CSC207.01 2014S, Class 14: Interfaces and Polymorphism

Overview

- Preliminaries.
  - Admin.
  - Upcoming Work.
  - Questions.
- Polymorphism in General
- Interfaces in Java.
- Polymorphism in Java.
- Lab.

Preliminaries

Admin

- Food! (Within some interpretations of the word.)
- Thanks for making it to class in the lovely weather. (At least to those of you who made it to class.)
- Mentor sessions are moving to Tuesday nights at 8, starting next week.
- I will not be holding a review session on Thursday. Sorry.
- Reminder: If you are planning to do work in CS and you haven't started applying for internships and such, you should be doing so soon!
- This lab and the next lab will probably take a class-and-a-half each, particularly as we insert explanations and such.
- I plan to distribute the exam in class tomorrow.
- Think about questions to ask WH on Monday.
- Extra credit:
  - CS Extra Thursday at 4:30: Stone on Red/Black Trees
  - CS Table Friday at noon: Law, Order, and Computers
  - Dance ensemble with Beloit, April 6 or 7. Somewhere in Chicago.
  - More?

Upcoming Work

- Reading for Wednesday: Inheritance.
- Today's writeup: Exercise 7 (one of the three options)
  - CSC 207 Writeup 9: Polymorphism
  - Due Friday.

- Keep working on homework 4!

Questions on the Homework

How do I identify the fractions if I also have division?

EW says "There are spaces between the values and operations, and fractions won’t have the spaces."

3 / 4/5 is "three divided by 4/5"

3/4 / 5 is "3/4 divided by 5"

3/4/5 is EXCEPTIONAL

What do you mean by "swap out" the interface?

You have a UI in the main. It calls sensible helper procedures to do the real work.

How do we use the registers?

I’d do something like the following

```
Fraction registers[] = new Fraction[8];

...

registers[regnum] = ...;
```

So, the Calculator class needs methods to store (and maybe get) values from registers?

Yes.

Should the constructor reduce fractions to simplest form an ensure that they have a positive denominator?

Yes.

So someone can write new Fraction(33,-99) and get -1/3?

Yes.

Polymorphism

Generally: The idea that you should not have to write nearly-identical code for nearly-identical inputs.

Why not copy-paste-change?

- Sam says it’s a bad idea. You should write general code.
- The agile manifesto says so. (or EW says that JS says the agile manifesto says so.)
- Wastes programmer time to write the similar procedures.
  - Wastes programmer time to have to read all the similar junk.

- Wastes space - lots of duplicated stuff.
- Expands the amount of effort to make a change.
- Some people (see above) claim that when you write general code, you think more carefully about it.
- Duplicated code increases the chance to make a mistake.

In Scheme

(define square (lambda (x) (* x x)))

We’d like to do something similar in Java, except ...

- Kinda high level. Maybe abstraction is expensive.

- Dangerous! We don’t know until runtime whether or not it’s save to use square on a value. Can we square strings? Can we square images?

  - Java likes to know at compile time that an operation is likely to be safe.

- Ideally, the generalized procedure says "I need this characteristic of the objects", objects say "I have this characteristic", and Java can check both issues.

Interfaces

- A mechanism to describe characteristics.

- An interface is a promise to implement procedures.

  public Interface Multipliable { public Object multiply(Object other) throws IncompatibleTypeException; }

  public Object square(Multipliable m) { return m.multiply(m); }

- Promise to meet characteristics with an implemetns clause

  public class ComplexNumber implements Multpliable { }

Polymorphism in Java

- You can treat interfaces as types
  - Parameters to methods
  - Variables
  - Etc.
- Often something we take advantage of in our own code, rather than relying on particular Java interfaces.
  - Useful standard interfaces include Iterable, Comparable, Cloneable

Text Boxes

Three methods

- rows
- cols
- getrow

Useful for describing interesting textual things

Lab