The Three Easy Pieces of Beginning Computer Programming

What do you think they are? Write your name and 1 up to 3 on the card, classmates on the aisles collect the cards. (Discussion invited!)

In 5 min we will scan and record YOUR choices, and hear about the Profs.
Reported class choices filled in after the lecture.

link to a web page for the choices made by the current class (under const)
C's 3 Pieces of Programming

1. Pick
2. Position
3. Parametrize
Pick an available operation, method call or pattern (like making a Java App or calling forward on a Turtle).

Mystery: What are your choices??
HOW do students and professionals find them out?

2. Position
Position where you type your pick.

Figure out the spot in your written program where you want the computer to do the thing you picked.

DON'T CHEAT by asking someone else to think it through for you.
3. Parametrize

**Parametrize**: Figure out and type in parameters or arguments or other things to specify details, like your program's name and how far the Turtle should go forward.

Use documentation (and other notes) to know what the parameters mean and might be.

DON'T CHEAT by asking someone else to think for you of what values to type, like 150 versus 100.
MEMORIZE the Java App pick:

```java
public class MyApp {
    public static void main(String[] a) {
        // Directions, written in good Java syntax, for what the computer will do when it runs your App.
    }
}
```

a name (you make up) for your app, say `MyApp`.

Save it as a file named `MyApp.java`.

(then in 1 year forget this and memorize the one for C)
MEMORIZE the pick for printing something (a big thing)

System.out.println();

Expression that expresses what the computer will print.
MEMORIZE the pick for printing something (a big thing)

`System.out.println( );`

Expression that expresses what the computer will print.

(What's expressed) is an example of a parameter value.
The same method (named `println`) can print A HUGE VARIETY of messages, depending on what value you program here.

MEMORIZE the pick for printing something (a big thing)!

```java
System.out.println();
```
Memorize or make notes on picking things that make the computer draw graphics.

```java
World wref =
    new World( );
Turtle tref =
    new Turtle( wref );
```

The bookClasses must be downloaded, unzipped, and DrJava's extra classpath resource preference MUST be set to the bookClasses folder for this to work.
The bookClasses must be downloaded, unzipped, and DrJava's extra classpath resource preference MUST be set to the bookClasses folder for this to work.

One goal of Proj1 and Lab2 is everyone to get this necessary IT step done once and for all the rest of the semester...leaving your attention to what's harder to learn but what you'll use throughout some careers.

Lab2 is your last chance (Wed and Thu) if you didn't finish Proj1!
Option: make a World whose Width and Height YOU CHOOSE with one Turtle in it.

World wref = new World();
Turtle tref = new Turtle( wref );
World wref =
    new World(1, 2);
Turtle tref =
    new Turtle(wref);
tref.forward(3);
tref.turn(4);
tref.forward(5);
tref.forward(6);

6 spots where you type in parameter values. Each value controls a detail about what a method call makes the computer do.
iclicker: Be sensitive to Java's regularities. What's common to all parameters (of methods)?

A) Must always be “quoted like this”
B) Must never be “quoted like this”
C) Must always be numbers.
D) Must always be between a ( and a )
E) Must always be variables with lower case names.
Be sensitive to Java's regularities. What's common to all parameters (of methods)?

A)  
B)  
C)  
D) Must always be between a ( and a )  
E)  

This is one of the Java syntax rules that you MUST MEMORIZE! (It's easy when you get the idea and have practiced with parameters a few times.)
3 method calls (3 big things):
3 Java instructions coded one after the other.

3 spots where you type in parameter values. Each value controls a detail about what a method call makes the computer do.
What is a method call?

- It is the instruction to make the computer do a big thing when the computer runs your program.

- A kind of computer instruction that tells the computer to RUN OTHER INSTRUCTIONS already written SOMEWHERE ELSE, and then RETURN back to the instruction after the method call.
Tale of 3 professors..

- Profs Guzdial and Ericson wrote those other instructions in Java (for the benefit of you and I) to make *Turtles* go forward drawing lines, and to turn changing their heading angle.

- Prof. Chaiken will write the instructions to make Turtles draw a golf club whenever your (or his) programs run the method call instruction:

  \[ \text{tref.club(} \text{...} \text{);} \]

- The parameter value will control the club's size.
1. Like Profs Guzdial & Ericson created book classes like World, Turtle, SimpleTurtle, I will create one class named GolfingTurtle.

2. My GolfingTurtle class will be software that enables a computer to have inside itself a Turtle object that can draw golf clubs.

3. I will create a 2nd application container class that will BE an application program that draws golf clubs when somebody runs it.
1. Glimpsing at those 2 Profs' work done 10 years ago...don't get scared or turned off.. (you might write library code for other people to use after a few YEARS of practice.)

Navigate in DrJava or a text editor like WordPad into the bookClasses directory/folder and look at files like Turtle.java, SimpleTurtle.java, World.java
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 will scan 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 GolfingTurtle extends Turtle {
	public GolfingTurtle(World wref) {
		super(wref);
	}
	public void club(int sizeParamVar) {
		this.forward(sizeParamVar);
	//Purpose: Draw the club's handle.
		hthis.turn(75);
	//Purpose: Make the Turtle face in the head's direction.
		hthis.forward(sizeParamVar/10);
	//Purpose: Draw the head.
		hthis.forward(-sizeParamVar/10);
	//Purpose: Begin the retrace to bring the Turtle back
	//to its starting state.
		hthis.turn(-75);
	//Purpose: Undo the turn made before.
		hthis.forward(-sizeParamVar);
	//Purpose: Finish the retrace.
	return;
	//Purpose: Pre-program that the instructions will
	//resume from where the method was called.
	}
}
Every statement HAS A PURPOSE

Real programmers ALWAYS think the purpose for EVERY LINE THEY WRITE. They SOMETIMES speak it team-mates and/or write the purpose in comments, especially for tricky stuff.
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.
    }
}
G&E Intro to Prog. Chapter
UA ch2, G&E ch3
Why & How to read it for UAlbany's 201

• Get IDEAS of what you can program a Turtle to do, and tips on how.

• NEVER copy textbook code verbatim. The Albany way is to always create complete applications you can save, run, explain, show off, and improve again and again. They don't do that in this chapter.

• Briefly read in other prof's words about the concepts: data, names, types, files, executing object methods, defining classes, creating objects, state of an object and how objects control their state.
World wref =
   new World(   ,   );
Turtle tref =
   new Turtle( wref );
tref.forward(   );
tref.turn(   );
tref.forward(   );

6 spots where you type in parameter values. Each value controls a detail about what a method call makes the computer do.
iclicker: Be sensitive to Java's regularities. What's common to all parameters (of methods)?

A) Must always be “quoted like this”
B) Must never be “quoted like this”
C) Must always be numbers.
D) Must always be between a ( and a )
E) Must always be variables with lower case names.
Be sensitive to Java's regularities. What's common to all parameters (of methods)?

A) 
B) 
C) 
D) Must always be between a ( and a ) 
E) 

This is one of the Java syntax rules that you MUST MEMORIZE! (It's easy when you get the idea and have practiced with parameters a few times.)
3 spots where the code gives *parameter values* in method calls.

tref.forward(1);
tref.turn(2);
tref.forward(3);

Saying “make the computer do a big thing” IS THE SAME AS

Saying “call a method”,
saying “executing a method”,
saying “invoking a method”,
and saying “sending a message to an object asking it to do something”.