CSSE 220 Day 15

Function Objects
Comparator
Work on Minesweeper

CSSE 220 Day 15

- Minesweeper Progress Report due today at the end of class.
  - Name it Day 15 progress Report.xlsx . Commit it to your repository.
  - Note that the middle of today is the half-way point for Minesweeper implementation. Are you almost halfway done?
  - Student assistants are available in the lab this afternoon, this evening, tomorrow afternoon, Sunday evening.
- Key Concepts Quiz has been set so that you can see your answers and the correct answers.

Future Exams

- **Next Exam** is Thursday, January 31, as originally announced in the syllabus
  - I have had some requests for an **evening exam** so there is less time-pressure
    - I will soon post a survey on ANGEL to get your opinion about that
  - **Another possibility**: Anyone who wishes can take the non-programming part of the exam 7:15-7:55 AM, so they have two full class periods for the programming part
- **Final Exam** is Monday, Feb 18 at 6 PM

Answers to your questions

- Minesweeper
- Material you have read
- Anything else

Today's agenda

- More on Algorithm analysis – Big Oh
- Function objects (a.k.a Functors)
- Work on Minesweeper

Recap: O, Ω, Θ

- f(N) is O(g(N) if there is a constant c such that for sufficiently large N, $f(N) \leq cg(N)$
  - Informally, the growth rate of f is bounded above by the growth rate of g
- f(N) is Ω(g(N) if there is a constant c such that for sufficiently large N, $f(N) \geq cg(N)$
  - Informally, the growth rate of f is bounded below by the growth rate of g
- f(N) is Θ(g(N) if f(N) is O(g(n)) and f(N) is Ω(g(N))
  - Informally, the growth rate of f is the same as the growth rate of g

Recap: Limits and asymptotics

$\lim_{n\to\infty} \frac{f(n)}{g(n)}$	Conclusions		
0	f(n) is O(g(n))	g(n) is not O(f(n))	f(n) is not Θ(g(n))
non-zero	f(n) is O(g(n))	g(n) is O(f(n))	f(n) is Θ(g(n))
∞	f(n) is not O(g(n))	g(n) is O(f(n))	f(n) is not Θ(g(n))
undefined	We cannot conclude anything from the limit of the ratios.		

Apply this limit property to the following pairs of functions

1. N and N^2
2. $N^2 + 3N + 2$ and N^2
3. **$N + \sin(N)$ and N**
4. $\log N$ and N
5. $N \log N$ and N^2
6. **N^a and N^N**
7. a^N and b^N $(a < b)$
8. $\log_a N$ and $\log_b N$ $(a < b)$
9. $N!$ and N^N

Function objects – recap

- The ability to pass functions as arguments to other functions can enable us to write code that is more flexible and generic
- Example that we examined in several different languages:
  - Pass a (built-in or user-defined) comparison function as one of the arguments to a sort function
- Unfortunately, Java (unlike C++) doesn't allow functions to be passed as arguments
- But we can create objects whose whole purpose is to pass a function into a method. They are called **function objects,** a.k.a. functors.
- For a (somewhat advanced overview of function objects in different languages:
  - http://en.wikipedia.org/wiki/Function_object
- Primary built-in Java example interface: Comparator

How to pronounce Comparator, Comparable

Merriam-Webster DICTIONARY

Atlas | Reverse Dictionary | Rhyming Dictionary

Dictionary | Thesaurus | Unabridged Dictionary

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

Dictionary | Thesaurus | Unabridged Dictionary

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

comparable
comparable worth
Go

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*

Comparator Interface

Install this.

See HW 15

```
package weiss.util;  // It's in java.util also.

import java.io.Serializable;

/**
 * Comparator function object interface.
 */
public interface Comparator extends Serializable
{
    /**
     * Return the result of comparing lhs and rhs.
     * @param lhs first object.
     * @param rhs second object.
     * @return < 0 if lhs is less than rhs,
     *           0 if lhs is equal to rhs,
     *         > 0 if lhs is greater than rhs
     * @throws ClassCastException if objects
     *     cannot be compared.
     */
    int compare( Object lhs, Object rhs )
                 throws ClassCastException;
}
```

Weiss provides code for several classes that are equivalent to those in **java.util,** so we can see how parts of the **java.util** classes might be implemented.

Generics would make this code slightly more complicated; we'll most likely deal with that later.

Example: Rectangles

```
public class SimpleRectangle {
    public SimpleRectangle( int l, int w ) {
      length = l; width = w;
    }

    public int getLength( ) {
      return length;
    }

    public int getWidth( ) {
      return width;
    }

    public String toString( ) {
      return "Rectangle " + getLength( )
                + " by "  + getWidth( );
    }

    private int length;
    private int width;
}
```

The **SimpleRectangle** class does *not* implement **Comparable**, because there is not one "natural" way to order **SimpleRectangle** objects.

FindMax Uses a Comparator object

```
public class CompareTest
{
    public static Object findMax( Object [ ] a, Comparator cmp )
    {
        int maxIndex = 0;
        for( int i = 1; i < a.length; i++ )
            if( cmp.compare( a[ i ], a[ maxIndex ] ) > 0 )
                maxIndex = i;

        return a[ maxIndex ];
    }

    public static void main( String [ ] args )
    {
        Object [ ] rects = new Object[ 4 ];
        rects[ 0 ] = new SimpleRectangle( 1, 10 );
        rects[ 1 ] = new SimpleRectangle( 20, 1 );
        rects[ 2 ] = new SimpleRectangle( 4, 6 );
        rects[ 3 ] = new SimpleRectangle( 5, 5 );

        System.out.println( "MAX WIDTH: " +
                            findMax( rects, new OrderRectByWidth( ) ) );
        System.out.println( "MAX AREA: " +
                            findMax( rects, new OrderRectByArea( ) ) );
    }
}
```

vs. a[i].compareTo(a[maxIndex])

Note that java.util.Collections.max has the functionality of this findMax method.

Without something like Comparators, we would need separate `findMax` functions for finding the max using different comparison criteria

The Actual Function Objects

```
class OrderRectByArea implements Comparator
{
    public int compare( Object obj1, Object obj2 )
    {
        SimpleRectangle r1 = (SimpleRectangle) obj1;
        SimpleRectangle r2 = (SimpleRectangle) obj2;

        return( r1.getWidth()*r1.getLength() -
                r2.getWidth()*r2.getLength() );
    }
}

class OrderRectByWidth implements Comparator
{
    public int compare( Object obj1, Object obj2 )
    {
        SimpleRectangle r1 = (SimpleRectangle) obj1;
        SimpleRectangle r2 = (SimpleRectangle) obj2;

        return( r1.getWidth() - r2.getWidth() );
    }
}
```

Two Comparator classes.

Examples: Arrays and Collections

static <T> int	binarySearch(T[] a, T key, Comparator<? super T> c) Searches the specified array for the specified object using the binary search algorithm.

static <T> void	sort(T[] a, Comparator<? super T> c) Sorts the specified array of objects according to the order induced by the specified comparator.

static <T> T	max(Collection<? extends T> coll, Comparator<? super T> comp) Returns the maximum element of the given collection, according to the order induced by the specified comparator.

static <T> void	sort(List<T> list, Comparator<? super T> c) Sorts the specified list according to the order induced by the specified comparator.

In-class Assignment

- You can (and should) talk to your neighbors, the student assistants, and me, but you should submit your own work.
- Starting code is in your individual **SVN** repository.
  Project name: CountMatches_Weiss_4.29_and4.30
- It includes JUnit tests that you should get to run successfully.
- Weiss problems 4.29, 4.30 (statements are on a very small handout).
- **EqualsK** (problem 30) should implement the interface from problem 29a. I called that interface **Matchable**
- Analogy with our Rectangle example:
  - **countMatches** (corresponds to findMax)in the example) is the method that takes an array and a function object as parameters.
  - **EqualsZero** (corresponds to OrderRectsByWidth) is a specific "function object" class.
  - **Matchable** (corresponds to Comparator) is the function object interface; you get to pick the name for its method.

Work on Minesweeper

- Don't forget to commit your progress report to the repository before the end of class.