CSSE 220 Day 15

Function Objects and the Comparator Interface
Merge Sort

Checkout *Day_15_201210* project from SVN

Questions

Today's Plan

- Merge sort recap
- A step back: Fraction class
- Introduction to function objects, **Comparator**
  - As much as time allows

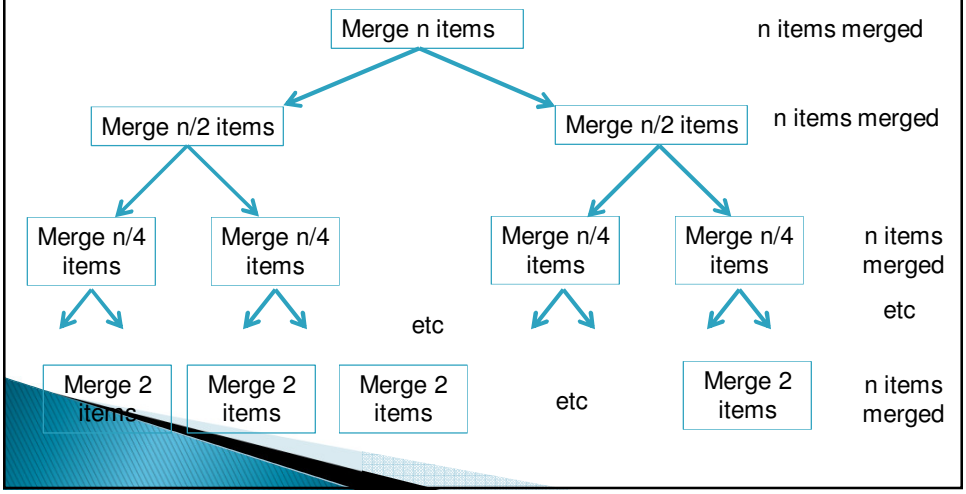
Merge Sort Recap

- Basic recursive idea:
  - If list is length 0 or 1, then it's already sorted
  - Otherwise:
    - Divide list into two halves
    - Recursively sort the two halves
    - **Merge** the sorted halves back together

Analyzing Merge Sort

If list is length 0 or 1,
then it's already sorted

- Otherwise:
  - Divide list into two halves
  - Recursively sort the two halves
  - **Merge** the sorted halves back together



A few Retro moments

- Well, maybe a lot of moments.
- Design, implement, test a Fraction class

Java.util.comparable

- How does it work?
- Sort an array of Integer.

Function Objects

- Another way of creating reusable code

A Sort of a Different Order

- Java libraries provide efficient sorting algorithms
  - `Arrays.sort(…)` and `Collections.sort(…)`
- But suppose we want to sort by something other than the "natural order" given by `compareTo()`
- *Function objects* to the rescue!

Function Objects

- Objects defined to just "wrap up" functions so we can pass them to other (library) code
- For sorting we can create a function object that implements `Comparator`
- Let's try it!

Function Objects (a.k.a. Functors)

- Why do methods have arguments in the first place?
- We'd like to be able to pass a method as an argument to another method
- This is not a new or unusual idea.
  - You pass other functions as arguments to Maple's *plot* and *solve* functions (on a later slide).
  - C and C++ provide *qsort*, whose first argument is a comparison function.
  - Scheme and Python also have *sort* functions that can take a comparison function as an argument.

In Scheme

- Scheme has a sort function that takes a function as an argument:

```
Chez Scheme Version 7.4
Copyright (c) 1985-2007 Cadence Research Systems
> (sort > '(7 3 9 -2 5 -6 0 4 1 -8))
(9 7 5 4 3 1 0 -2 -6 -8)
> (sort (lambda (x y) (< (abs x) (abs y)))
        '(7 3 9 -2 5 -6 0 4 1 -8))
(0 1 -2 3 4 5 -6 7 -8 9)
```

Similar example in Python

```
>>> list = [4, -2, 6, -1, 3, 5, -7]
>>> list.sort()
>>> list
[-7, -2, -1, 3, 4, 5, 6]
>>> def comp (a, b):
        return abs(a) - abs (b)

>>> list.sort(comp)
>>> list
[-1, -2, 3, 4, 5, 6, -7]
```

The comp function is passed as an argument to the sort method

Similar example in Maple

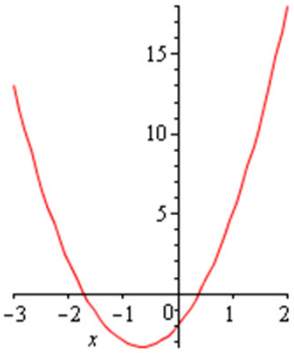
```
> sort([3, 7, -3, 4, -6, 1, 8], `<`);
                    [-6, -3, 1, 3, 4, 7, 8]
> sort([3, 7, -3, 4, -6, 1, 8], `>`);
                    [8, 7, 4, 3, 1, -3, -6]
> absless := (x, y) → abs(x) < abs(y);
                    absless := (x, y) → |x| < |y|
> sort([3, 7, -3, 4, -6, 1, 8], `absless`)
                    [1, -3, 3, 4, -6, 7, 8]
```

More Maple

```
> f := x->3*x^2 + 4*x - 2;
```

$$f := x \to 3x^2 + 4x - 2$$

```
> plot(f(x), x=-3..2);
```



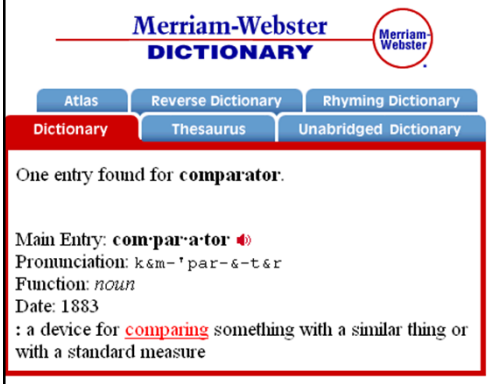

```
> solve(f(x), x);
```

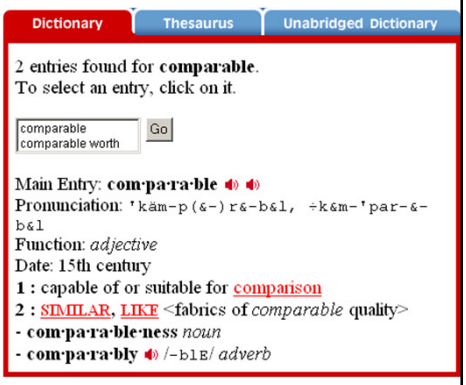
$$-\frac{2}{3} + \frac{\sqrt{10}}{3}, -\frac{2}{3} - \frac{\sqrt{10}}{3}$$

Java Function Objects

- What's it all about?
  - Java doesn't (yet) allow passing functions as arguments.
  - So, we create objects whose sole purpose is to pass a function into a method
  - Called **function objects**
    - a.k.a. functors, functionoids, more fun than a barrel of monkeys
- Most famous Function Object Class: `Comparator`

You say "tomato", I say "toe-mah-toe"





Java: "imposed" ordering

"natural" ordering

Sorting Arrays and Collections

- `java.util.Arrays` and `java.util.Collections` are your friends!
- On Written Assignment 2
  - The **CountMatches** implementation problem asks you to write code that creates and uses function objects.