Begin by reading about the four applications of variables from last week's lecture and answering the first 3 lab questions on the sheet. Then do the programming asked for the remaining lab questions.

4 applications of variables

1) A parameter variable stores, holds, remembers a parameter value. (eg. `sizeParam`)  

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

3) 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`)  

4) 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.)

Here is the dome method from last week's lecture.

```java
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); //思想: Total amount of turn is 180 degrees.
    int numOfMore;
    numOfMore = 4;
    while (numOfMore > 0)  
    {
      this.forward(sizeParam);//Draw lines 1-4
      this.turn(72/2);//Turn angles 1-4.
      numOfMore = numOfMore - 1; //Count one more line
      //and angle.
    }
    this.forward(sizeParam);//Draw fifth line
    this.turn(18);//Turn fifth angle.
    //Thought: Total amount of turn is 180 degrees.
    this.moveTo(xOrig, yOrig);//Move to REMEMBERED position.
    return; //Return to the code of the caller.
  }
}
```

Lab Question 1:

`wref` is a parameter variable and it also stores a reference to a `World` type object. Therefore, `wref` is an example of applications 1) and 3) of variables.

`sizeParam` is an example of application______1)______

You fill in the application number (1, 2, 3 or 4) for what the other variables `xOrig` and `yOrig` are examples of______________

Variable `numOfMore` is an example of ____________
The while statement commands the simplest looping operation in Java, so we teach it first. The slides below help explain that.

It would be better if we could say: “Hey, computer! Please repeat the block of code below 4 times:”

```java
int numberOfMoreTimes;
    Set up a variable is a step we can do.
numberOfMoreTimes = 4;
    Copy 4 into that variable is another such step.
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( 72/2 );
    numberOfMoreTimes=numberOfMoreTimes–1;
    Subtract 1 from the memorized count.
    }
```

There is no Java operation that commands this directly; so we need to program it in terms of simpler steps.

Here (above) is how you CAN program that repetition in terms of four "little things" that Java lets you program: (1) Set up a variable with a name. (2) Copy a value into it. (3) Test and tell whether to repeat. (4) Subtract 1 from a variable's value.

Here is the completed changeXYLocationsColor method from last week's lab:

```java
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;
    }
}
```

Lab Question 2: What number 1), 2), 3) or 4) application is each variable an example of?

`xParam` and `yParam`_____________ `pixToChangeRef`_____________
`cParam` ______________and__________
Once you know how to program G&E's API for changing the color of a **Pixel** at one location (which the solution of last week's lab shows), and you know how to program loops, it is easy to program a method that uses a loop to change the color of a horizontal line of **Pixels**. (Programmers need to exercise judgment in choosing what parameters a method should have. Here, that is exactly how the parameters will determine exactly which horizontal line to re-color.) Here is the solution:

```java
public class EditablePicture extends Picture
{
    public EditablePicture(String filenameParam )
    {
        super(filenameParam);
    }
    public void changeAHorizLinesColor(int xParam, int yParam, int length
            java.awt.Color cParam )
    {
        //Use a loop to repeat re-coloring Pixels length times.
        int n;  //Set up the loop variable for how many Pixels we finished coloring.
        n = 0;  //We didn't color any yet, so zero is “how many”.
        while( n < length )  //Repeat coloring one when number colored < num. we want.
        {
            //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 + n, yParam);
            //Purpose: Set the Color of the Pixel to change
            //to the value of the Color parameter.
            pixToChangeRef.setColor( cParam );
            n = n + 1;  //Count the Pixel we just colored.
        }
        //Purpose: Return control to the spot where the method call was.
        return;
    }
}
```

**Lab Question 3:** What number 1), 2), 3) or 4) application is each variable an example of?

- xParam, yParam and length________________________
- pixToChangeRef________________________
- cParam __________and__________
- n_____________

**QUESTION 4 Colored horizontal line demonstration**

- After adding the `changeAHorizLinesColor` to the **EditablePicture** class TEST AND DEMONSTRATE your work. Here is how to do that:
- Edit the `main` method of the application (**Lab5App.java**) so that it (1) asks you for (with `System.out.println("...");`) and then (2) gets from you (the user, when the program runs, which `sc.nextInt()` three times) three integers: Two for an x-y location and the third for a length.
- It calls the `changeAHorizLinesColor` on the **EditablePicture** (through `pRef` just like last week), with four parameter values. The first three parameter values are of course given

```java
public class EditablePicture extends Picture
```
by the variables that hold the three int numbers that you had typed in. The fourth is the color that was picked by the ColorChooser.pickAColor( ) method.

- Make sure the explore( ) method is called on the EditablePicture AFTER the method changeAHorizLinesColor is called. (On some people's computer setups outside of lab, explore( ) fails. If that's so for you, use show( ) in place of explore( ).)
- Like you did for the changeXYLocationsColor method the spot last week: Note again where your nose is. Make the differently colored line start about an inch left of your face, go through your nose, and come out about symmetrically on the other side of your face.

**QUESTION 5 Colored vertical line method and demonstration**

- Method name: changeAVerticalLinesColor
- Parameters and types: int x, int ymin, int length and java.awt.Color color
- It should make a given recolored vertical line of width 1 (pixel).
- Parameter value effects: The topmost pixel to be colored should have location (coordinates) x and ymin. The number of pixels to be made purple should be length. In other words, length, a number of pixels at (x, ymin), (x, ymin+1), (x, y+2) up to (x, ymin+length-1) should get their color changed to the parameter color. Again, You MUST do it in a LOOP!
- How to do it is similar to the horizontal line question. Tip for programming speed: Copy and paste your code, and then edit it to make the line vertical instead of horizontal. You MUST change the names of the parameter variables for the sake of clarity to a human reader!
- TEST AND DEMONSTRATE: Make the vertical line be visible according to your input.

**QUESTION 6 Thick colored horizontal line and demonstration**

- Method name: changeAThickHorizLinesColor
- Parameters and types: int xmin, int y, int length and java.awt.Color color
- It should recolor a horizontal line of width 3 (pixels).
- Parameter value effects: The top and left most pixel to be recolored should have location (coordinates) (xmin and y-1). The bottom and right most pixel should have location (xmin+length-1, y+1). The number of pixels to be recolored should be length*3.
  
  In other words, the thick line is composed of three thin lines of the parametrized length. The first thin line should begin at (xmin, y-1), the second at (xmin, y) and the third at (xmin, y+1). Tip: Draw your own sketch to figure out what all those words mean.
- How: Write the method definition with the 3 parameter variable declarations and begin and end the body. Now you have a choice for how to do it: (Idea: you and a neighbor decide so you do it one way and your neighbor do it the other. Then compare your answers.)
  - Way 1: Use a loop again. Do it fast: First, copy your code from changeAHorizLinesColor. Add four code lines into the body of the loop. Put them near your original code to recolor one Pixel. Two of them should color the pixel right above that one pixel. The other two should color the pixel right below that one pixel.
● Way 2: Code the body of `changeA ThickHorizLinesColor` with NO LOOP AT ALL. Simply code `this.changeAHorizLinesColor( ?? , ?? , ?? );` THREE separate times with suitable expressions for the parameter values in each of the three calls.

● TEST AND DEMONSTRATE: Make the thick horizontal line be visible anywhere you like.
● TEST AND MAKE A NOTE: First, make sure your software works properly when you specify a thick line that is properly contained in the Picture. Do this: Type in 0 for the y location value. MAKE A NOTE OF WHAT HAPPENS__________________________ Figure out why.

**Lab6 Followup Task: Chromakey controlled horizontal line.**

● Method name: `changeHLineOnGreenOnly`

● Parameters and types: `int xmin, int y, int length and java.awt.Color color`

● It should make a recolored horizontal line of width 1 (pixel). However, it should recolor ONLY pixels that are green, leaving the others alone. The effect will be that the line drawn through your face will appear ONLY BEHIND YOU.

Study G&E’s programs 45 and 46 for clues to how to use an if statement to make the decision to color a Pixel purple or just skip it. Those clues tell how to code getting the red, green and blue intensity values from a Pixel, and make the computer add and compare them to make a decision about whether a Pixel is really green or not. You will have to interchange the roles of green and blue, since Prof. Guzdial’s screen was blue. Find those programs on pages 489-491 of the UA custom edition or pages 203-205 in the red covered paper edition. (Ask a friend, TA or the prof to show you that if you don't have the G&E material handy.)

● Parameter value effects and how: Just like in `changeAHorizLinesColor` except the recoloring instruction line is enclosed within the BODY (between the `{ ... }`) of an `if`-statement.

● TEST AND DEMONSTRATE: Note again where your nose is. Make the line start about an inch left of your face, go through your nose, and come out about symmetrically on the other side of your face. BUT..the line should appear BEHIND your head!

● For LAB CREDIT, upload your two Java files and two class files to Blackboard under Lab06 followup.

For the future: (1) Put `import java.awt.Color;` at the top of your Java files, so you can briefly type `Color` instead of `java.awt.Color`. (2) Develop a method that has a `Color` parameter plus parameters that determine a rectangle (with horizontal and vertical sides). The method should color in the given rectangle with the given color. Super challenge: Remove the horizontal and vertical restriction. Tip: Look up line rasterization on the web, and/or

`http://groups.csail.mit.edu/graphics/classes/6.837/F02/lectures/6.837-7_Line.pdf`
What you have to know to make loops that do what you want:

YOU must MEMORIZE (1), to set up a loop control variable
\[
\text{int numberOfMoreTimes; The computer can an do what we coded: Set up a variable.}
\]
(Of course, any other legal name is fine too, not just numberOfMoreTimes.)

YOU must MEMORIZE (2), to copy the right initial value into your loop control variable.
After \text{int numberOfMoreTimes;} you must code
\[
\text{numberOfMoreTimes = 4;}
\]
That is the ASSIGNMENT STATEMENT that, when the code runs,
\text{COPIES the literal value 4 into the variable named numberOfMoreTimes}

YOU must MEMORIZE (3), to code the test for when to repeat, and begin the block of what to repeat.
After \text{int numberOfMoreTimes;} \text{ numberOfMoreTimes = 4;}
you must write a WHILE TEST that, when the code runs,
(a) Tests whether the value of \text{numberOfMoreTimes} is strictly > 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.
\[
\text{while( numberOfMoreTimes > 0 )}
\]
\text{Tell whether the count of more times is zero or not.}
\text{When not, repeat the body below and tell again...}
\{
\text{this.forward(sizeParam);} \\
\text{this.turn(72/2);} \\
\text{numberOfMoreTimes = numberOfMoreTimes - 1;}
\text{Subtract 1 from the memorized count.}
\}

Finally, YOU must MEMORIZE (4) how to pre-program so when the loop runs, the computer will subtract 1 from the memorized count after each repetition of the desired two statements. Item (4) to memorize is underlined above and is called the \text{increment operation}.

Here is a 4-part checklist for programming a loop:
1. DECLARE (which means set up) the \text{loop control variable}, giving it a name.
2. ASSIGN with = (which means COPY) the initial value that the \text{loop variable} should have.
3. Code a while statement’s HEAD and BODY so you write as the HEAD is
\text{while( TEST )}
Here is a \text{TEST} for the computer to do each time on the \text{loop variable}
4. Code inside the BODY both
   (4a) what to repeat e.g. \text{forward(); turn();}
   (4b) to CHANGE the \text{loop variable’s value}
\text{e.g. numberOfMoreTimes = numberOfMoreTimes - 1;}