Lab 0.1

Friday, Feb 5th  |  11AM-2PM  |  32-123
Read lab before coming (on Stellar)
Bring your laptop

Exploration 1
Out Monday, Feb 8th
Due Monday, Feb 15th

No Java Experience?
Come talk to us
Brief Overview
  Administrivia
  Big Ideas
  Brief Review
  Debugging

What you should learn
  Basic Eclipse Use
  Simple Java Syntax
  Debugging Techniques
Always think before you code.

Iterate – solve a simple problem first, then improve.

Make code self-documenting.

Make your life easier – use the Java libraries.

Learn from your own mistakes, but don't waste time (don't be afraid to ask questions).
Hailstone Sequence
Next number in sequence is:
\( n/2 \) if \( n \) is even
\( 3n+1 \) if \( n \) is odd
Until \( n == 1 \)
### Hailstone Sequence

<table>
<thead>
<tr>
<th>Iterative Implementation</th>
<th>Recursive Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static int hailstoneLengthImp(int n) {</td>
<td>public static int hailstoneLengthRec(int n) {</td>
</tr>
<tr>
<td>int moves = 0;</td>
<td>if (n == 1) {</td>
</tr>
<tr>
<td>while (n != 1) {</td>
<td>return 0;</td>
</tr>
<tr>
<td>if (n % 2 == 0) {</td>
<td>} else if (n % 2 == 0) {</td>
</tr>
<tr>
<td>n = n / 2;</td>
<td>return 1 + hailstoneLength(n/2);</td>
</tr>
<tr>
<td>} else {</td>
<td>} else {</td>
</tr>
<tr>
<td>n = 3 * n + 1;</td>
<td>return 1 + hailstoneLength(3*n + 1);</td>
</tr>
<tr>
<td>} +moves;</td>
<td>}</td>
</tr>
<tr>
<td>} return moves;</td>
<td>}</td>
</tr>
</tbody>
</table>

- **Iterative Implementation**
  - Function **hailstoneLengthImp**
    - Takes an integer **n** as input.
    - Initializes **moves** to 0.
    - While **n** is not 1, the function checks if **n** is even or odd.
    - If **n** is even, it divides **n** by 2 and increments **moves**.
    - If **n** is odd, it multiplies **n** by 3 and adds 1, then increments **moves**.
    - Returns **moves**.

- **Recursive Implementation**
  - Function **hailstoneLengthRec**
    - Takes an integer **n** as input.
    - If **n** is 1, returns 0.
    - If **n** is even, adds 1 to the result of calling **hailstoneLengthRec** with **n/2**.
    - If **n** is odd, adds 1 to the result of calling **hailstoneLengthRec** with **3*n + 1**.
Hailstone Sequence

<table>
<thead>
<tr>
<th>Iterative Implementation</th>
<th>Recursive Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static int hailstoneLengthImp(int n) {</td>
<td>public static int hailstoneLengthRec(int n) {</td>
</tr>
<tr>
<td>int moves = 0;</td>
<td>if (n == 1) {</td>
</tr>
<tr>
<td>while (n != 1) {</td>
<td>return 0;</td>
</tr>
<tr>
<td>if (isEven(n)) {</td>
<td>} else if (isEven(n)) {</td>
</tr>
<tr>
<td>n = n / 2;</td>
<td>return 1 + hailstoneLength(n/2);</td>
</tr>
<tr>
<td>} else {</td>
<td>} else {</td>
</tr>
<tr>
<td>n = 3 * n + 1;</td>
<td>return 1 + hailstoneLength(3*n + 1);</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
<tr>
<td>++moves;</td>
<td>}</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
<tr>
<td>return moves;</td>
<td>}</td>
</tr>
</tbody>
</table>

public static boolean isEven(int n) { return n % 2 == 0; }
Max Moves

What's the length of the longest hailstone sequence between start and end?

```java
/**
 * Returns the length of the longest hailstone sequence between start and end, inclusive.
 */
public static int maxMoves(int start, int end) {
    int maxMoves = 0;
    for(int i = start; i <= end; i++) {
        int moves = hailstoneLength(i);
        if(moves > maxMoves) {
            maxMoves = moves;
        }
    }
    return maxMoves;
}
```

Let's give it a try...
Three Techniques

Using System.out.println()

Using Eclipse
Add breakpoints, then Debug As...
Step through code
Conditional Breakpoints

Using Assertions
assert something;
Add “-ea” to VM arguments
Using Breakpoints

Right click on gray bar to left of line you want debugger to break at; select “Toggle Breakpoint”

To run in debug mode, right click class, choose “Debug As” > “Java Application”
Conditional Breakpoints

Right click on breakpoint and select “Breakpoint Properties”

Check “Enable Condition” and enter conditional expression in text box
Right click class, select "Run As" > "Run Configurations"
Enabling Assertions (2)

On “Arguments” tab, add “-ea” to the VM (virtual machine) arguments.

You can now add assertions in your code in the form “assert expression;”. 

Eclipse Debugger
Memoization

Use “Map” to store known values
Map store key->value pairs

When we calculate a value, store it in the Map

Check Map to see if we've already calculated value