Admin

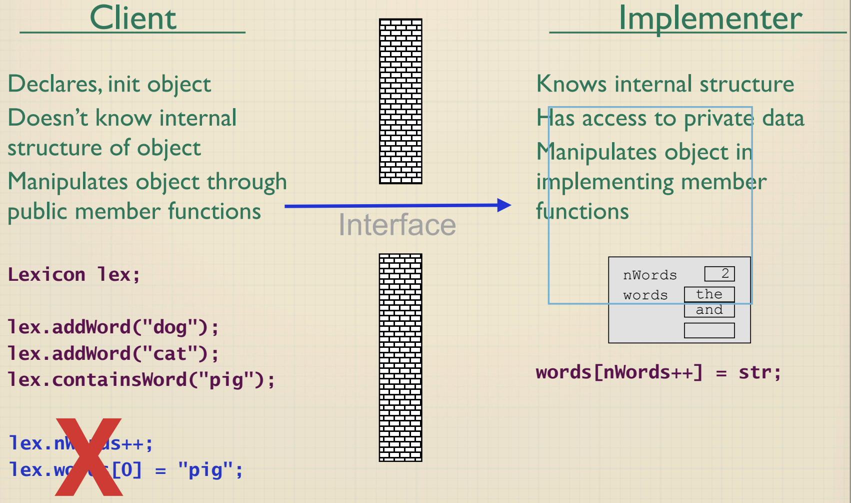
- Midterm still being graded
  - Hopefully back on Monday
- Today's topics
  - Let's implement Vector!
- Reading
  - Ch 10 implementing class templates
- Terman café today after class

Lecture #18

ADTs (abstract data types)

- Client uses class as abstraction
  - Invokes public operations only
  - Internal implementation not relevant!
- Client can't and shouldn't muck with internals
  - Class data should private
- Imagine a "wall" between client and implementor
  - Wall prevents either from getting involved in other's business
  - Interface is the "chink" in the wall
    - Conduit allows controlled access between the two
- Consider Lexicon
  - Abstraction is a word list, operations to verify word/prefix
  - How does it store list? using array? vector? set? does it matter to client?

Wall of abstraction



Client

Declares, init object

Doesn't know internal structure of object

Manipulates object through public member functions

```
Lexicon lex;

lex.addWord("dog");
lex.addWord("cat");
lex.containsWord("pig");

lex.nWords++;
lex.words[0] = "pig";
```

Implementer

Knows internal structure

Has access to private data

Manipulates object in implementing member functions

```
words[nWords++] = str;
```

Why ADTs?

- Abstraction
  - Client insulated from details, works at higher-level
- Encapsulation
  - Internals private to ADT, not accessible by client
- Independence
  - Separate tasks for each side (once agreed on interface)
- Flexibility
  - ADT implementation can be changed without affecting client