

+



Simple Data Visualization & Objects

+ Quiz 2

```

if (x < 100) {
  if (y < 10) {
    println("good job!");
  }
} else if (x < 50) {
  if (y > 10) {
    println("great job!");
  } else {
    println("what happened?");
  }
} else {
  if (y > 7) {
    println("not bad!");
  } else {
    println("nice try...");
  }
}
    
```

// variable declarations
int a = 2, b = 5;
float x = 2.0;
// expression
*b/a * x;*

+ Review

- **Array**
 - `int[] diameters = new int[10];`
 - `diameters[0], diameters[2], diameters[9]`
 - `diameters.length`

- Indexing starts at 0
- A way to have a collection of variables instead of individual ones

+ lab02 #1

```

double[] values = {0.6, 0.2, 0.3, 0.0, 0.5, 0.3, 0.7};
int limit = values.length/2;
for (int k=0; k<limit; k++) {
  double tmp = values[k];
  values[k] = values[values.length-k-1];
  values[values.length-k-1] = tmp;
}
println(values);
println(values[0]);
    
```

+ Built-in Array Functions

- `append(array, item)`
 - returns a new array expanded by one and add item to end
- `expand(array, newSize)`
 - returns a new array with size increased to newSize
- `shorten(array)`
 - returns a new array shortened by one
- `concat(array1, array2)`
 - returns a new array that is the concatenation of array1 and array2
- `subset(array, offset [, length])`
 - returns a subset of array starting at offset and proceeding for length (or end)
- `splice(array, value [, array2, index]) or`
 - returns a new array with value or array2 inserted at index
- `sort(array)`
 - returns a new array sorted numerically or alphabetically
- `reverse(array)`
 - returns a new array with all elements reversed in order

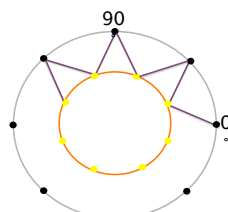
+ Recall

```

// convert from polar to cartesian coordinates
float x = centerX + cos(angleOffset + angle*i)*radius;
float y = centerY + sin(angleOffset + angle*i)*radius;

// draw a the point
vertex(x, y);
    
```

repeat for radius2 adding angle/2 to the expression inside sin and cos



+ Plots

- Line Charts
- Bar graphs
- Functions

+ Snowfall in Bryn Mawr, PA

Date	Snowfall in inches
Feb 8	0.2
Feb 9	0.3
Feb 10	0.4
Feb 11	0.0
Feb 12	0.2
Feb 13	0.0
Feb 14	0.0
Feb 15	1.0

+ Basic Visualization

- Given an array of data (values)

```
float[] snowfall = {
    0.20, 0.30, 0.40, 0.00, 0.20, 0.00, 0.00, 1.00
};
```

- How do we visualize?
 - plot each value?
 - connect the plotted points with lines (line chart)?
 - draw rectangles with each value as height (bar graph) *

+ Example Bar Chart

- snowViz

+ Let's plot $\sin(x)$ (with lines or points)

- Problem: Plot the graph of the function $y = \sin(x)$
- This is a continuous function. How do we plot each point?
- Let's start with plotting x and y for one value of x
- $x = \pi/4, y = \sin(x) = 0.707106781$
- Let's first draw our axes with the x and y axis centered on $\text{width}/2, \text{height}/2$

```
line(0, height/2, width, height/2);
line(width/2, 0, width/2, height);
```
- Where should $\pi/4$ (45 degrees) be?

+ Direct Example

- Set size of canvas
- define axis center locations
- define tick marks
- plot x and y axis
- plot a sample point at every pixel in the x axis from -360 degrees to 360 degrees

+ Indirect Example

- Sample the function to plot 720 points.
 - save x values in one array
 - save y values in another array
- Call a function void plot(float[] x, float[] y)
 - the plot function
 - creates x and y axes
 - plots each point in x and y using a for loop.

+ What is an Object?

- An **object** is an **instance** of a **class**.
- What is an **instance**?
 - An **instance** is a distinct example of the class that
 - is **in memory**
 - has specific **assignments** for the **variables declared by the class** it represents.
 - has functionality based on the class.
- What is a **class**?
 - A complex data type.
 - The design for objects of its type.

+ Defining Your Own Object with Classes

- Classes are blueprints or **prototypes** for new objects
- Classes define all **field** and **method declarations**
 - ... which are repeated for each new object created
- Classes **DO NOT** set the **data values** stored in fields
 - ... but they likely determine how
- Using a class to create a new object is called **instantiating** an object
 - ... creating a new object **instance** of the class
- Classes often model real-world items

+ Class vs. Object

