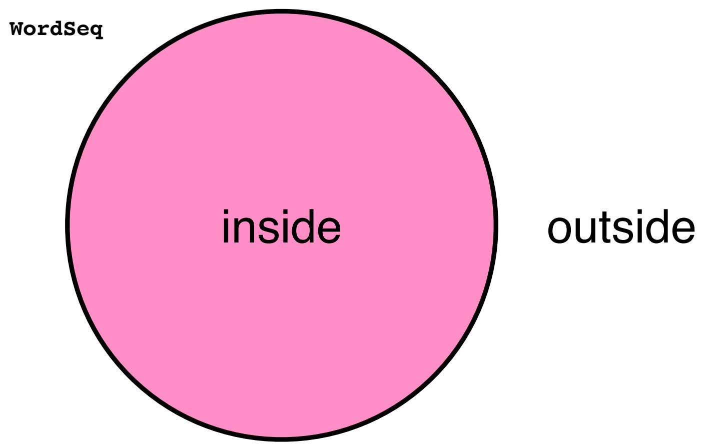
Q: How might we represent a sequence of words using a class?

How I organize a class

```
  class Foo {

1   /* INSTANCE VARIABLES */
    int bar;     // number of foos
    String baz; // foo name


2   /* CONSTRUCTOR */
    public Foo() { … }


3   /* INSTANCE METHODS */
    public int getBar() { … }
    public void setBar(int b) { … }


4   /* STATIC METHODS */
    public static void main(…) {…}

  }
```

Abstraction

Think of a class as having two sides.



Design so “user” never needs to “look inside”.

Think of a class as having two sides.

The outside: A class should represent **one concept**, and the class’s methods should support working with that one concept.

E.g., `WordSeq`**:** Represents an arbitrarily long sequence of words.

You can:

- `append` to it
- `remove` from it
- ask it for its `size`…
- convert it `to String`
- etc.

Think of a class as having two sides.

The inside: A class should contain whatever is necessary to achieve that one idea and nothing else.

E.g., `WordSeq`**:** Represents an arbitrarily long sequence of words.

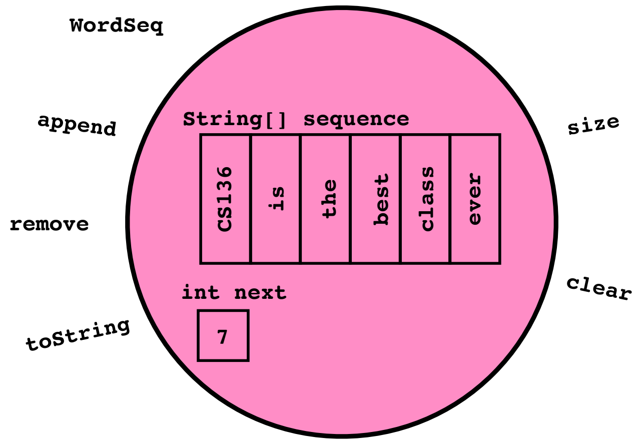
Stores:

- `String[]` of words
- Position of `next` word.

Ensures:

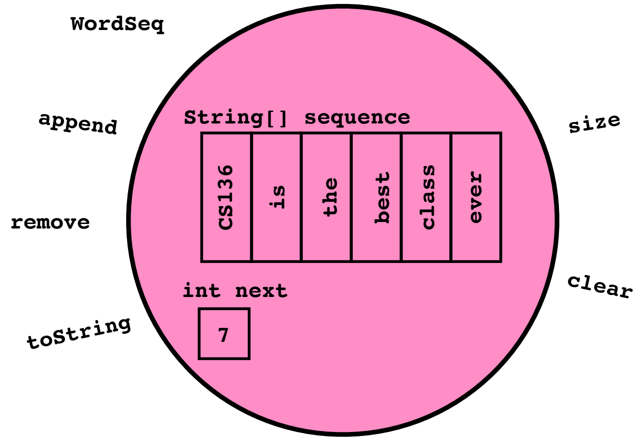
- `String[]` is always big enough (via `expand`)

Think of a class as having two sides.

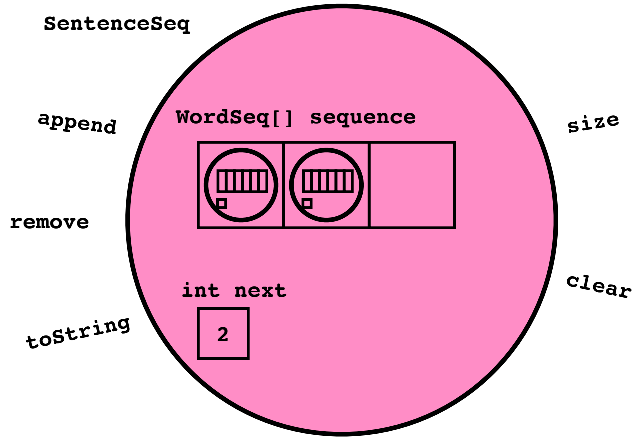


Design so user never needs to "look inside".

Hiding data inside a class is called: encapsulation



Classes can encapsulate other classes!



This is how we construct complex software.

Let’s build a `WordSeq` class

See website for posted code.

One way to get familiar with Java:

retype the code!

Recap & Next Class

Today:

- Abstraction
- WordSeq

Next class:

- Vectors and generics