2.5 ARRAYS AND LOOPS

Learning Objectives

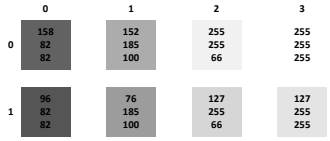
Learn what loops are, and some conventions for writing loops such as using ++ and declaring variables outside the loop

Learn different loops constructs and purposes

Changing the Red in a Picture

- One way to change a picture is to reduce the amount of red in it
  - What if we want to decrease it by half?
    - If we have a value of 200 what should the new value be?
    - How do we reduce any value by half?
  - What if we want to increase it by 25%?
    - If we have a value of 100 what should the new value be?
    - How do we increase any value by 25%?

Example: Decreasing Red



- Each column of the 2D array depicts a color and the same color with its red value reduced by half.
- The result might not seem intuitive -- the pixels don't turn from red to pink, but rather from red to greenish blue.
  - This is because, as we decrease the amount of red in a pixel, the overall ratio of green and blue to red goes up
  - Remember, in RGB, all colors are specified using a combination of red, green, and blue values).

Example: Decreasing Red

- There are length x height pixels in any picture
- Do we really want to write the code to change each one of these?
  - Get the current pixel
  - Get the red value of the current pixel
  - Change the red value to an appropriate percentage of the original value
  - Put the new red value in the current pixel
- Note that every color in a pixel has a maximum (255) and a minimum (0). The R(red), G(green), or B(blue) color of a pixel cannot be increased beyond its maximum (255) or decreased below its minimum (0).

- Changing one or two pixels should not be too much of a problem. However, if we want to manipulate the *entire* image, it could take some time.
- Remember how many lines of code it took to draw a line on your image? Now multiply that by four, and by the number of pixels in your image!
- For example, if there are 172800 pixels in the image, you would have to write 691,200 lines of code to finish this task. If we were to do this for an entire image by hand, it would be a very very long program. We need a smarter way of doing this.

We Need a Loop (Iteration)

- A way to execute a series of statements is in the body of a loop
  - With something changing each time the statements are executed
    - Ex. Different pixel to change
  - And some way to tell when we are done with the repetition
    - Some test to see if the loop should stop
      - Done with all of the items in an array

Loops

- It is critical when writing a loop that we have a way to test whether we are done repeating things.
  - In this case, we are done when we have looped through all of the pixels.
- This test is called the end condition, and it takes the form of a test to see if the loop should stop.
- A loop is used to do something a fixed number of times.
  - In this case, a loop will use a counter to keep track of when it is finished. Each time the loop repeats, the counter is incremented. When the counter reaches the goal number, the loop stops.

Example: A simple loop in pseudocode

- Suppose you are trying to use a robot to beat the sit-up world record, which stands at 133,986 consecutive situps.
- You may want to use a loop to do this, since you will have the robot complete many repetitions of the situp command.

Example: A simple loop in pseudocode

- Remember a loop has three main components:
  - A command that is executed;
  - an end condition which tells when the loop should stop
  - a counter.
- What we need is:

```
// initialize our counter to 0 since no situps have been done yet
int numSitups = 0;

// repeat this until numSitups > 133,986
//do a situp
System.out.println("The robot sat up!");

// don't forget to increment the counter after each situp
// or the loop will go on forever!
 numSitups = numSitups + 1;
```

Note:

- Equivalent ways to increment the counter variable numSitups :
  - numSitups = numSitups + 1;
  - numSitups += 1;
  - numSitups++;
- What += does is to increment the value of the variable and reassign it to the variable.
- Similarly, numSitups -= 1; and numSitpus--; are equivalent to numSitups = numSitups - 1; All three statements decrement the value of the variable.
- Another name for ++ is the increment operator and another name for -- is the decrement operator.

While Loops

1. We create a counter and set it to the start value.
2. Then we check that the counter is less than the stop value.
   - If it is less, then execute the statements in the loop.
3. We next add one to the counter and go back to check that the counter is less than the stop value.

The while Statement

- The while statement has the following syntax:

```
while ( condition ) {
   statement;
}
```

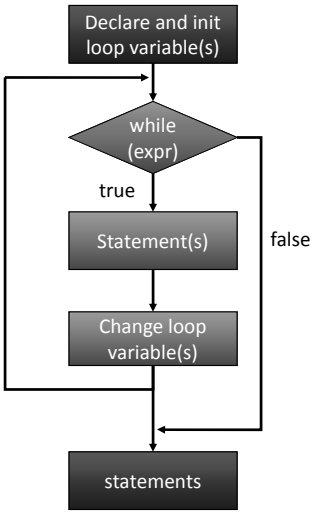
while is a reserved word

If the condition is true, the statement is executed. Then the condition is evaluated again.

- The statement is executed repetitively until the condition becomes false.

13

Logic of a while loop



- Note:
  - if the condition of a while statement is false initially, the statement is never executed
  - Therefore, the body of a while loop will execute *zero or more times*

14

Example:

- We can take the idea from before and add the code necessary to create a while loop for sitting up.
- Since 133,986 is enough situps to fill many many pages of your output panel, let's try doing this for 133 situps.
- If you want to experiment with a larger number, feel free. You can always kill the process if you get tired of watching "The robot sat up!" scroll by.


```
int numSitups = 0;
while (numSitups <= 133) {
   System.out.println(numSitups + ". The robot sat up!");
   situp numSitups = numSitups + 1;
}
```

Infinite Loops

- The body of a while loop must eventually make the condition false
- If not, it is an *infinite loop*, which will execute until the user interrupts the program
- This is a common type of *logical error*
- You should always double check to ensure that your loops will terminate normally

17

Nested Loops

- The body of a loop can contain another loop
- Each time through the outer loop, the inner loop will go through its entire set of iterations.

18

Decrease the red value in pictures

- How do we decrease the red values in an image?
  1. Get the current pixel
  2. Get the red value of the current pixel
  3. Change the red value to an appropriate percentage of the original value
  4. Put the new red value in the current pixel

Decrease the red value in pictures

- We cannot do the looping until we have a way to store the image.
- First, we need to create a copy of ALL the pixels in the image into a Pixel array:

```
Pixel[] pixelArray = this.getPixels();
```

- We need then a couple more variables:
  1. To store the value of red at that pixel: `int value;`
  2. To hold the current Pixel we are looking at: `Pixel tmp;`
  3. To be our counter: `int counter = 0;`

Decrease the red value in pictures

- Now we can loop:

```
while (count < pixelArray.length) {
```

1. Get the current pixel
   `tmp = pixelArray[count];`
2. Get the red value of the current pixel
   `value = tmp.getRed();`
3. Change the red value
   `value = value / 2;`
4. Put the new red value in the current pixel
   `pixelArray[count].setRed(value);`
5. Increment the counter
   `count++;`

Decrease the red value in pictures

- We cannot forget to add the relevant code to the main class so that we can test our work:

```
mypic.show();
mypic.decreaseRed();
mypic.repaint();
```

The do..while Loop

- The do statement has the following syntax:

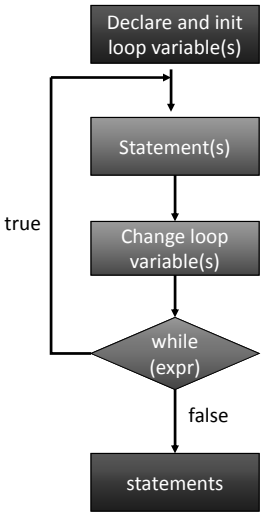
the do and while are reserved words

```
do {
    statement;
} while ( condition );
```

- The statement is executed once initially, then the condition is evaluated.
- The statement is repetitively executed until the condition becomes false

23

Logic of a do..while loop



- Note:
  - A do..while loop is similar to a while loop, except that the condition is evaluated after the body of the loop is executed
  - Therefore, the body of a do loop will execute *at least one time*

24

The for Loop

- The for statement has the following syntax:

Reserved word

The *initialization* portion is executed once before the loop begins

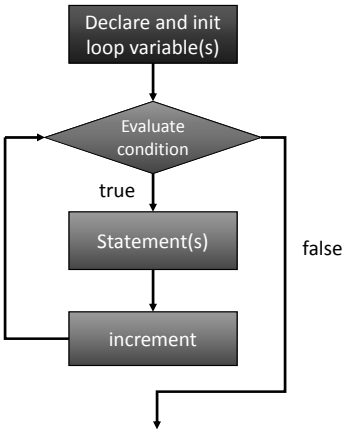
The statement is executed until the *condition* becomes false

```
for ( initialization ; condition ; increment ) {
   statement;
}
```

The *increment* portion is executed at the end of each iteration

25

Logic of a for loop



- Like a *while* loop, the condition of a *for* statement is tested prior to executing the loop body
- Therefore, the body of a for loop will execute zero or more times
- It is well suited for executing a specific number of times that can be determined in advance

26

The for Loop

- A *for* loop is equivalent to the following *while* loop structure:

```
initialization;
while ( condition ) {
   statement;
   increment;
}
```

27

The for Loop

- Each expression in the header of a *for* loop is optional
  - If the initialization is left out, no initialization is performed
  - If the condition is left out, it is always considered to be true, and therefore creates an infinite loop
  - If the increment is left out, no increment operation is performed
- Both semi-colons are always required in the for loop header

28

For-each Loop

- In Java if we want to do something to each item in an array
  - We can use the for-each loop (new since 1.5)

```
for (Type : variableName arrayName) {
    // body of the loop
}
```

  - Which means for each element in the array do the statements in the body of the loop

Method to Decrease Red

```
public void decreaseRed()   {
Pixel[] pixelArray = this.getPixels();
    int value = 0;

    // loop through all the pixels in the array
    for (Pixel pixelObj : pixelArray)    {
        // get the red value
        value = pixelObj.getRed();
        // decrease the red value by 50% (1/2)
        value = value / 2;
        // set the red value of the current pixel to the new value
        pixelObj.setRed(value);
    }
}
```

How This Works

- First we have the method declaration

  public void decreaseRed()

  - This is a public method that does not return anything and the name is decreaseRed. This method does not take any parameters since there is nothing in the ()
- Inside the body of the method, we declare an array of pixels and get them from the current Picture object

  Pixel[] pixelArray = this.getPixels();
- Then we declare a primitive variable to hold the current red value at the pixel

  int value = 0;

How This Works - Cont

- Next start the for-each loop

  for (Pixel pixelObj : pixelArray) {

  - Each time through the loop set the variable pixelObj to refer to the next Pixel object in the array of Pixel objects called pixelArray.
- Get the red value from the pixelObj and decrease the red

  value = pixelObj.getRed();

  value = value / 2;
- Set the red value at the pixelObj to the new value

  pixelObj.setRed(value);
- End the for-each loop body and the body of the method

Exercise

- Write a method increaseRed() to loop through all the pixels in a picture and double the red values
  - Multiply by 2
- To try this method do the following:

```
String fileName = FileChooser.pickAFile();
Picture pictObj = new Picture(fileName);
pictObj.increaseRed();
pictObj.repaint();
```


Summary

- A 2d array has columns and rows
- You can get a pixel at a particular x and y location
- You can use a for-each loop to execute a series of statements on each item in an array
- You can use a while loop to repeat a series of Java statements while some test is true
  - Often you will use a counter to get track of how many times the loop has executed
  - Declare the counter before the loop and increment it at the end of the loop