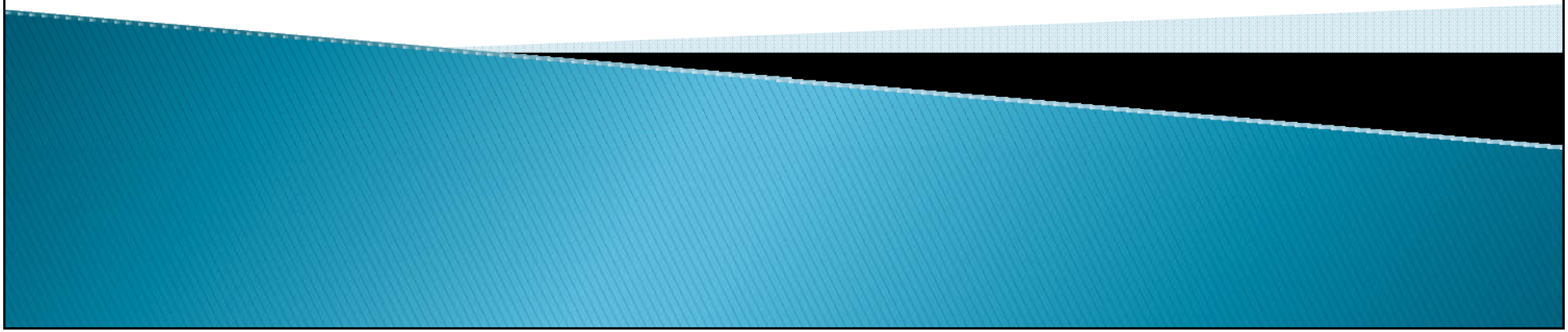
CSSE 220 Day 26

Continue the Sorting intro
Work on Spellchecker Project



CSSE 220 Day 26

- Turn in written problems now.
- Thanks to those who have posted links to dictionaries. We will standardize on one soon.
- There will be time in class to work with your team every day. Do not miss it!

- Questions?

- Today:
  - Work on Spellchecker
  - Continue the Sorting intro

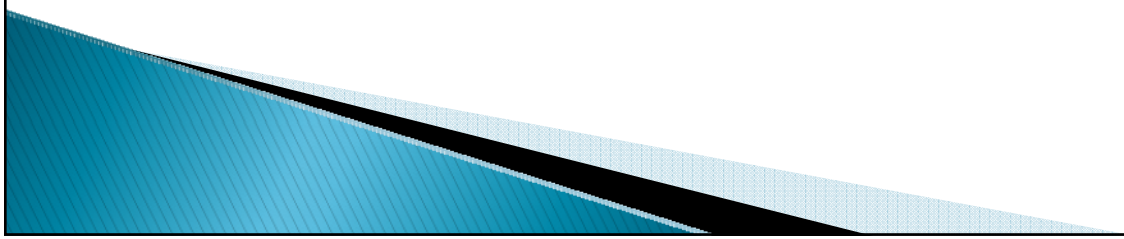
Project presentation/demonstration

- Day 30 in class
- Informal and informational
- What does your program do? How does it do it?
- Data Structures and algorithms.
- Intended audience: Your classmates
  - Already know what the project is.
  - Already know Java
  - Already know the data structures involved.
- No more than 7 minutes, including Q&A time.

- Since you need to present your final project, it is due at the beginning of your class time on Day 30. No late days may be used for this one.
  - I also don't want it to interfere with studying for exams

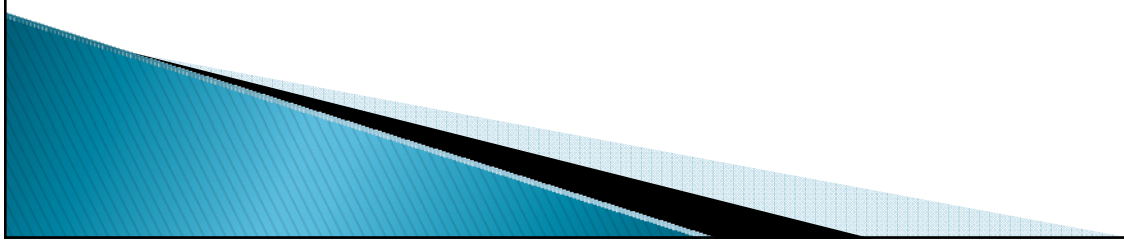
Project work

- Before you leave today:
  - UML Class Diagram
  - Iterative enhancement plan
  - Commit to your repository
- Finish UML diagram and iterative enhancement plan before midnight tonight.



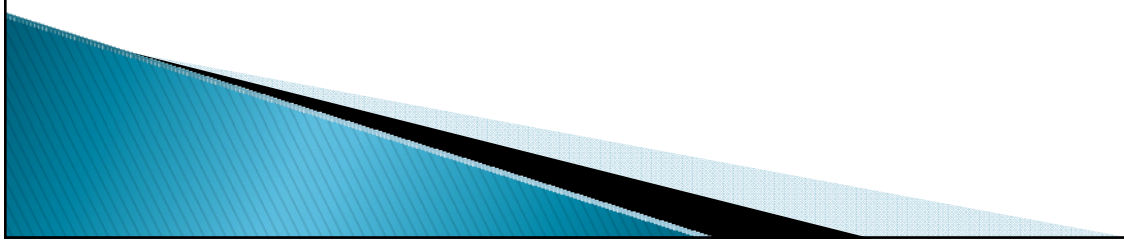
Homework

- Finish UML Class Diagram and IEP today/tonight
- Markov partner evaluation survey
- Progress on SpellChecker



Sorting Intro

- What do we mean by "sort"?
- What is the best sorting algorithm?
- The three very simple algorithms
  - Selection sort
  - Bubble Sort
    - Why is it so slow?
  - Insertion sort
- Inversions and movement
- Faster algorithms



Knowledge of Elementary Sorts

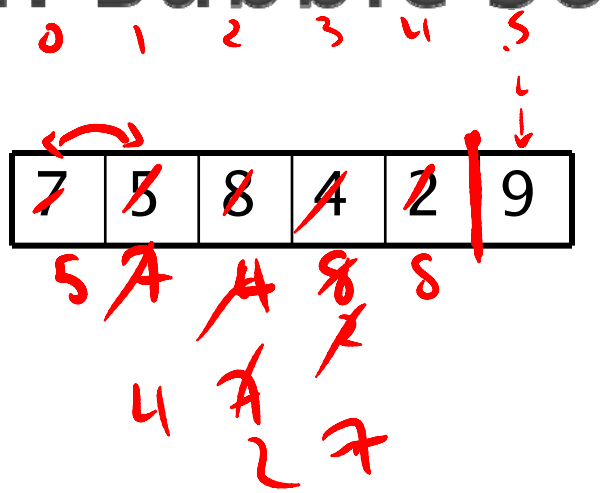
- What should you know/be able to do by the end of this course?
  - The basic idea of how each sort works
    - insertion, selection, bubble, shell, merge
  - Can write the code on paper in a few minutes
    - insertion, bubble, selection
    - perhaps with a minor error or two
    - not because you memorized it, but because you understand it
  - What are the best case and worst case orderings of N data items? For each of these:
    - Number of comparisons
    - Number of data movements

2. Bubble Sort

- *Idea:* continual swapping gets results (eventually)
  - Items "bubble" up
- Each adjacent pair of elements is swapped if they are out of order.
- http://www.cs.ubc.ca/~harrison/Java/sorting-demo.html
- http://www.geocities.com/siliconvalley/network/1854/Sort1.html

```
n = a.length
for (i = n-1; i > 0; i--){
  swapped = false
  for (j = 0; j <= i; j++){
    if (a[j] > a[j+1]){
      swap(a, j, j+1)
      swapped = true
    }
  }
  if (!swapped) return;
}
```

2. Bubble Sort




```
n = a.length
for (i = n-1; i > 0; i--){
  swapped = false
  for (j = 0; j <= i; j++){
    if (a[j] > a[j+1]){
      swap(a, j, j+1)
      swapped = true
    }
  }
  if (!swapped) return;
}
```

4	5	2	7	8	9

After outer loop repeats 3 times
(12 comparisons and 21 assignments)

2. Bubble Sort

1 2 3 4 5

- What's the runtime?
  - Worst? $\theta(n^2)$ comps., swaps
  - Best? $\theta(n)$ comps, 0, $\theta(1)$
  - Average? $\theta(n^2)$
- Extra space? $\theta(1)$
- Runtime measured in:
  - number of comparisons
  - number of swaps
- Note the one redeeming feature of bubble sort

```
n = a.length
for (i = n-1; i > 0; i--){
  swapped = false
  for (j = 0; j <= i; j++){
    if (a[j] > a[j+1]){
      swap(a, j, j+1)
      swapped = true
    }
  }
  if (!swapped) return;
}
```

Back to selection sort

- Quiz question asks for the worst-case number of comparisons and swaps
- How does this differ from bubble sort?

```
n = a.length
for (i = 0; i < n-1; i++) {
  minPos = 0
  // find the smallest
  for (j=i+1; j < n; j++){
    if (a[j]<a[minPos]){
      minPos = j
  }
  // move it to the start
  swap(a, i, minPos)
}
```

Interlude: A 5-year old's understanding of swapping

- Courtesy of my son Caleb…

3. Insertion Sort

7	5	8	4	9	2

- for (i=1; i< N; i++)
  - place a[i] in its correct position relative to a[0] …a[i-1]
  - to do this, we need to move "right" each of those items that is larger than a[i].
- Write code together now.

4	5	7	8	9	2

After outer loop repeats 3 times (7 comps, 10 assns)

3. Insertion Sort

- What is the runtime?
  - Best? $\Theta(n)$ or $\Theta(1)$ "swaps"
  - Worst? $\Theta(n^2)$ comps, $\Theta(n^2)$
  - Average? $\Theta(n^2)$
  - Extra space? $\Theta(1)$
- Runtime measured in:
  - number of comparisons
  - number of swaps

```
for(i=1; i<a.length; i++){
  temp = a[i];
  j = i;
  while (j>0 && temp<a[j-1])
       a[j] = a[j-1];
       j--;
  a[j] = temp;
}
```

Experimental Analysis

- Demo
- Use the results to confirm your answers to quiz #2-4