CSE 8B Today

Upcoming schedule:
PSAs:
• PSA6 due tomorrow! (Wednesday 14th)
• PSA7 due Monday 19th
• PSA EXTRA (optional) due Monday 26th
• PSA8 (LAST!!) due Wednesday 4th June
Exams:
• Interm exam 4: Tuesday 27th May

Still waiting to hear about FINAL ROOM, I’ll update ASAP

THIS WEEK practice with

- Applets and Graphic letters!
  (keyboard events with KeyListener)

- Simple examples of recursion
REMEMBER: PART 1, FIRST DO ALL of it in DR. JAVA
Once it’s finished, try the commands to compile and see it as a web applet
CSE 8B Today

Recursion

As close as CS gets to magic
A few review ideas we need to refresh this week

• How methods work…
• Turtles and their Worlds!
How methods work…

public int demo(int x) {
    return x + f(x);
}
public int f(int x) {
    return 11*g(x) + g(x/2);
}
public int g(int x) {
    return -1 * x;
}

What is \texttt{demo(-4)} ?

Remember:

• Each method has its own scope (every time we call, it’s a new one!)
• If the method has a return value, we can use the “method” directly inside an expression

When we start run \texttt{demo(-4)}, this is what we have in memory…

\begin{align*}
\texttt{demo} \\
x &= -4 \\
\text{return } -4 + f(-4)
\end{align*}
What is `demo(-4)`?
What is \texttt{demo(-4)}? 

```java
public int demo(x) {
    return x + f(x);
}
public int f(x) {
    return 11*g(x) + g(x/2);
}
public int g(x) {
    return -1 * x;
}
```

And then this is in memory, when we get to call \texttt{f}

\begin{align*}
\texttt{demo} & \quad x = -4 \\
& \quad \text{return } -4 + \text{\texttt{f(-4)}}
\end{align*}

\begin{align*}
\texttt{f} & \quad x = -4 \\
& \quad \text{return } 11\texttt{g(x)} + \text{\texttt{g(x/2)}}
\end{align*}

These are different \texttt{x}'s!
public int demo(x) {
    return x + f(x);
}
public int f(x) {
    return 11*g(x) + g(x/2);
}
public int g(x) {
    return -1 * x;
}

What is \texttt{demo(-4)}? 

And then this is in memory, when we get to call g the first time
public int demo(x) {
    return x + f(x);
}

public int f(x) {
    return 11* g(x) + g(x/2);
}

public int g(x) {
    return -1 * x;
}

What is demo(-4)?
What is \texttt{demo(-4)}?
What is `demo(-4)`?

```java
public int demo(x) {
    return x + f(x);
}
public int f(x) {
    return 11*g(x) + g(x/2);
}
public int g(x) {
    return -1 * x;
}
```

How methods work…
public int demo(x) {
    return x + f(x);
}

public int f(x) {
    return 11*g(x) + g(x/2);
}

public int g(x) {
    return -1 * x;
}

What is \texttt{demo(-4)} ?
How methods work...

public int demo(x) {
    return x + f(x);
}
public int f(x) {
    return 11* g(x) + g(x/2);
}
public int g(x) {
    return -1 * x;
}

What is \texttt{demo(-4)} ?
What is demo(-4)?
Recursion

What does the above code do when we call `foo(5)`

A) Prints out 5
B) Prints out the numbers from 5 down to 1
C) Prints out the numbers from 1 up to 5
D) I have NO IDEA! How can you call `foo` from inside `foo`!?!?!

and if all the methods I call have the same name?

Well, they ARE actually the SAME method …
public void foo(int x) {
    if (x>1) {
        foo(x-1);
    }
    System.out.println(x);
    return; // not necessary
}
// call foo(5)

1
How to deal with recursion

• When writing a recursive method/algorithm you need to do two things:
  – **Identify a base case**
    • NO recursive call in the base case
    • A version of the problem so simple that you already know the answer, so just do it!
  – **The recursive step**
    • Assume that your function will WORK (magic!) for smaller versions of the input
    • Remember to do a small piece of the work
    • Remember that your new input must be heading toward your base case

• You can think about these two steps in any order. Whatever works for you!
(1) What does `chai` draw?
(No recursion here, just Turtle practice. Remember all Turtles need to be added to a World)

```java
void chai(Turtle t, int size) {
    t.forward(size);
    t.turnLeft();
    t.forward(size/2);
    t.turnRight();
    t.turnRight();
    t.forward(size);
    t.turnLeft();
    t.turnLeft();
    t.forward(size/2);
    t.turnRight();
    t.backward(size);
}
```

---

Turtle Graphics *Practice*

(2) Let’s finish `rwalk` so it draws a "stock-market" path of `nSteps` 10-pixel steps. *Use recursion!*

```java
import java.util.Random;
void rwalk(Turtle t, int nSteps):  
    if ( nSteps == 0 ) {  
        return;
    }
    Random r = new Random();
    if ( r.nextInt( 2 ) == 0 ) {  // turn left
        t.turn( -45 );
    }  
    else {  // turn right
    }
```

**Ex Cr:** How could you make it a bull (or a bear) market?

One possible result of `rw(20)`
Thinking *sequentially*

**factorial**

\[ 5! \, = \, 120 \]

\[ 5! \, = \, 5 \ast 4 \ast 3 \ast 2 \ast 1 \]

\[ N! \, = \, N \ast (N-1) \ast (N-2) \ast \ldots \ast 3 \ast 2 \ast 1 \]
Thinking *recursively*

**factorial**

\[ 5! = 120 \]
\[ 5! = 5 \times 4 \times 3 \times 2 \times 1 \]

\[ N! = N \times (N-1) \times (N-2) \times \ldots \times 3 \times 2 \times 1 \]
\[ N! = N \times (N-1)! \]
What is wrong with the following?

```java
public int fac(int n) {
    return n * fac(n-1);
}
```

A. A function cannot call itself in Java
B. It will run forever because it is missing a base case
C. It will not compile
D. It will not compute the correct value of factorial because it is missing the recursive step
Thinking *recursively*

**Factorial**

\[ N! = N \times (N-1) \times (N-2) \times \ldots \times 3 \times 2 \times 1 \]

\[ N! = N \times (N-1)! \]

(And \(1! = 1\))

Recursive step, do a tiny bit of work, and then assume you have code that solves a smaller version of the same problem.

Base case (what to do when the problem can’t be simplified any further)