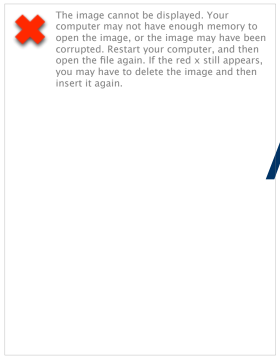

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

April 24: Interfaces, Sorting, Timing and Abstract Classes

CMPSCI 121, Spring 2012

Introduction to Problem Solving with Computers

Prof. Learned-Miller

Logistics

- Final: May 8 (Tuesday), 1:30 PM in Totman Gym.
  - Covers material from entire course, with emphasis on the second half.
  - Review: In Section next Monday.
- LAST ASSIGNMENTS.
  - Chapter 12 Reading- Thursday
  - Chapter 12 Exercises – Next Tuesday
- Last Day of class ATTENDANCE REQUIRED!!!
  - 5 points on final for showing up. Bring your id.

Review: Interfaces

- Work almost like classes
- Declare attributes and methods
- Can't create native objects of that type
  - can't use "new"
- Can create references of that type
- Can implement multiple interfaces in the same class

Writing a generic sorting algorithm

1. Need to be able to compare objects
   Use Comparable interface
   NOTE: Comparable is part of Java (you don't have to write it yourself.
2. Class to be sorted should implement Comparable
3. What class should sorting method be a part of ?
   - not Infants, not Cars, not Integers...
   - Make it own class.
4. How to write a sorting method?
   Use "bubble sort".

Studying for final

- If you can
  - Define your own Comparable interface (MyComp)
  - Define a class that implements MyComp.
  - Write a sorting method that uses MyComp to compare objects.
  - Create an array of objects that are MyComp, and use your sorting method to sort them....

- then you will know 90% of what you need to know on the final....

- DrJava

Abstract Classes

- A class you can derive from, but can’t make an instance of.
- Why would you want to do this?
  - Because you might want to make lots of different kinds of classes that all are guaranteed to have the same one capability.

```java
public abstract class JobTimer {

  public abstract void doJob();

  // keeps track of time and calls doJob
  public void runJob() {
    //call the garbage collector to make more memory available
    System.gc();
    long s1 = System.currentTimeMillis();
    doJob();
    long s2 = System.currentTimeMillis();
    long runTime = (s2 - s1);
    System.out.println("running time in milliseconds: " + runTime);
  }
}
```