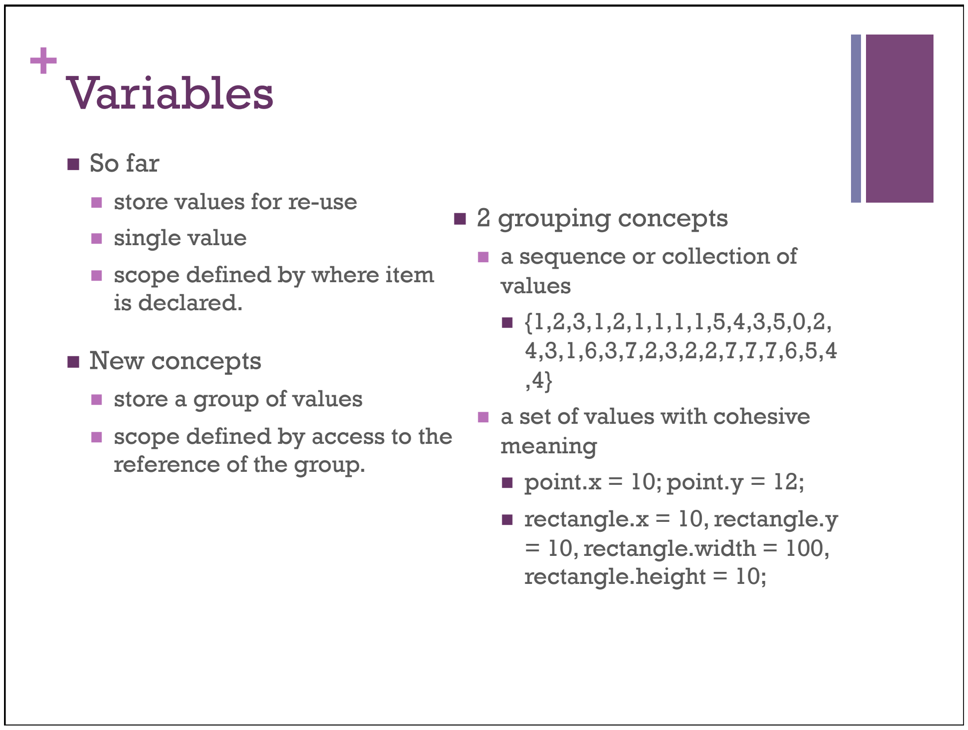
+

Arrays

+

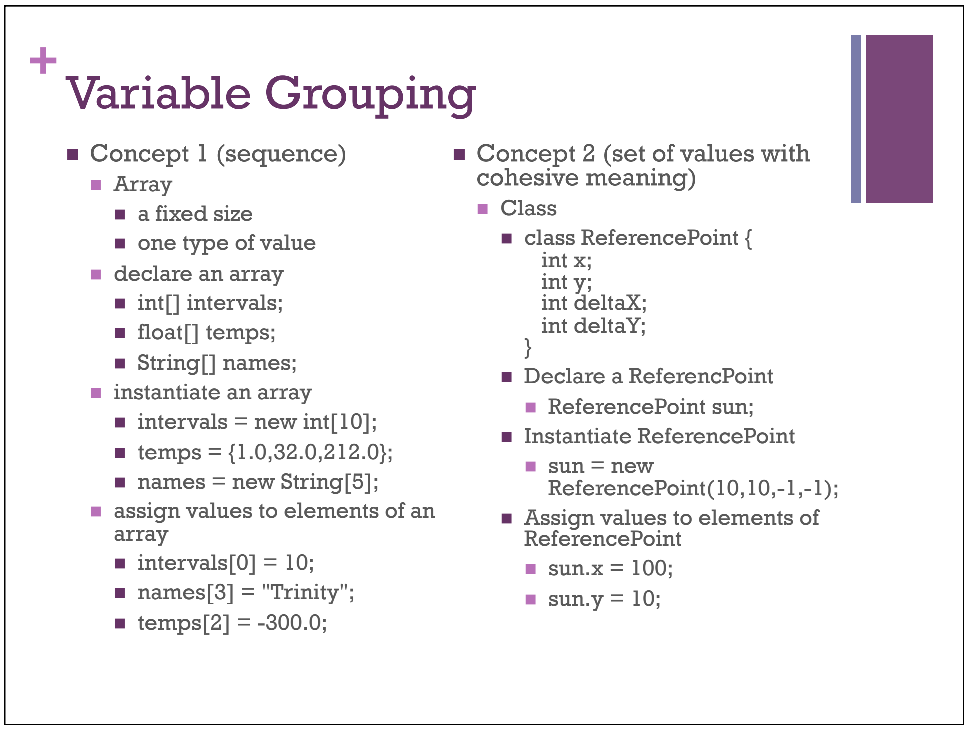
So far..

- A program consists of
  - actions:
    - call draw functions
      - line, rect, ellipse, etc.
    - change the drawing canvas
      - size, background, translate, rotate
    - do math
      - *,+,-,/,%,cos, etc.
    - Input
      - read
        - file
        - console
      - mouse
      - keyboard
- actions are done on:
  - literals
    - 1,2,3,'a',"hello",1.0,true, etc.
  - variables
    - int x;
    - String test;
    - etc.
- Actions happen sequentially unless
  - if(condition){}else if(condition){}else{}
  - switch(variable){ case value: … default: }
  - while(){}, for(){}, do{}while()
  - functionCall();



Variables

- So far
  - store values for re-use
  - single value
  - scope defined by where item is declared.
- New concepts
  - store a group of values
  - scope defined by access to the reference of the group.
- 2 grouping concepts
  - a sequence or collection of values
    - {1,2,3,1,2,1,1,1,1,5,4,3,5,0,2, 4,3,1,6,3,7,2,3,2,2,7,7,7,6,5,4 ,4}
  - a set of values with cohesive meaning
    - point.x = 10; point.y = 12;
    - rectangle.x = 10, rectangle.y = 10, rectangle.width = 100, rectangle.height = 10;



Variable Grouping

- Concept 1 (sequence)
  - Array
    - a fixed size
    - one type of value
  - declare an array
    - int[] intervals;
    - float[] temps;
    - String[] names;
  - instantiate an array
    - intervals = new int[10];
    - temps = {1.0,32.0,212.0};
    - names = new String[5];
  - assign values to elements of an array
    - intervals[0] = 10;
    - names[3] = "Trinity";
    - temps[2] = -300.0;
- Concept 2 (set of values with cohesive meaning)
  - Class
    - class ReferencePoint {

      ```
        int x;
        int y;
        int deltaX;
        int deltaY;
      }
      ```
    - Declare a ReferencPoint
      - ReferencePoint sun;
    - Instantiate ReferencePoint
      - sun = new ReferencePoint(10,10,-1,-1);
    - Assign values to elements of ReferencePoint
      - sun.x = 100;
      - sun.y = 10;

+ Arrays

- A special kind of variable that holds not one, by many data items of a given type.
- Declared like variables, only type is followed by a pair of brackets.

```
float[] xs;
```

- Can be initialized using a special syntax involving the `new` keyword, the type, and a *size* in brackets.

```
// Ten diameters
int[] diameters = new int[10];
```

+ Arrays

- Individual data items are accessed with an index and square brackets.
  - `diameters[0]`, `diameters[1]`, etc
  - **Indexes start at 0!**
- The length of an array can be determined using its `length` property.
  - `diameters.length`
  - The length of an array is one greater than the last valid index. (Because the first index is 0.)
- Arrays can be passed to, and returned from functions.

Arrays

- declare an array
  - `int[]    intervals;`
  - `float[]  temps;`
  - `String[] names;`
- instantiate an array

  -
    ```
    intervals =
       new int[10];
    ```

  -
    ```
    temps =
       {1.0, 32.0, 212.0};
    ```
  - `names = new String[5];`
- assign values to elements of an array
  - `intervals[0] = 10;`
  - `names[3] = "Trinity";`
  - `temps[2] = -300.0;`

  -
    ```
    int j    = 1;
    temps[j] = 98.6;
    ```
- get the length of an array

  -
    ```
    System.out.println(
    "There are " +
    temps.length
    + " temperatures.");
    ```

Example

- Problem: Create 10 circles each with a random diameter at random positions on the display. Move each circle 1 diameter towards the center of the display once per second.

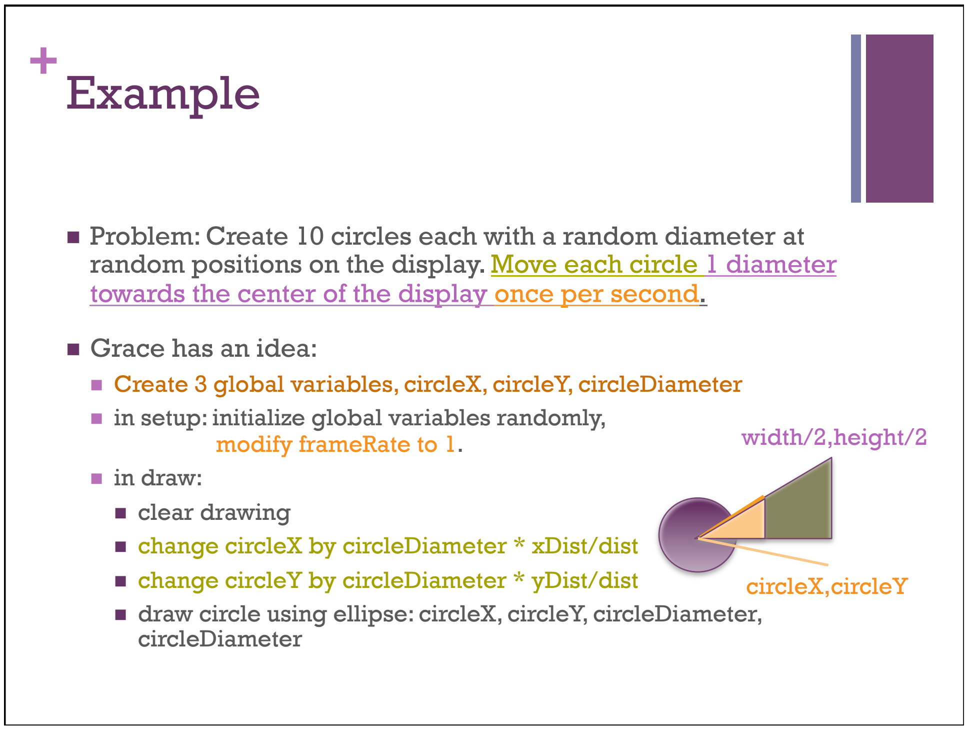
+ Example

- Problem: Create 10 circles each with a random diameter at random positions on the display. Move each circle 1 diameter towards the center of the display once per second.
- Ada has an idea:
  - loop 10 times
    - initialize a diameter, d, with a random value from 10 to 100
    - create a circle using ellipse() with
      - random x from 0 to width
      - random y from 0 to height
      - d width and d height

+ Example

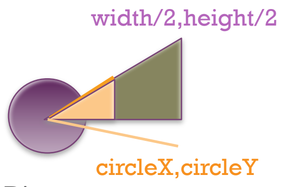
- Problem: Create 10 circles each with a random diameter at random positions on the display. Move each circle 1 diameter towards the center of the display once per second.
- Ada has an idea:
  - loop 10 times
    - initialize a diameter, d, with a random value from 10 to 100
    - create a circle using ellipse() with
      - random x from 0 to width
      - random y from 0 to height
      - d width and d height

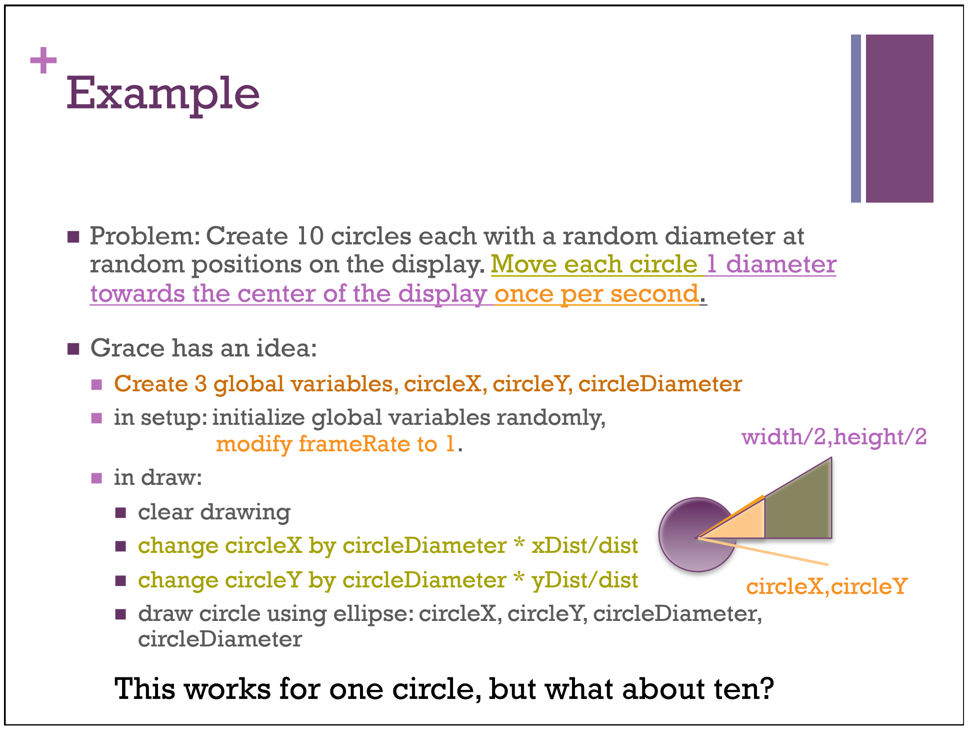
This works for the setup, but what about the second step?



+ Example

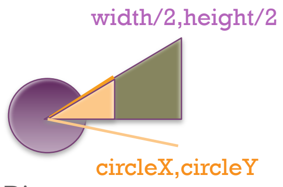
- Problem: Create 10 circles each with a random diameter at random positions on the display. Move each circle 1 diameter towards the center of the display once per second.
- Grace has an idea:
  - Create 3 global variables, circleX, circleY, circleDiameter
  - in setup: initialize global variables randomly, modify frameRate to 1.
  - in draw:
    - clear drawing
    - change circleX by circleDiameter * xDist/dist
    - change circleY by circleDiameter * yDist/dist
    - draw circle using ellipse: circleX, circleY, circleDiameter, circleDiameter



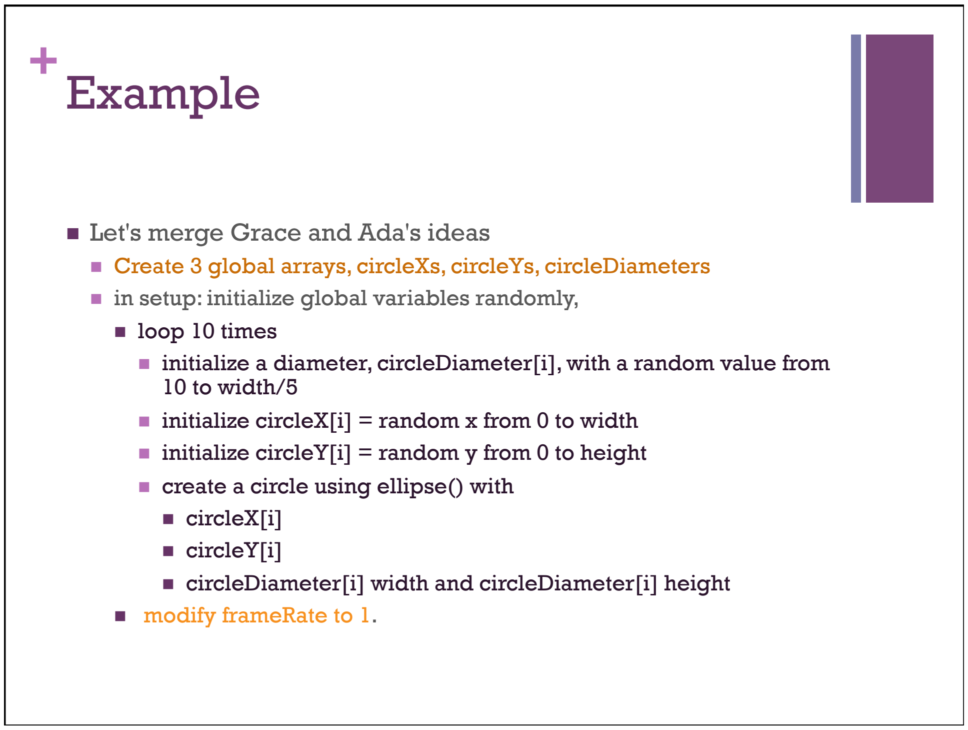


+ Example

- Problem: Create 10 circles each with a random diameter at random positions on the display. Move each circle 1 diameter towards the center of the display once per second.
- Grace has an idea:
  - Create 3 global variables, circleX, circleY, circleDiameter
  - in setup: initialize global variables randomly, modify frameRate to 1.
  - in draw:
    - clear drawing
    - change circleX by circleDiameter * xDist/dist
    - change circleY by circleDiameter * yDist/dist
    - draw circle using ellipse: circleX, circleY, circleDiameter, circleDiameter

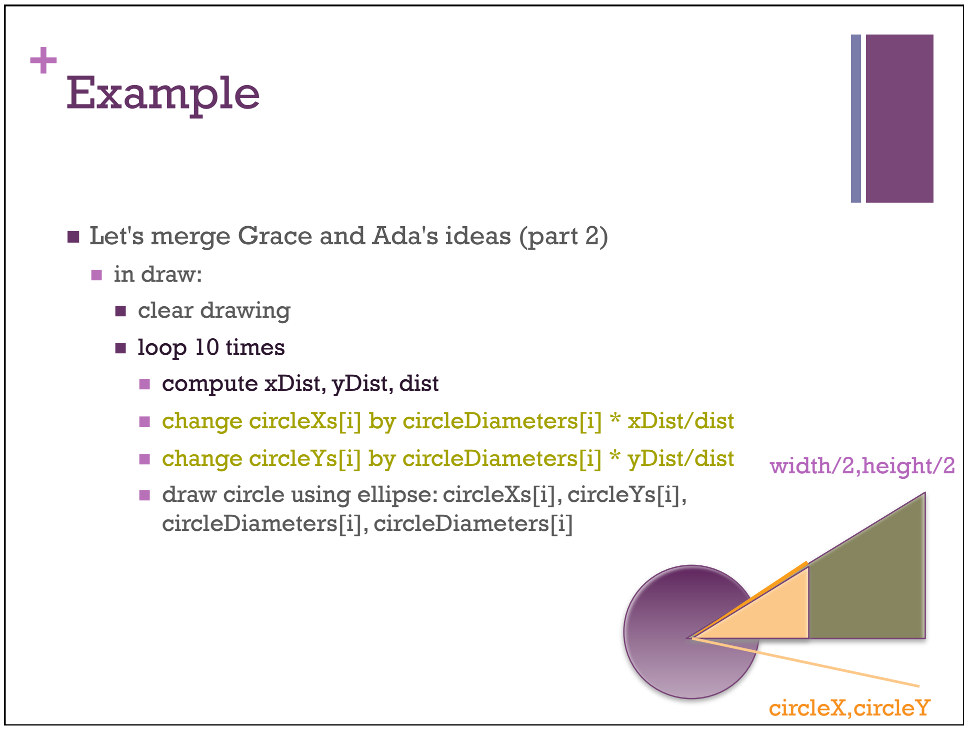


This works for one circle, but what about ten?



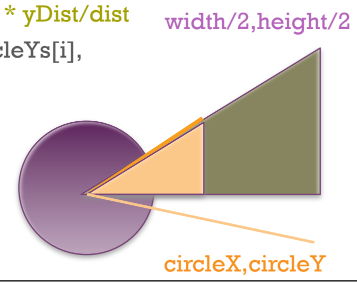
+ Example

- Let's merge Grace and Ada's ideas
  - Create 3 global arrays, circleXs, circleYs, circleDiameters
  - in setup: initialize global variables randomly,
    - loop 10 times
      - initialize a diameter, circleDiameter[i], with a random value from 10 to width/5
      - initialize circleX[i] = random x from 0 to width
      - initialize circleY[i] = random y from 0 to height
      - create a circle using ellipse() with
        - circleX[i]
        - circleY[i]
        - circleDiameter[i] width and circleDiameter[i] height
    - modify frameRate to 1.



+ Example

- Let's merge Grace and Ada's ideas (part 2)
  - in draw:
    - clear drawing
    - loop 10 times
      - compute xDist, yDist, dist
      - change circleXs[i] by circleDiameters[i] * xDist/dist
      - change circleYs[i] by circleDiameters[i] * yDist/dist
      - draw circle using ellipse: circleXs[i], circleYs[i], circleDiameters[i], circleDiameters[i]



+

```
int[] diameters = new int[10];
float[] circleXs  = new float[10];
float[] circleYs  = new float[10];
void setup() {
  size(displayWidth, displayHeight);
  background(200);
  // loop 10 times initializing values randomly
  for (int i=0; i<diameters.length; i++) {
    diameters[i] =  int(random(0, width/2));
    circleXs[i] = random(width);
    circleYs[i] = random(height);

    // draw initial circles.
    fill(255, 0, 0);
    ellipse(circleXs[i], circleYs[i],
            diameters[i], diameters[i]);
  }
  frameRate(1);
}
```

+

```
void draw() {
  background(200);
  for (int i = 0; i < diameters.length; i++) {
    // compute distances
    float xDist = width/2 – circleXs[i];
    float yDist = height/2 – circleYs[i];
    float dist = sqrt(xDist*xDist + yDist*yDist);
    // modify position by 1 diameter towards center
    circleXs[i] += diameters[i] * xDist/dist;
    circleYs[i] += diameters[i] * yDist/dist;
    // draw circle
    ellipse(circleXs[i], circleYs[i],
            diameters[i], diameters[i]);
  }
}
```

What is an Array?

- An array has 3 roles:
  - holds a group of values
  - a limited example of an object
  - Array variables are references, not values.
- Limited example of an object
  - must use new to instantiate it
  - has `length` as a field
  - When a variable represents an array it is always a reference.

Functions Informally (reminder)

- A function A function is like a subprogram, a small program inside of a program.
- The basic idea – we write a sequence of statements and then give that sequence a name. We can then execute this sequence at any time by referring to the name.
- Function definition: this is where you create a function and define exactly what it does
- Function call: when a function is used in a program, we say the function is *called*.
- A function can only be defined once, but can be called many times.

Function Examples

```
void setup() { … }
void draw() { … }


void line( float x1, float y1, float x2, float y2) { … }
… and other graphic functions


float float( … )
… and other type-conversion functions

… etc.
```

+ Functions

Modularity

- Functions allow the programmer to break down larger programs into smaller parts.
- Promotes organization and manageability.

Reuse

- Enables the reuse of code blocks from arbitrary locations in a program.

Function Parameters

- Parameters (arguments) can be "passed in" to function and used in body.
- Parameters are a comma-delimited set of variable declarations.
- Parameters act as input to a function.
- Passing parameters provides a mechanism to execute a function with many different sets of input
- We can call a function many times and get different results by changing its parameters.
  - rect(1,1,10,10); rect(40,40,103,405); rect(x,y,z w);
  - for(int i = 0; i < 100; i++) {

```
    rect(i *2, i *3,random(i,width),random(i,height);
}
```

What happens when we call a function?

- Execution of the calling program/function is suspended.
- The argument expressions are evaluated.
- The resulting values are copied, or references are passed, into the corresponding parameters.
- The statements in the function's body are executed in order.
- Execution of the calling program/function is resumed when a function exits (finishes).

Variable Scope

The part of the program from which a variable can be accessed.

Rules:

1. Variables declared in a block are only accessible within the block.
2. Variables declared in an outer block are accessible from an inner block.
3. Variables declared outside of any function are considered global (available to all functions).
4. Arrays and classes are passed by reference instead of copied

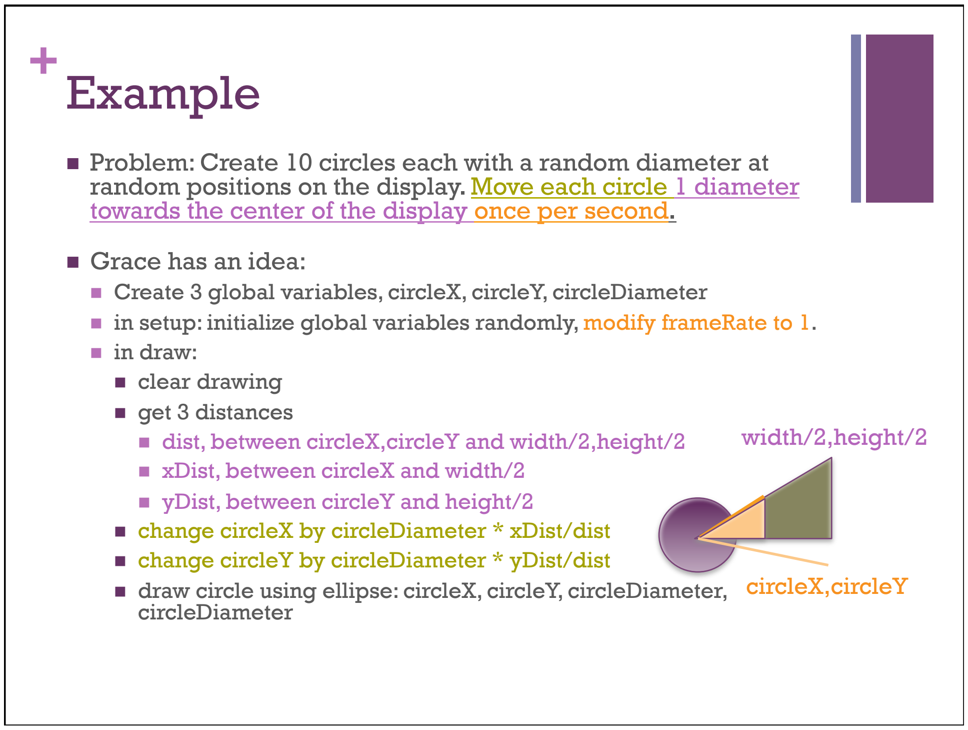
Variable Lifetime

- Variables cannot be referenced before they are declared.
- Variables can be declared in…
  - the global scope
  - the body of a function or constructor
  - the arguments of a function or constructor
  - a statement block (for, while, if, …).
- A variable is created and initialized when a program enters the block in which it is declared.
- A variable is destroyed when a program exists the block in which it was declared.

+
array functions

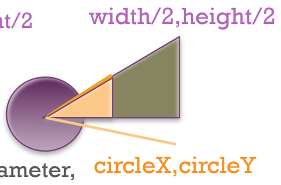
- Make a function `void zeros(int[] changeMe)` that takes an array of ints and sets all of its values to 0;
- Make a function `float[] zeros(int size)` that creates a float array of length `size`, sets all of the values to 0.0, and returns the array.

+



+ Example

- Problem: Create 10 circles each with a random diameter at random positions on the display. Move each circle 1 diameter towards the center of the display once per second.
- Grace has an idea:
  - Create 3 global variables, circleX, circleY, circleDiameter
  - in setup: initialize global variables randomly, modify frameRate to 1.
  - in draw:
    - clear drawing
    - get 3 distances
      - dist, between circleX,circleY and width/2,height/2
      - xDist, between circleX and width/2
      - yDist, between circleY and height/2
    - change circleX by circleDiameter * xDist/dist
    - change circleY by circleDiameter * yDist/dist
    - draw circle using ellipse: circleX, circleY, circleDiameter, circleDiameter


width/2,height/2
circleX,circleY