CSSE 220 Day 15

Function Objects and the Comparator Interface
Merge Sort

Checkout *FunctionObjects* project from SVN

Questions

Today's Plan

- Merge sort recap
- Introduction to function objects, **Comparator**

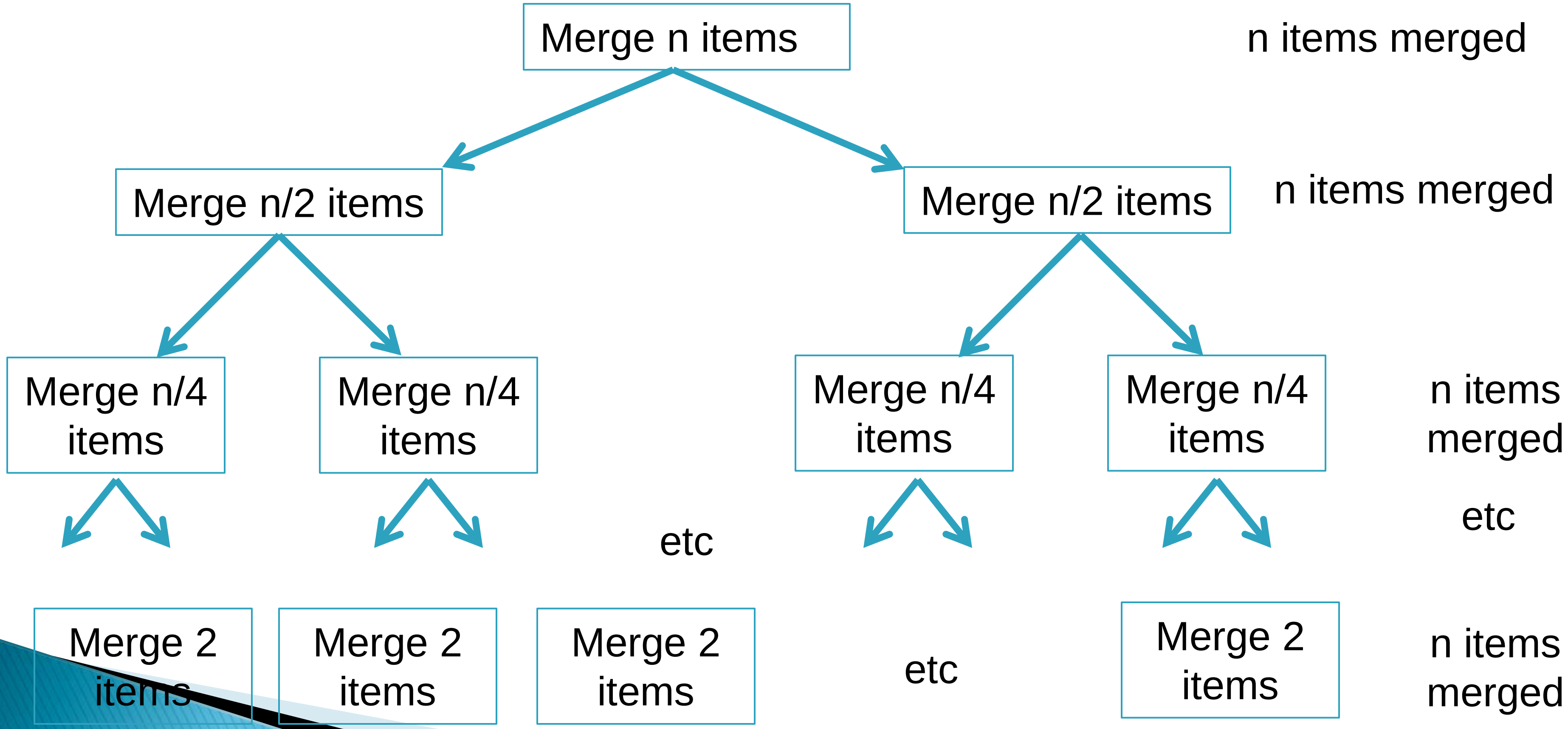
Merge Sort Recap

- 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



Merge n items

n items merged

Merge n/2 items

Merge n/2 items

n items merged

Merge n/4 items

Merge n/4 items

Merge n/4 items

Merge n/4 items

n items merged

etc

etc

Merge 2 items

Merge 2 items

Merge 2 items

etc

Merge 2 items

n items merged

java.util.Comparable

- How does it work?
- How would we 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 (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)
```

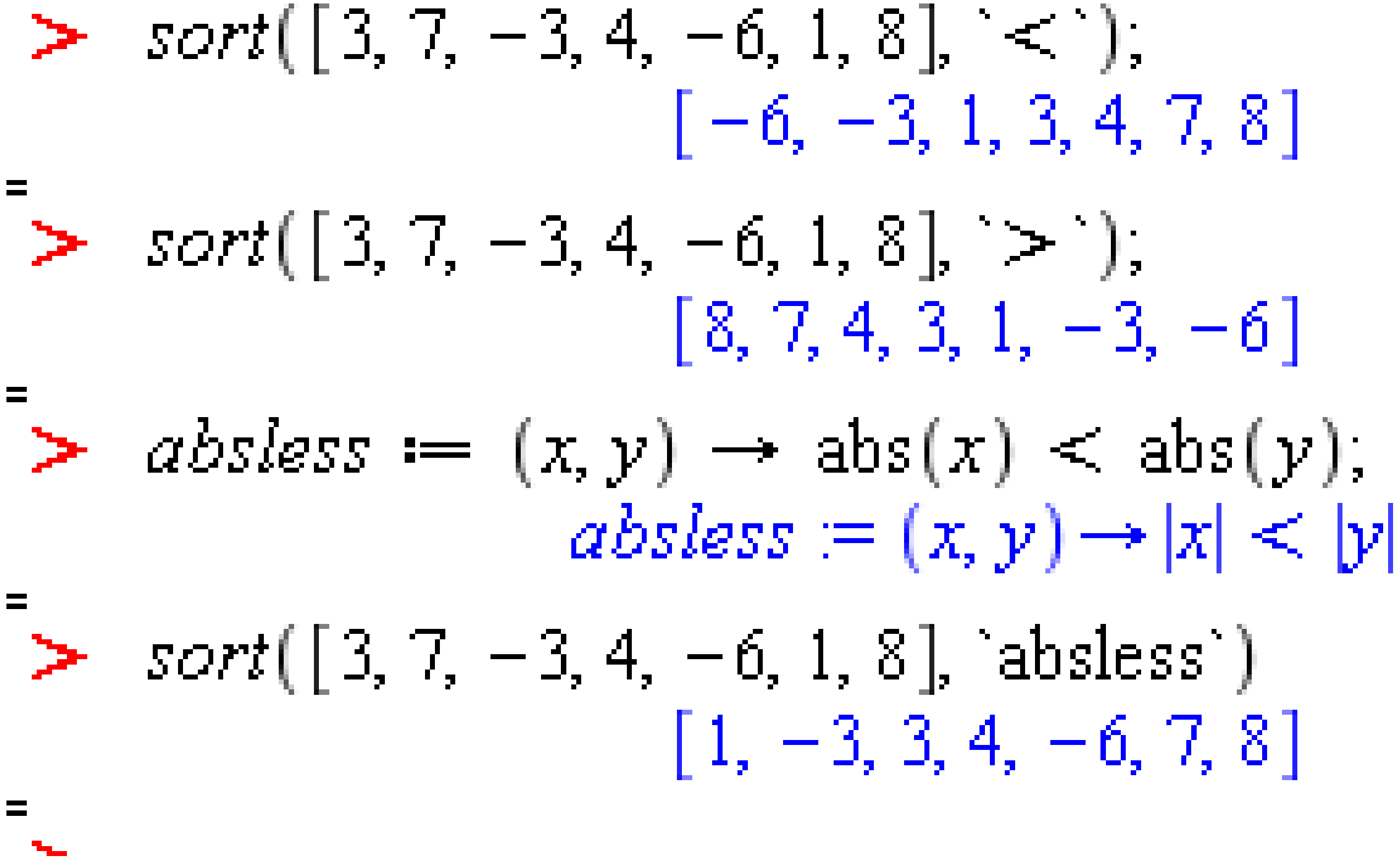
Similar example in Python

```
>>> 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]
=
```

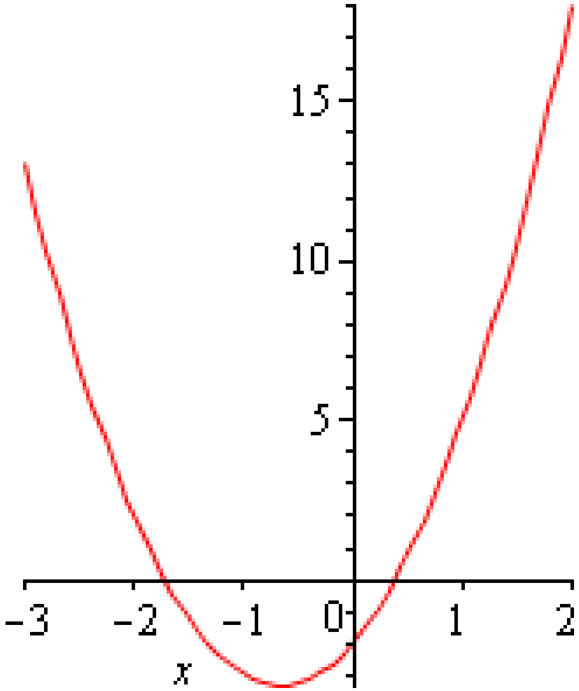
More Maple

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

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

=

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



=

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

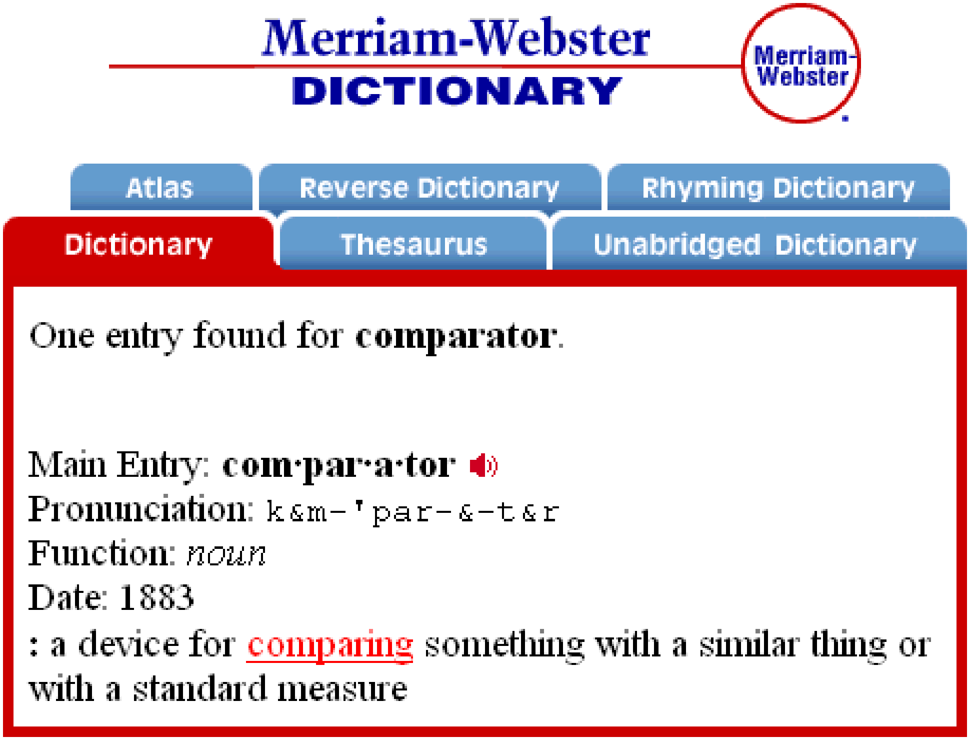
$$-\frac{2}{3}+\frac{\sqrt{10}}{3},\ -\frac{2}{3}-\frac{\sqrt{10}}{3}$$

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

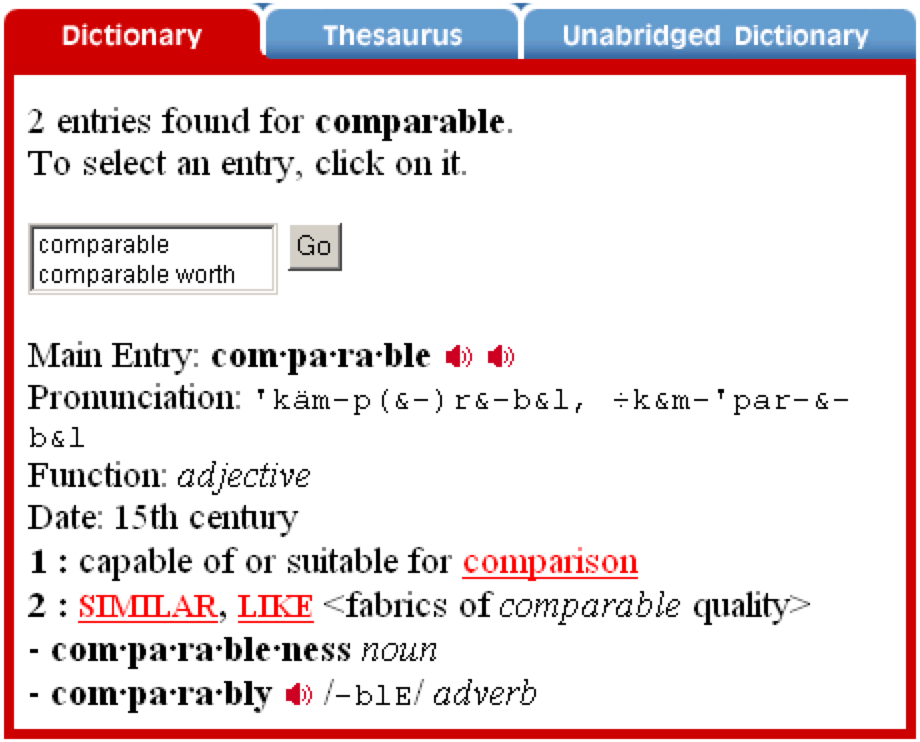
- Most famous Function Object Class: ***`Comparator`***

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



One entry found for **comparator**.

Main Entry: **com·par·a·tor** 🔊
Pronunciation: k&m-'par-&-t&r
Function: *noun*
Date: 1883
: a device for comparing something with a similar thing or with a standard measure



2 entries found for **comparable**.
To select an entry, click on it.

comparable
comparable worth

Main Entry: **com·pa·ra·ble** 🔊 🔊
Pronunciation: 'käm-p(&-)r&-b&l, ÷k&m-'par-&-b&l
Function: *adjective*
Date: 15th century
1 : capable of or suitable for comparison
2 : SIMILAR, LIKE <fabrics of *comparable* quality>
- com·pa·ra·ble·ness *noun*
- com·pa·ra·bly 🔊 /-blE/ *adverb*

Java: "imposed" ordering

"natural" ordering

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!*