CSCI 136:
Data Structures
and
Advanced Programming

Lecture 5

Abstraction

Instructor: Dan Barowy

Williams

Announcements

- No class on Friday (Winter Carvinal)
- No TA hours on Friday either!
- But there are prof. office hours (professors are no fun 💩).

- "For the pop quizzes, do we only need to know the assigned quiz prompt for that day? Or could it be any quiz prompt from the past?"

Outline

Study tip

Purpose of a class

Abstraction

Encapsulation

Generics

Life skill #5

Did you run into obstacles on Lab 1?

Life skill #5

Did these obstacles feel like somebody else's fault?

Life skill #5: reflection

Suppose you had a time machine and could time-travel back to last Monday.

What would you tell yourself to do differently?

Take a moment and write (privately).

Life skill #5: reflection

Unforeseen obstacles are common.

Life skill #5: reflection

Think about how your advice will help with Lab 2.

Java book and cheat sheet

`https://introcs.cs.princeton.edu/java/home`

`https://introcs.cs.princeton.edu/java/11cheatsheet/`

Note that Lab 2 also requires a design doc.

Please print two copies.

`WordSeq` class

Problem: **`WordSeq`** can run out of space.

Let's fix this.

(code)

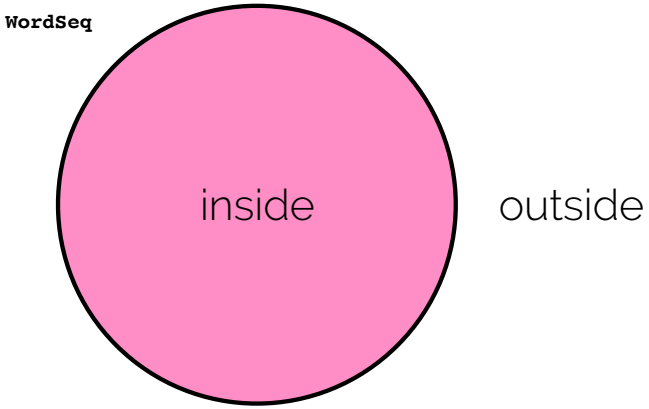
The purpose of a class:

To "abstract away" problems.

Abstraction

Abstraction is the process of **removing irrelevant information** so that a program is easier to understand.

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 idea**, and the class's methods should support working with that one idea.

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.

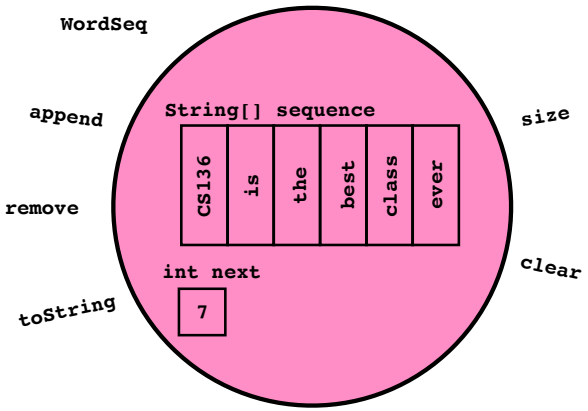
Stores:

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

Ensures:

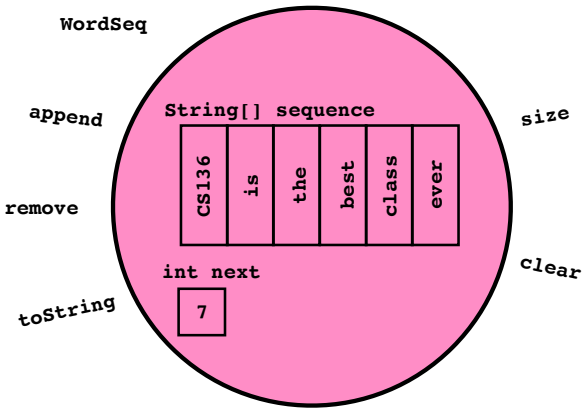
- `String[]` is always big enough (via `expand`)

Think of a class as having two sides.

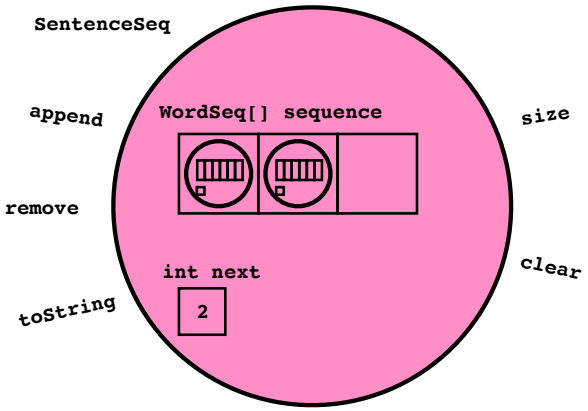


Design so user never needs to "look inside".

Hiding data inside a class is called: encapsulation



Classes can encapsulate other classes!



This is how we design complex software.

Q: Our WordSeq only works for String; could we make it work for any type?

Imagine that we could write one class that could handle any sequence.

The dream is a reality!

```
class Program {
      public static void main(String[] args) {
            Essay e = new Essay();
            // the rest of the code
      }
}

class Essay {
      Vector<Paragraph> paragraphs;
      public Essay() {
            paragraphs = new Vector<Paragraph>();
      }
      /* methods */
}

class Paragraph {
      Vector<Sentence> sentences;
      public Paragraph() {
            sentences = new Vector<Sentence>();
      }
      /* methods */
}
```

Generic types

A **generic type** is a placeholder (a **type variable**) for a type **to be specified later**. Generic types permit the creation of common algorithms and data structures (e.g., a generic sequence), thus **reducing code duplication**. Generics allow for **data type abstraction**.

`Vector` is a generic class;
it works with any type.★

Generic class

```
Vector<T> v = new Vector<T>();
```

Type parameter (fill in with the type you want)

(`Vector` documentation)

★ The type parameter you use must be a class type.

~~`Vector<int> v = new Vector<int>();`~~

Primitives (like `int`) do not work.
Use "boxed" types instead.

```
Vector<Integer> v = new
  Vector<Integer>();
```

Q: What are the Java primitive types?

You can make your own generic classes.

We will revisit later in the semester.

Recap & Next Class

Today we learned:

- Purpose of a class
- Abstraction
- Encapsulation
- Generics

Next class:

- Dictionary
- Random Sampling
- “Boxes and arrows” model