Current Readings

- Albany Ch2 (G&E Introduction to Programming) ONLY to get ideas of what you can program a Turtle to do, plus review of method ideas. DON'T literally copy code examples since at Albany, we code complete Java applications.

- Albany Ch3 (Gaddis Java Fundamentals) Study and use detailed facts WHEN YOU NEED THEM. This also for Ch8 (Gaddis Methods)

- Albany Ch4 (G&E Modifying Pictures Using Loops) ONLY (1) get the idea of how digital Colors and images work and (2) read IDEA ONLY of some Picture and Pixel methods. DON'T JUST COPY G&E CODE for Lab5.
We can pre-program a method so the computer uses variables to REMEMBER some numbers and then USE them later.

Plan: Example of this opportunity makes programming of a dome drawing method much easier.
From last week...
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`:

```
this.forward(sizeParam);
```

will make one line with length `sizeParam`
Plan: what to repeat

this.forward(sizeParam);
this.turn(72);
Plan: what to repeat
Check: Each repetition is a different color

```javascript
this.forward(sizeParam);
this.turn(72);
```
Plan: how many times?

5

this.forward(sizeParam);
this.turn(72);
Let's code it..
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!”);
    }
}
public void pentagon( int sizeParamVar )
{
    this.forward( sizeParamVar );
    this.turn( 72 );
    this.forward( sizeParamVar );
    this.turn( 72 );
    this.forward( sizeParamVar );
    this.turn( 72 );
    this.forward( sizeParamVar );
    this.turn( 72 );
    this.forward( sizeParamVar );
    this.turn( 72 );
}
Hay..
Where did that integer value 100 come from?
good question..
Here is the application program.

```java
public class Lect08App {
    public static void main(String[] a) {
        World wref = new World();
        ArtisticTurtle tref = new ArtisticTurtle( wref );
        tref.pentagon( 100 );
        //Purpose: Draw one size 100 pentagon club.
    }
}
```

Java statement that CALLS the pentagon method.

PARAMETER VALUE 100 (coded by a literal).
What if somebody coded 59 instead?

public class Lect08App
{
    public static void main(String[] a)
    {
        World wref = new World();
        ArtisticTurtle tref = new ArtisticTurtle( wref );
        tref.pentagon( 59 );
        //Purpose: Draw one size 59
        //pentagon club.
    }
}

Java statement that CALLS the pentagon method.

PARAMETER VALUE 59 (coded by a literal).
public void pentagon( int sizeParamVar )
{
    this.forward( sizeParamVar );
    this.turn( 72 );
    this.forward( sizeParamVar );
    this.turn( 72 );
    this.forward( sizeParamVar );
    this.turn( 72 );
    this.forward( sizeParamVar );
    this.turn( 72 );
    this.forward( sizeParamVar );
    this.turn( 72 );
}
Lecture 09
Current Readings

- Albany Ch2 (G&E Introduction to Programming) ONLY to get ideas of what you can program a Turtle to do, plus review of method ideas. DON'T literally copy code examples since at Albany, we code complete Java applications.

- Albany Ch3 (Gaddis Java Fundamentals) Study and use detailed facts WHEN YOU NEED THEM. This also for Ch8 (Gaddis Methods)

- more: (3) READ ABOUT LOOPS... Albany Ch4 (G&E Modifying Pictures Using Loops) (1) get the idea of how digital Colors and images work and (2) read IDEA ONLY of some Picture and Pixel methods. DON'T JUST COPY G&E CODE for Labs and Projects...
We can pre-program a method so the computer uses variables to REMEMBER some numbers and then USE them later.

Plan: Example of this opportunity makes programming of a dome drawing method much easier.

First: Review of the Parameter Idea
Here is the application program...

```java
public class Lect08App {
    public static void main(String[] a) {
        World wref = new World();
        ArtisticTurtle tref = new ArtisticTurtle( wref );
        tref.pentagon( 100 );
        //Purpose: Draw one size 100 pentagon club.
    }
}
```

Java statement that CALLS the pentagon method.

PARAMETER VALUE 100 (coded by a literal).
public class Lect08App
{
    public static void main(String[] a)
    {
        World wref = new World();
        ArtisticTurtle tref = new ArtisticTurtle( wref );
        tref.pentagon( 59 );
        //Purpose: Draw one size 59 pentagon club.
    }
}
public void pentagon( int sizeParamVar )
{
    this.forward( sizeParamVar );
    this.turn( 72 );
    this.forward( sizeParamVar );
    this.turn( 72 );
    this.forward( sizeParamVar );
    this.turn( 72 );
    this.forward( sizeParamVar );
    this.turn( 72 );
    this.forward( sizeParamVar );
    this.turn( 72 );
}
We used a parameter variable to maintain the temporary copy of the parameter value from the caller. That's how ONE SAME method can use MANY DIFFERENT parameter values when that one method is called MANY DIFFERENT times.
A new challenge: 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
5 turns of 36 degrees makes a total turn of $5 \times 36$ or 180 degrees, so our Turtle should end up facing the way opposite from its start.
Let's try it..
How much should we tilt it first, so it ends up looking like a dome?
?? + 36 + 36 + 36 + 36 + ?? we want to sum to 180
2*?? + 144 we want to sum to 180
2*?? we want to be 180-144 which is 36
The solution for ?? is 18

Applying algebra:
1) IMAGINE the problem solved.
2) Use symbols like ?? for unknown quantities.
3) Figure out math facts about the unknown quantities.
4) Use algebra to figure out what the unknown must be.
The computer steps must be

(1) turn 18 deg;
(2) repeat four times
   { forward( param ); turn 36; }
(3) turn 18 deg;
(4) method RETURN

Let's try it..
We can pre-program a method so the computer uses variables to REMEMBER some numbers and then USE them later.

Plan: Example of this opportunity makes programming of a dome drawing method much easier.
public class ArtisticTurtle extends Turtle
{
public ArtisticTurtle(World wref) {
  super(wref);
}
public void dome(int sizeParam)
{
  this.turn(18);
  this.forward(sizeParam);  // Draw the first line.
  this.turn(72/2);  // 1st angle is \(\frac{1}{2}\) pentagon's
  // Repeat the forward/turn sequence 4 more times.
  this.forward(sizeParam);
  this.turn(72/2);  // 2nd time
  this.forward(sizeParam);
  this.turn(72/2);  // 3rd time
  this.forward(sizeParam);
  this.turn(72/2);  // 4th time
  this.forward(sizeParam);
  this.turn(18);  // 5th and last time.
  // Thought: Total amount of turn is 180 degrees.
  return;
  // Return to the code of the caller.
}
Problem:
It takes too much math to figure out how far to move the Turtle so it goes back to its original state.

(A Turtle's state includes its x and y positions and its angle of heading.)
public class ArtisticTurtle extends Turtle {
    public ArtisticTurtle(World wref) {
        super(wref);
    }

    public void dome(int sizeParam) {

        int xOrig; int yOrig; // Make 2 VARIABLES
        // to REMEMBER where the Turtle should go back to.
        xOrig = this.getXPos(); // Actually remember x...
        yOrig = this.getYPos(); // Actually remember y...

        this.turn(18);
        this.forward(sizeParam); // Draw the first line.
        this.turn(72/2); // 1st angle is ½ pentagon's
        // Repeat the forward/turn sequence 4 more times.
        this.forward(sizeParam);
        this.turn(72/2); // 2nd time
        this.forward(sizeParam);
        this.turn(72/2); // 3rd time
        this.forward(sizeParam);
        this.turn(72/2); // 4th time
        this.forward(sizeParam);
        this.turn(18); // 5th and last time.

        // Thought: Total amount of turn is 180 degrees.
        this.moveTo(xOrig, yOrig);
        // Move to REMEMBERED position.
        return; // Return to the code of the caller.
    }
}
We pre-programmed a method so the computer uses variables to REMEMBER some numbers and then USEs them later.

Our example of this opportunity made programming of our dome drawing method much easier.
public class ArtisticTurtle extends Turtle {

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

    public void dome( int sizeParam )
    {
        this.turn( 18 );
        this.forward( sizeParam );
        //Draw the first line.
        this.turn( 72/2 );
        //Turn the 1st ang., 1/2 the pentagon amt.
        //Repeat the forward/turn sequence 4 more
        //times.
        //code omitted.
        this.turn( 18 );
        return ;
    }
}

forward and turn method
DEFINITIONS were written
by G&E in the book Classes.

look into SimpleTurtle.java if we have
time, and find getXPos, getYPos,
moveto method definitions..
Variables can also be used to remember references to objects.
Java code: `GolfingTurtle tref = new GolfingTurtle(wref);`

sets up the variable named `tref` and then copies the ArtisticTurtle's location into `tref` as its value.

WHEN we first (1) compile AND THEN second (2) 
RUN it, 
WHAT HAPPENS???
An instance of the `GolfingTurtle` class is created in the `World` where, the location, in computer memory that `GolfingTurtle` object is placed.

Java code `GolfingTurtle tref=new GolfingTurtle(wref);` sets up the variable named `tref` and then copies the `ArtisticTurtle`'s location into `tref` as its value.
Here is the application program..
Version 2

```java
public class GolfClubDrawingApp {
    public static void main(String[] a) {
        World wref = new World();
        GolfingTurtle tref = new GolfingTurtle( wref );
        tref.club( 200 );
        //Purpose: Draw one golf club.
    }
}
```

WHEN we first (1) compile AND
THEN second (2) RUN it,
WHAT HAPPENS This time???
BEFORE `tref.club( 200 );`
is executed

where, the location, in computer memory that GolfingTurtle object is
AFTER

tref.club( 200 );

is executed.

where, the location, in computer memory that GolfingTurtle object is 200 pixels in size
3 applications of variables

• A parameter variable stores, holds, remembers a parameter value. (eg. \texttt{sizeParamVar})

• A variable stores, holds, remembers some data so the method can use that data later. (eg. \texttt{xOrig, yOrig})

• A variable stores, holds, remembers a reference to an object (where an object is inside the computer) so methods can be called ON that PARTICULAR OBJECT (eg. \texttt{tref and pixRef})
public class ArtisticTurtle extends Turtle
{
    public ArtisticTurtle(World wref)
    { super(wref); }
    public void dome(int sizeParam)
    {
        int xOrig;
        int yOrig;
        xOrig = this.getXPos();
        yOrig = this.getYPos();
        this.turn(18);
        this.forward(sizeParam); //1st
        this.turn(72/2);
        this.forward(sizeParam); //2nd
        this.turn(72/2);
        this.forward(sizeParam); //3rd
        this.turn(72/2);
        this.forward(sizeParam); //4th
        this.turn(72/2);
        this.forward(sizeParam); //The last..
        this.turn(18); //has a different angle.
        this.turn(180);
        this.penUp();
        this.moveTo(xOriginal, yOriginal);
        this.penDown();
        return;
        //Return to the code of the caller.
    }
}
public class ArtisticTurtle extends Turtle
{
    public ArtisticTurtle(World wref)
    { super(wref); }
    public void dome( int sizeParam )
    {
        int xOrig;
        int yOrig;
        xOrig = this.getXPos();
        yOrig = this.getYPos();
        this.turn( 18 );
        this.forward( sizeParam ); //1st
        this.turn( 72/2 );
        this.forward( sizeParam ); //2nd
        this.turn( 72/2 );
        this.forward( sizeParam ); //3rd
        this.turn( 72/2 );
        this.forward( sizeParam ); //4th
        this.turn( 72/2 );
        this.forward( sizeParam ); //The last..
        this.turn( 18 ); //has a different angle.
        this.turn( 180 );
        this.penUp();
        this.moveTo( xOriginal, yOriginal );
        this.penDown();
        return;
        //Return to the code of the caller.
    }
}
Hey, computer! Please REPEAT the block of code below 4 times.

```javascript
{  
  this.forward( sizeParam );
  this.turn( 72/2 );
}
```

Computer: Duh, sorry. I'm TOO DUMB to know how to do that.
int numberOfMoreTimes;

    Can do: Set up a variable.

numberOfMoreTimes = 4;

    Copy 4 into that variable.

while ( numberOfMoreTimes > 0 )

    Tell whether the count of more times is zero or not.
    When not, repeat the body below and tell again...
{
    this.forward( sizeParam );
    this.turn( 75/2 );
    numberOfMoreTimes=numberOfMoreTimes–1;

        Subtract 1 from the memorized count.

    Computer: Duh, sorry. I'm TOO DUMB
    to know how to repeat 4 times. But, I can do what you preprogrammed above with a variable.
}
YOU must MEMORIZE (1)

Here's how you write a DECLARATION that, when the code runs,
(a) SETS UP in computer memory A VARIABLE that can memorize int values (32 bit signed binary in Java)
(b) GIVES that VARIABLE the name YOU choose, like

    numberOfMoreTimes

int numberOfMoreTimes;

Can do: Set up a variable.
(Of course, any other legal name is fine too, not just numberOfMoreTimes.)
YOU must MEMORIZE (2)

After

```java
int numberOfMoreTimes;
here's how you write an ASSIGNMENT STATEMENT
that,
when the code runs,
    COPIES the literal value 4 into the variable named
        numberOfMoreTimes
```

```java
numberOfMoreTimes = 4;
    Copy 4 into that variable.
```
YOU must MEMORIZE (3)

After
int numberOfMoreTimes;
numberOfMoreTimes = 4;
here's how you write a WHILE TEST that, when the code runs,
(a) Tests whether the value of numberOfMoreTimes > 0
(b) When that test is true, repeat the body the first or another time and then do the test again.
(c) When that test is false, stop the repeating.

while ( numberOfMoreTimes > 0 )
{
  this.forward(sizeParam);
  this.turn(72/2);
  numberOfMoreTimes=numberOfMoreTimes-1;
}
In
int numberOfMoreTimes;
numberOfMoreTimes = 4;
while ( numberOfMoreTimes > 0 ) {
    this.forward(sizeParam);
    this.turn(72/2);
    numberOfMoreTimes=numberOfMoreTimes-1;
    Subtract 1 from the memorized count.
}

MEMORIZE (4) How to pre-program so WHEN THE LOOP RUNS, the computer will subtract 1 from the memorized count after each repetition.
int numberOfMoreTimes;

    Can do: Set up a variable.
numberOfMoreTimes = 4;

    Copy 4 into that variable.
while ( numberOfMoreTimes > 0 )
    Tell whether the count of more times is zero or not.
{
    this.forward( sizeParam );
    this.turn( 75/2 );
    numberOfMoreTimes=numberOfMoreTimes–1;
        Subtract 1 from the memorized count.
}

Computer: Duh, sorry. I'm TOO DUMB
to know how to repeat 4 times. But, I can do what
you preprogrammed above with a variable.
4 step checklist for coding a loop

1. DECLARE (which means set up) the loop control variable, giving it a name.

2. ASSIGN with = (which means COPY) the initial value that the loop variable should have.

3. Code a while statement's HEAD and BODY so you write as the HEAD while( TEST ) a TEST for the computer to do each time, on the loop variable

4. Code inside the BODY both

   (4a) what to repeat e.g. forward(); turn();

   (4b) to CHANGE the loop variable's value e.g. numberOfMoreTimes=numberOfMoreTimes–1;


iClicker What if you omit step 4B?

```java
int numberOfMoreTimes;
numberOfMoreTimes = 4;
while ( numberOfMoreTimes > 0 );
{
    this.forward(sizeParam);
    this.turn(72/2);
    numberOfMoreTimes=numberOfMoreTimes-1;
    Subtract 1 from the memorized count.
}

A) It still does 4 lines, turns and stops.
B) It crashes.
C) It keeps doing lines and turns until you click DrJava's RESET button.
D) It does nothing.  E) Nothing, and gets stuck.
```
Lab5 Followup

- Extend Picture to EditablePicture by adding the changeXYLocationsColor method.
- USE that method in an app.
- Start of Lab6: Add a method that changes the color of ALL Pixels in a horizontal line segment.
  - Plan that method
  - Code that method
- USE that method is an app.
public class EditablePicture extends Picture
{
    public EditablePicture(String filenameParam)
    {
        super(filenameParam);
    }

    public void changeXYLocationsColor(int xParam, int yParam,
                                       java.awt.Color cParam)
    {
        //Write the one line of code to achieve each purpose already written.
        //Purpose: Set up a Pixel reference variable with a declaration.
        Pixel pixToChangeRef;
        //Purpose: Call the getPixel method to get a reference to the
        //Pixel to change, and assign or copy that reference into a variable.
        pixToChangeRef = this.getPixel(xParam, yParam);
        //Purpose: Set the Color of the Pixel to change
        //to the value of the Color parameter
        pixToChangeRef.setColor(cParam);
        //Purpose: Return control to the spot where the method call was.
        //(Written for you.)
        return;
    }
}
public class EditablePicture extends Picture
{
    public EditablePicture(String filenameParam) 
    { super(filenameParam); }

    public void changeHLinesColor(int xParam, int yParam, int lenParam, java.awt.Color cParam) 
    {
        //Purpose: Set up a Pixel reference variable with a declaration.
        Pixel pixToChangeRef;
        //P: Set up a loop control variable
        int count;
        //P: count of Pixels actually changed is 0 initially
        count = 0;
        //P: make the loop repeat until lenParam Pixels are changed
        while (count < lenParam)
        {
            //P: MARK THE TOP (like a garbage can cover) of the body
            //P: Set up a Pixel ref variable.
            Pixel pRef;
            //P: Copy a reference to the Pixel to change according to plan.
            pRef = this.getPixel(xParam + count, yParam);
            //P: Actually change the color.
            pRef.setColor(cParam);
            //P: Make the value of count be correct again.
            count = count + 1;
        }
        //P: CAREFULLY MARK THE BOTTOM OF THE LOOP BODY!
        //P: Return control to the spot where the method call was.
        return;
    }
}
public class Lab6App
{
    public static void main(String[] a)
    {
        FileChooser.pickMediaPath();
        String digPhotoFileName;
        digPhotoFileName = FileChooser.pickAFile();
        EditablePicture myP;
        myP = new EditablePicture( digPhotoFileName );
        myP.explore( );
        java.awt.Color cRef;
        cRef = ColorChooser.pickAColor();
        System.out.println("You just picked Color " + cRef );
        System.out.println("Please magnify your picture in the explorer window."");
        System.out.println("Then, decide where to start the line and how long it should be." );
        System.out.println("Type in x y coordinates and then length in that order, separated by spaces." );
        java.util.Scanner sc = new java.util.Scanner(System.in);
        int xIn, yIn, lIn;
        xIn = sc.nextInt();
        yIn = sc.nextInt();
        lIn = sc.nextInt();
        myP.changeHLinesColor( xIn, yIn, lIn, cRef );
        myP.explore();
    }
}
Memory in General if there is time...
4 applications of variables

- A parameter variable stores, holds, remembers a parameter value. (eg. `sizeParamVar`)

- A variable stores, holds, remembers some data so the method can use that data later. (eg. `xOrig, yOrig`)

- A variable stores, holds, remembers a reference to an object (where an object is inside the computer) so methods can be called ON that PARTICULAR OBJECT

  (eg. `tref` and `pixRef`)

- A loop control variable controls when a loop should stop and might repeatedly provide a useful numbers, like `count` in

  `this.getPixel(xParam+count, yParam);`

  (where `this` refers to the G&E Picture you are editing.)