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

CHECK STARTING CODE in last class slides
PSA7 – PART1

Extend a JApplet

message

Add your graphicLetters in one JPanel

Organize these two components in another JPanel

textfield clearButton
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...

```java
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 `demo(-4)`?

When we start run `demo(-4)`, this is what we have in memory...

```
demo
  x = -4
  return -4 + f(-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
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...

What is $\text{demo}(-4)$?

```
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 $f$

![Diagram showing the calculation process for $\text{demo}(-4)$]

These are different $x$'s!
How methods work...

```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;
}
```

What is `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) ?
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)} ?
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)}?
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) ?

-.demo
  x = -4
  return -4 + f(-4)

- f
  x = -4
  return 11* 4  + 2
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)}? 

```
demo
  x = -4
  return -4 + f(-4)

f
  x = -4
  return 11* 4  + 2
```
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 $\text{demo}(-4)$?
public void foo(int x) {
    if (x > 1) {
        foo(x - 1);
    }
    System.out.println(x);
    return
}

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 …

See why in the next slides…
public void foo(int x) {
    if (x>1){
        foo(x-1);
    }
    System.out.println(x);
    return; // not necessary
} // call foo(5)

1
Recursion and Stack frames

public void foo(int x) {
    if (x>1) {
        foo(x-1);
    }
    System.out.println(x);
    return;  // not necessary
}
// call foo(5)

1
2
public void foo(int x) {
    if (x>1){
        foo(x-1);
    }
    System.out.println(x);
    return; //not necessary
} // call foo(5)

1
2
3

```java
public void foo(int x) {
    if (x>1){
        foo(x-1);
    }
    System.out.println(x);
    return; //not necessary
}
```

```
foo
  x = 5
  foo(4)
     foo
       x = 4
       foo(3)
        foo
           x = 3
           foo(2)
           System.out.println(3)
```
Recursion and Stack frames

```java
public void foo(int x) {
    if (x>1) {
        foo(x-1);
    }
    System.out.println(x);
    return; //not necessary
}
// call foo(5)
```

1
2
3
4
public void foo(int x) {
    if (x>1){
        foo(x-1);
    }
    System.out.println(x);
    return; //not necessary
}
// call foo(5)

foo
    x = 5
    foo(4)
    System.out.println(5)
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 (UPDATE somehow the parameters that you input in the recursive call)

• You can think about these two steps in any order. Whatever works for you!
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);
}
```

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
        t.turn( 45 );
    }
```

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

**Download the sample code on the class website!**