Lectures 6 and 7
Tue Feb 11
Tue Feb 18 (today)
Proj3 DUE Feb 24
These lectures tell you how.
Lect 6 2-15
Lect 7 16-end
Useful computers are pre-programmed.
future diagram
What is this course about?

Serve students who want or need to actually program computers...
Designed for CS majors and minors...typically students headed for jobs or grad. studies (like CS Masters) where the expectation is “this candidate can program computers.”
Programming is like ice-skating, painting, playing sports or music, writing poetry and other creative hands-on endeavors in that is a real misery and is impossible to do well if you need to but don't want to.
Specific facts and skills that will be graded

- How YOU, in ICSI201 Spr14 now, can pre-program a computer in Java (+bookClasses) AND ACTUALLY DEMONSTRATE your computer DOING what we specify.

- Understand, apply, and answer exam questions about the exact rules and vocabulary for “How ...” plus what UNSEEN changes to stored data and other operations occur inside the computer.
Some specific professional “Hows” you must follow when YOU write Java in ICSI201

- Pre-program directions for DOING, in the FUTURE, big things inside DEFINITIONS OF YOUR OWN METHODS.
  - Pick, Place, Parametrize a sequence of
  - big (other methods calls) and
  - little things (Java primitive operations).

- Pre-program directions to ACTUALLY DO those big things by P.P.P. CALLS TO YOUR OWN METHODS inside YOUR Application's main method.
The Albany Way to pre-program your own methods (like club())

• (to start) extend G&E's Turtle class, like

```java
public class ArtisticTurtle extends Turtle {
    public ArtisticTurtle(World wref) {
        super(wref);
    }
    public void club(int sizeParam) {
        this.forward(sizeParam);
        /* ... */
        return;
        /* preprogrammed instr to draw one club */
    }
}
```

• (don't program stuff into Turtle.java, like the book says.)
The Albany Way to USE your own pre-programmed methods (like club( ))

• USE your extended Turtle in main

```java
public class Proj3App {
    public static void main(String[] a) {
        World wref = new World();
        ArtisticTurtle tref = new ArtisticTurtle(wref);
        tref.club(100);
        tref.turn(55);
        tref.club(140);
    }
}
```

• Don't waste time typing Java commands again and again (like the book says.) Make an App, show it off, improve your safely saved code, TEST IT ONCE MORE, upload it for grade credit. Then celebrate.
2A Make a plan!
Remember the goal.
Pick, Position, Parametrize
things you know might be pre-
programmed steps.

2B Translate your positioned steps
into Java code.

DO STUFF LIKE 2A  ON PAPER
scanned after class
REALLY DO THIS!!! It will make projects much much LESS FRUSTRATING!

Work out plans ON PAPER with PENCILS so you can erase and improve them!
public class ArtisticTurtle extends Turtle {

    public ArtisticTurtle(World wref) {
        super(wref);
    }

    public void club(int sizeParam) {
        this.forward( sizeParam );
        this.turn( 75 );
        this.forward( sizeParam/10 );
        //... omitted stuff
        return ;
        /*preprogrammed instr to draw one club and then return, ONLY WHEN the club method is called.*/
    }
}
public class Proj3App {

public static void main(String[] a) {

    World wref = new World();
    ArtisticTurtle tref = new ArtisticTurtle(wref);
    tref.club(100);
    /*Purpose: Actually draw the 1st club.*/
    tref.turn(55);
    tref.club(140);
    /*Purpose: Draw the 2nd club, bigger.*/
}
}
public class Proj3App
{
public static void main(String[] a)
{
    World wref = new World();
    ArtisticTurtle tref = new ArtisticTurtle(wref);
    tref.club(100);
    tref.turn(55);
    tref.club(140);
}
}

public class ArtisticTurtle extends Turtle
{
    public void club(int sizeParam)
    {
        this.forward(sizeParam);
        /* ... */
        return;
        /* preprogrammed instr to draw one club*/
    }
}
Live coding demo of starting Project 3!
(read the 4-page project assignment to review this and keep going)
2A Make a plan!
Remember the goal.
Pick, Position, Parametrize
things you know might be pre-programmed steps.

2B Translate your positioned steps into Java code.

DO STUFF LIKE 2A ON PAPER
scanned after class
REALLY DO THIS!!! It will make projects much much LESS FRUSTRATING!

Work out plans ON PAPER with PENCILS so you can erase and improve them!
Making 3 plans ON PAPER, WITHOUT PROGRAMMING REALLY COUNTS!!

Plan 1: Your improved golf club.
Plan 2: Your first original graphic design.
Plan 3: Your second original graphic design

MUST BE RECEIVED before YOUR CODE IS RECEIVED!!
public class GolfingTurtle extends Turtle
{
    public GolfingTurtle(World wref)
    {
        super( wref );
    }
    public void club( int sizeParamVar )
    {
        this.forward( sizeParamVar );
        //Purpose: Draw the club's handle.
        this.turn( 75 );
        //Purpose: Make the Turtle face in the head's direction.
        this.forward( sizeParamVar/10 );
        //Purpose: Draw the head.
        this.forward( - sizeParamVar/10 );
        //Purpose: Begin the retrace to bring the Turtle back
        //to its starting state.
        this.turn( -75 );
        //Purpose: Undo the turn made before.
        this.forward( - sizeParamVar );
        //Purpose: Finish the retrace.
        return;
        //Purpose: Pre-program that the instructions will
        //resume from where the method was called.
    }
    public static void main(String[] a)
    {
        System.out.println("GolfingTurtle is NOT AN App!");
    }
}
public class GolfClubDrawingApp
{
    public static void main(String[] a)
    {
        World wref = new World();
        GolfingTurtle tref = new GolfingTurtle( wref );
        tref.club( 200 ); //Draw a vertical club.
        tref.turn( 45 );  //Prepare to draw the 45 deg. club
        tref.club( 150 );  //Draw the 45 deg. one smaller.
        tref.turn( 90 );   //Prepare to draw a slant downward club.
        tref.club( 270 );  //Draw the down slanted club longest.
        tref.turn( - (45+90)); //Restore Turtle's original state.
    }
}
Vocabulary word 1

Method DEFINITION (not call)

```java
public class ArtisticTurtle extends Turtle {
    public ArtisticTurtle(World wref) {
        super(wref);
    }
    public void club(int sizeParam) {
        this.forward(sizeParam);
        this.turn(75);
        this.forward(sizeParam/10);
        //... omitted stuff
        return;
        /*preprogrammed instr to draw one club
        and then return, ONLY WHEN the club
        method is called.*/
    }
}
```
Method CALL (not definition)

```java
class Proj3App {
    public static void main(String[] a) {
        World wref = new World();
        ArtisticTurtle tref = new ArtisticTurtle(wref);
        tref.club(100);
        /*Purpose: Actually draw the 1\textsuperscript{st} club.*/
        tref.turn(55);
        tref.club(140);
        /*Purpose: Draw the 2\textsuperscript{nd} club, bigger.*/
    }
}
```

A second, different CALL to the same method named \texttt{club} belonging to 
\texttt{ArtisticTurtles}. 
public class Proj3App
{
    public static void main(String[] a)
    {
        World wref = new World();
        ArtisticTurtle tref = new ArtisticTurtle(wref);
        tref.club(100);
        tref.turn(55);
        tref.club(140);
    }
}

public class ArtisticTurtle extends Turtle
{
    public void club(int sizeParam)
    {
        this.forward(sizeParam);
        /* ... */
        /* preprogrammed instr to draw one club */
        return;
    }
}
public class DoWhatBossWantsApp {
    public static void main(String[] a) {
        World wref = new World();
        BossableTurtle tref = new BossableTurtle( wref );
        tref.helloAndLine( 100 );
        //Purpose: Make the computer do
        //what's pre-programmed
        //in the helloAndLine method.
    }
}

This week's lab: Save As ... DoWhatBossWantsApp.java

WHEN (not before!) you click Run—The computer will sequentially
First: Make instantiate a new World object.
Second: Make a new BossableTurtle “living” inside that World object
THIRD: DO a big thing: CALL the method named helloAndLine on that BossableTurtle referred to by tref.
public class BossableTurtle extends Turtle {
    public BossableTurtle(World wref) {
        super(wref);
    }

    public void helloAndLine(int sizeParam) {
        System.out.println("Hello.");
        //Purpose: Demonstrate printing.
        this.forward(sizeParam);
        //Purpose: Demonstrate line drawing..
        return; //Purpose: Stop ...
    }
}

WHEN the computer DOES that big thing, it follows these instructions sequentially
First: print “Hello.”
Second: Make this Turtle draw one line.
Third: Stop these instructions and resume from code after the call
public class DoWhatBossWantsApp
{
    public static void main(String[] a)
    {
        World wref = new World();
        BossableTurtle tref = new BossableTurtle( wref );
        tref.helloAndLine( 100 );
        //Purpose: Make the computer do
        //what's pre-programmed
        //in the helloAndLine method.
    }
}

public class BossableTurtle extends Turtle
{
    public BossableTurtle(World wref)
    {
        super(wref);
    }
    public void helloAndLine(int sizeParam)
    {
        System.out.println("Hello.");
        //Purpose: Demonstrate printing.
        this.forward( sizeParam );
        //Purpose: Demonstrate line drawing...
        return ; //Purpose: Stop ... .
    }
}
public class DoWhatBossWantsApp {
    public static void main(String[] a) {
        World wref = new World();
        BossableTurtle tref = new BossableTurtle(wref);
        tref.helloAndLine(100);
        //Purpose: Make the computer do what's pre-programmed in the helloAndLine method.
    }
}

public class BossableTurtle extends Turtle {
    public BossableTurtle(World wref) {
        super(wref);
    }
    public void helloAndLine(int sizeParam) {
        System.out.println("Hello.");
        //Purpose: Demonstrate printing.
        this.forward(sizeParam);
        //Purpose: Demonstrate line drawing.
        return; //Purpose: Stop...
    }
}

Sequential control flow
Method CALL
Method DEFINITION
Concepts

- Pre-programming. Planning time, coding time, compile time, run time.
- Sequential \((1,2,3,4,5,6,...)\) execution during run time.
- Method defining: pre-programming a big thing.
  - Done by Profs G&E in their 10 year old book
  - Done by us in our \texttt{BossableTurtle} or \texttt{ArtisticTurtle}
- Method calling: Statements executed in the sequence they are written except for
  - jumping TO a method body during a method call.
  - jumping FROM the body back during a method return.
public class Proj3App
{
    public static void main(String[] a)
    {
        World wref = new World();
        ArtisticTurtle tref = new ArtisticTurtle(wref);
        tref.club( 100 );
        tref.turn( 55 );
        tref.club( 140 );
    }
}

class ArtisticTurtle extends Turtle
{
    public void club(int sizeParam)
    {
        this.forward( sizeParam );
        /* ... */
        return;
        /* preprogrammed instr to draw one club*/
    }
}
public class DoWhatBossWantsApp {
    public static void main(String[] a) {
        //What a app does is preprogrammed in main.
        World wref = new World();
        BossableTurtle tref = new BossableTurtle( wref );
        tref.helloAndLine( 100 );
        //Purpose: Make the computer do
        //what's pre-programmed
        //in the helloAndLine method.
    }
}

iclicker: We know calling the helloAndLine method makes the computer print hello and draw a line. Where do those instructions for printing and drawing come from?
(A) The DoWhatBossWants class
(B) The book classes
(C) The Turtle class
(D) The BossableTurtle class
(E) The World class
public class GolfingTurtle extends Turtle
{
    public GolfingTurtle(World wref)
    { super( wref ); }  
    public void club( int sizeParamVar )
    {
        this.forward( sizeParamVar );
        //Purpose: Draw the club's handle.
        this.turn( 75 );
        //Purpose: Make the Turtle face in the head's direction.
        this.forward( sizeParamVar/10 );
        //Purpose: Draw the head.
        this.forward( - sizeParamVar/10 );
        //Purpose: Begin the retrace to bring the Turtle back
        //to its starting state.
        this.turn( -75 );
        //Purpose: Undo the turn made before.
        this.forward( - sizeParamVar );
        //Purpose: Finish the retrace.
        return ;
        //Purpose: Pre-program that the instructions will
        //resume from where the method was called.
    }
    public static void main(String[] a)
    {
        System.out.println("GolfingTurtle is NOT AN App!");
    }
}
public class GolfingTurtle extends Turtle {

    public GolfingTurtle(World wref) {
        super(wref);
    }

    public void club(int sizeParamVar) {
        this.forward(sizeParamVar);
        //Purpose: Draw the club's handle.
        this.turn(75);
        //Purpose: Make the Turtle face in the head's direction.
        this.forward(sizeParamVar/10);
        //Purpose: Draw the head.
        this.forward(-sizeParamVar/10);
        //Purpose: Begin the retrace to bring the Turtle back to its starting state.
        this.turn(-75);
        //Purpose: Undo the turn made before.
        this.forward(-sizeParamVar);
        //Purpose: Finish the retrace.
        return;
        //Purpose: Pre-program that the instructions will resume from where the method was called.
    }

    public static void main(String[] a) {
        System.out.println("GolfingTurtle is NOT AN App!");
    }
}
public class GolfingTurtle extends Turtle
{
    public GolfingTurtle(World wref)
    {
        super( wref );
    }
    public void club( int sizeParamVar )
    {
        this.forward( sizeParamVar );
        this.turn( 75 );
        this.forward( sizeParamVar/10 );
        this.forward( - sizeParamVar/10 );
        this.turn( -75 );
        //Purpose: Undo the turn made before.
        this.forward( - sizeParamVar );
        return;
    }
    public static void main(String[] a)
    {
        System.out.println("GolfingTurtle is NOT AN App!");
    }
}
Professional Practices taught in live code demo.

- Fresh, empty, separate directory/folder for each project.
- Incremental compiling and development—add just a little code and compile over and over.
- Indent to make the logical structure OBVIOUS.
- Document what a method is for and how SOMEBODY ELSE should use it in a Javadoc comment.
public class DoWhatBossWantsApp
{
    public static void main(String[] a)
    {
        // What a app does is preprogrammed in main.
        World wref = new World();
        BossableTurtle tref = new BossableTurtle( wref );
        tref.helloAndLine( 100 );
        // Purpose: Make the computer do
        // what's pre-programmed
        // in the helloAndLine method.
    }
}

iclicker: We know calling the helloAndLine method makes the computer print hello and draw a line. Where do those instructions for printing and drawing come from?
(A) The DoWhatBossWants class
(B) The book classes
(C) The Turtle class
(D) The BossableTurtle class
(E) The World class
Let's develop an ArtisticTurtle method to draw a pentagonal spiral we plan first...
Plan: Do the Math of the pentagon

The Turtle must turn an amount of turn (degrees) so that

\(? + ? + ? + ? + ?\)

is 5 times ?

is 360 degrees.

So the amount of angle for each turn must be 360/5 or 72 degrees.
Plan: how a parameter controls size

`sizeParam` (the value of it) is the number somebody wrote in the method CALL.

Make each of the 5 lines have length given by `sizeParam`:

```java
this.forward(sizeParam);  // will make one line with length `sizeParam`
```
Plan: figure out what to repeat.
We know it's 5 `this.forward(sizeParam);`
but what about the turns?
Plan: what to repeat

72
72
72
72
72
72
72
72
72
72

this.forward(sizeParam);
this.turn(72);
Plan: what to repeat

Check: Each repetition is a different color

This code snippet is likely related to a task or project involving graphics or animation, where 'forward' and 'turn' methods are used to manipulate a cursor or pointer in a specific pattern.
Plan: how many times?

5

this.forward(sizeParam);
this.turn(72);
Let's code it..
Let's make a 5 sided dome
72 degrees is too big an angle
Let's make a 5 sided dome. 72 degrees is too big an angle. PAPER IS MUCH EASIER TO WORK WITH.
Next...

- See the result of making the angles 36 degrees.
- Plan how the dome can be drawn horizontally.
- See it drawn horizontally.
- Make variables to save the original x position and y position; use them to move turtle back to its original position.
- Do the same for the heading.
- Move to the start state with the pen up, and put it down again.
Write a literal representing the integer value zero.
A)  zero
B)  0
C)  int literal;
D)  literal = 0;
E)  public class Demo
   {
       public static void main(String[] a)
       {
           int liberty;
           liberty = 0;
           System.out.println( liberty );
       }
   }
public class Demo {

    public static void main(String[] a) {
        int liberty = 0;
        System.out.println( liberty );
    }
}

The complete definition of an application container class named Demo

The definition of the (static) method named main
what's what..

A) zero  The Java identifier named zero
B) 0    The literal representing int value zero
C) int literal;   A declaration
D) literal = 0;  An assignment or PLEASE COPY statement, the command to copy data in memory.
E) public class Demo
   {
       public static void main(String[] a)
       {
           int liberty;
           liberty = 0;
           System.out.println( liberty );
       }
   }  A complete Java application program
what's what..and what they make the computer do

A) `int liberty;` A declaration

It makes the computer lay out or set up a piece of RAM (memory) for storing int type data. That piece is called a variable; and here its name is `liberty`

C) `liberty = 0;` An assignment or PLEASE COPY statement; command to copy data in memory.

WHEN the program containing this statement RUNS, the computer COPIES the int data value 0 into the RAM variable named `liberty`
Circle the \textbf{literal} that represents the integer value 0 within the complete program below.

```java
public class Demo {
    public static void main(String[] a) {
        int liberty;
        liberty = 0;
        System.out.println(liberty);
    }
}
```
A complete ASSIGNMENT STATEMENT that, when run, COPIES the integer value zero into the variable named liberty as that variable's value.
Beginners' Types of literals

- integer -- written as a sequence of one or more digits.
- String -- written as a sequence of zero or more of any character(s), enclosed between “ “

Examples: “hello” “good bye!” “00” “0” “9” “ “including “ “ the infamous empty string!"
other types of literals

- Floating point number -- written with a decimal point and maybe an exponent (see books) (The computer's style of scientific notation.)
  
  \[ 37.3256 \text{ or } 0.017 \] everywhere, in Java programs too.

  \[ 6.02 \times 10^{23} \] in a chem book is \[ 6.02e+23 \] in a Java program

- character -- written as EXACTLY ONE character between single quotes—chars are INSIDE Strings but they are NOT Strings themselves.
Concepts

- Pre-programming.
- Sequential \((1, 2, 3, 4, 5, 6, \ldots)\) execution.
- Method defining.
- Method calling.
- Syntactic elements: symbols like \(; = +\)
  
  keywords like `class`, `extends`, `public`, `void`, `static`
  
  non-keyword names like `lots dPet8U`
  
  literals like `0 98 67.83 "I Am A String"`

Nested elements expressed by MATCHING parenthesis-type symbols.

\[
\{ ( ) \{ ( [ ] ) ( ) \} \}
\]
public class Demo {
    
    public static void main(String[] a) {
        
        int liberty;
        liberty = 0;
        System.out.println(liberty);
    }
}
Nesting is expressed by MATCHING
\{ ( ) \{ ( [ ] ) ( ) \} \} parenthesis-type symbols.

Like Russian Matryoshka dolls—nesting expressed by matching top and bottom symbols is ALL OVER Java.
public class Proj3App
{
    public static void main(String[] a)
    {
        World wref = new World();
        ArtisticTurtle tref = new ArtisticTurtle(wref);
        tref.club(100);
        tref.turn(55);
        tref.club(140);
    }
}

iclicker: We know calling the `club` method makes the turtle draw some lines. Where do those instructions for drawing come from?

(A) The `Proj4App` class
(B) The book classes
(C) The `Turtle` class
(D) The `ArtisticTurtle` class
(E) The `World` class