CSC207.01 2013F, Class 13: Inheritance

Overview

- Prelmiinaries.
  - Admin.
  - Questions on HW4.
- More on the text block example.
- Inheritance basics.
- Lab.

Admin

- Warning! It's Friday the 13th (class)
- Reading for Monday: Documentation with Javadoc
  - Ready!
- EC Opportunities
  - CS Extras Thursday @ 4:30: Jennelle Nystrom on Microsoft
  - CS Table Friday (pair programming)
  - CS Table Friday next (The Story of Mel, A Real Programmer)
  - Other?
- If you choose to drink alcohol this weekend, please choose to drink responsibly.
- I'm trying to do very little lecture today. We'll see if I succeed.
  - Minor fail
- Updates
  - Lab continued on Monday
  - Homework due on Tuesday night

HW4

How should we evaluate the following?

```
r1 = 2/3
1 + r1
```

You should give something like the following

```
2/3, 5/3
```

What would you put in the Fraction class? (We'll do this as a group.)

- Constructors
  - `Fraction(int,int)`
  - `Fraction(int)` - N/1

    - `Fraction(BigInteger,BigInteger)`
    - NOT `Fraction(Fraction)` - That's just cloning
    - `Fraction(BigInteger)`
    - `Fraction(String)` - `new Fraction("11/15")`
    - `Fraction(long,long)`
    - `Fraction(double)`
- Public methods
    - `Fraction add(Fraction other)`
    - `Fraction subtract(Fraction other)`
    - `Fraction multiply(Fraction other)`
    - `Fraction divide(Fraction other)`
    - `Fraction pow(int expt)`
    - `double doubleValue()`
    - `BigDouble bigDoubleValue()`
    - `Fraction reciprocal()`
    - `Fraction negate()`
    - `BigInteger numerator()`
    - `BigInteger denominator()`
    - `Fraction fractionalPart()`
    - `BigInteger wholePart()`
- Standard methods
    - `Fraction clone()`
    - `String toString()`
    - `int hashCode()`
    - `int compareTo(Fraction other)`
    - `boolean equals(Object other)`
- Private methods
    - `void simplify()`
    - `void cleanup()`
- Methods that we don't really want to implement and Sam won't expect us to implement
    - `Fraction pow(Fraction expt)`
    - `Fraction sqrt()`
    - `Fraction log()`
    - `BigInteger round()`
    - `BigInteger ceiling()`

Hash code implementation

```
* ‘int hashCode() { return 1; }‘
* ‘int hashCode() { return numerator.hashCode() * denominator.hashCode(); }‘
```

Why won't my test code work

```
    public void test() throws Exception {
        ...
    }
```

More on the text block example

- A line is a text block
- The vertical compositon of two text blocks is a text block
- A textblock that is boxed is a textblock

```
+-------+
| Hello |
+-------+
```

- What are the important ideas in this example?
  - Recursive definitions of data types
  - Supported by polymorphism
  - Extensible!

Inheritance basics

- General concept: We can define new classes in terms of old
- Why not have the language do what we would do by hand Foo extends Bar
  - Implicitly copy all of the fields and methods of Bar into Foo
  - Foo can add new methods and fields
  - Foo can replace methods (but not fields)
- Supports polymorphism
- Lets Aristotilian philosophy add to Platonic philosophy

Lab

- Start it in class.
- We'll continue it on Monday. People are having more difficulty than I expected.
- Sorry that I failed in being succinct. I'll try again on Monday.

Sample code

```
public class Counter {
   int count;
   int start;

   public Counter(int i) {
       this.count = i;
       this.start = i;
   } // Counter(int)

} // class Counter
```