

COMP 1010- Summer 2015 (A01)

Jim (James) Young

young@cs.umanitoba.ca

jimyoung.ca

Hello!

James (Jim) Young

young@cs.umanitoba.ca

jimyoung.ca

office hours T / Th: 17:00 – 18:00

EITC-E2-582

(or by appointment, arrange by email)

Example: wandering star

What if we want the points to go back to their original spot if we click the mouse?

How can we save the original points??

Make another set of arrays

- instead of making the original points wander around, work with a copy – the original sticks
- When the mouse is pressed, copy the originals into the copy arrays

passing arrays to and from functions

pass arrays to/from function

note: array variable types can be used the same as any variable:

```
type[] someFunction(type[] somevariable){...
```

e.g., make a function that returns an int array of size n , with data from $1..n$.

```
int[] makeArray(int n) {...
```

Use it in your program, draw lines from the points (array[i], 0) to the mouse.

Make a function to draw the points

The header is as normal: take the array as a parameter:

```
void drawLines(int[] xValues)
```

Arrays, functions, and memory

What happens here? Is this okay?

```
int[] numbers = new int[1];  
void setup()  
{  
    numbers = makeArr(10);  
    numbers[9]++;  
}
```

```
int[] makeArr(int n) {  
    int [] data = new int[n];  
    for (int i = 0; i < data.length; i++) {  
        data[i] = i+1;  
    }  
    return data;  
}
```

Just to peek... modify to print out the memory addresses

```
int[] numbers = new int[1];
void setup()
{
  println(""+numbers);
  numbers = makeArr(10);
  println(""+numbers);
  numbers[9]++;
}
```

```
int[] makeArr(int n) {
  int [] data = new int[n];
  for (int i = 0; i < data.length; i++) {
    data[i] = i+1;
  }
  return data;
}
```

```
int[] numbers = new int[1];
```

```
void setup()
```

```
{
```

```
  println(""+numbers);
```

```
  numbers = makeArr(10);
```

```
  println(""+numbers);
```

```
  numbers[9]++;
```

```
}
```

```
int[] makeArr(int n) {
```

```
  int [] data = new int[n];
```

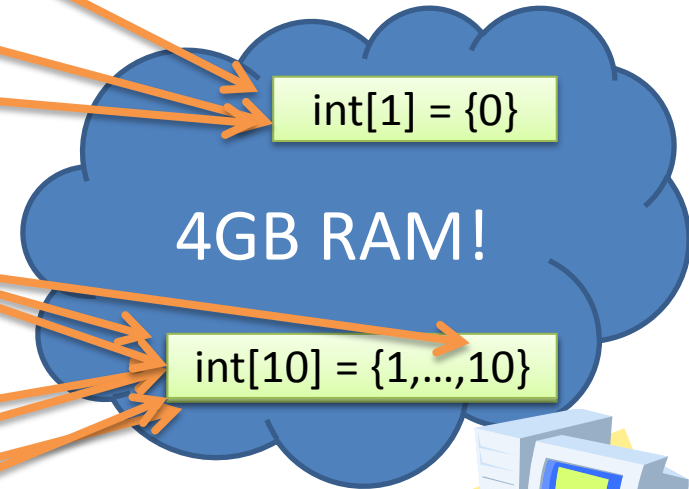
```
  for (int i = 0; i < data.length; i++) {
```

```
    data[i] = i+1;
```

```
  }
```

```
  return data;
```

```
}
```



example: generate first n Fibonacci numbers

remember: Fibonacci numbers are a sequence :

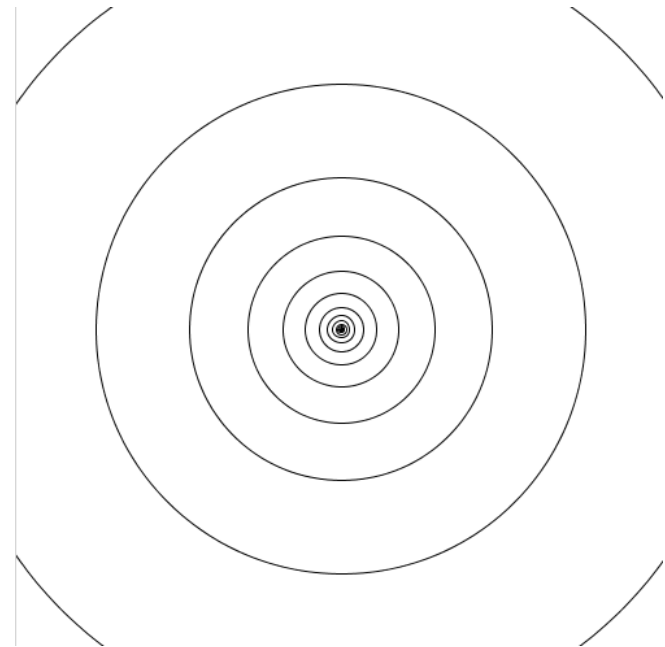
0, 1, 1, 2, 3, 5...

$$F_0 = 0,$$

$$F_1 = 1,$$

$$F_n = F_{n-1} + F_{n-2}$$

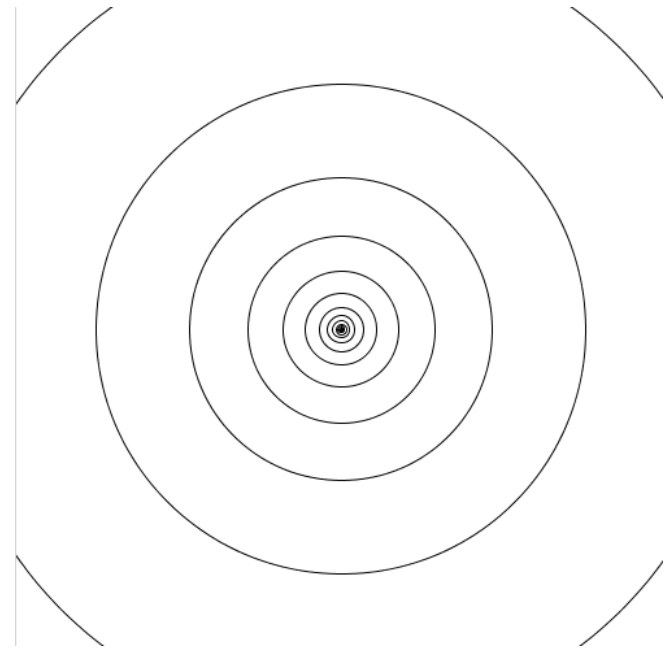
```
int[] fibonacciSequence(int n){
```



Visualize...

Draw circles with the radii as the Fibonacci numbers.

Put all the code in the setup
– not interactive



Questions..

```
int[] fib = fibonacci(COUNT);
```

Wait! I just created the variable and did not instantiate the array.. Is this okay?

changing values in a function....

consider:

```
void setup() {  
    int i = 15;  
    printInflation(i);  
    println(i);  
}
```

output:

16

15

The function does not alter the value of `int i` in `setup` because when `printInflation` is called, a copy of `i` is made

```
void printInflation(int number) {  
    number += 1;  
    println(number);  
}
```

changing **array** values in a function....

```
void setup() {  
    int i[] = {1, 2, 3};  
    printInflationArray(i);  
    println(i[0]);  
}
```

output:

2

2

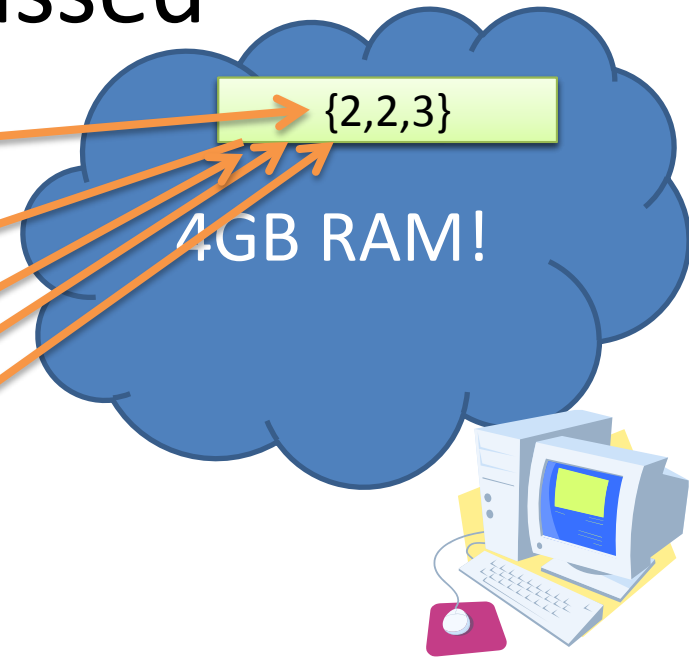
Why does this happen???

```
void printInflationArray(int[] intarray) {  
    intarray[0] += 1  
    println(intarray[0]);  
}
```


Function calls only pass the **reference** to the array, the **address**. The entire array is not copied and passed

```
void setup() {  
    int i[] = {1, 2, 3};  
    printInflationArray(i);  
    println(i[0]);  
}
```

```
void printInflationArray(int[] intarray) {  
    intarray[0] += 1  
    println(intarray[0]);  
}
```



consider...

```
void setup() {  
    int i[] = {1, 2, 3};  
    makeNewArray(i);  
    println(i[0]);  
}
```

```
void makeNewArray(int[] intarray) {  
    intarray = new int[3];  
    intarray[0] = 5;  
    println(intarray[0]);  
}
```

output:

5

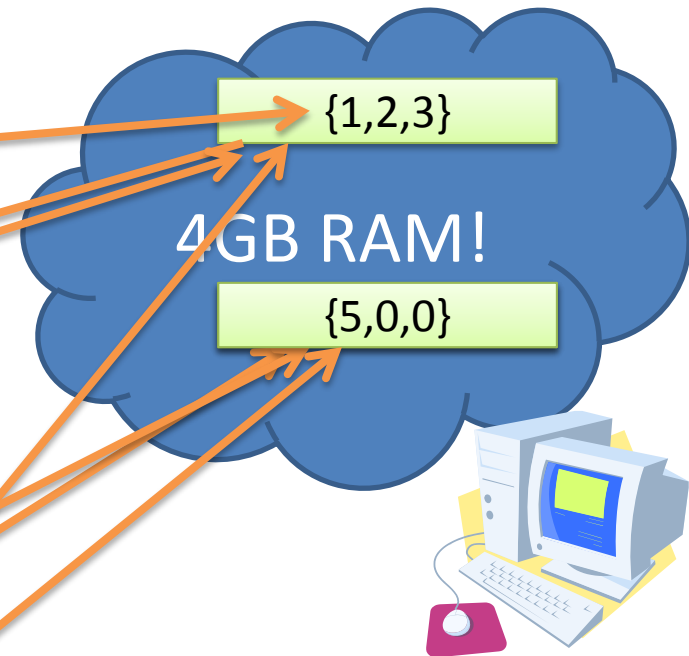
1

Why does this happen???
the old array didn't
change..

function passes the reference to the object. the original variable is unchanged

```
void setup() {  
    int i[] = {1, 2, 3};  
    makeNewArray(i);  
    println(i[0]);  
}
```

```
void makeNewArray(int[] intarray) {  
    intarray = new int[3];  
    intarray[0] = 5;  
    println(intarray[0]);  
}
```



Example::

In this example, we will

- create an array in a function, and return it
- modify an array in a function
- send multiple arrays to a function

Make a random collection of dots that you can steer with the keyboard

Create a function `newRandomArray` that takes an `n`, the number of bins, and `max`, the largest value

- creates the array
- fills with random values
- returns the array

Use it in `setup` to set globals for `x` and `y` points

Next..

Make a function `drawPoints` that takes `x`, and `y` arrays, and draws the points

Takes arrays and uses them

next

Make a function, `addToArray`, that takes an array and an integer value, and adds that value to every bin

This modifies the array in the function

In draw:

- if mouse pressed, new random arrays

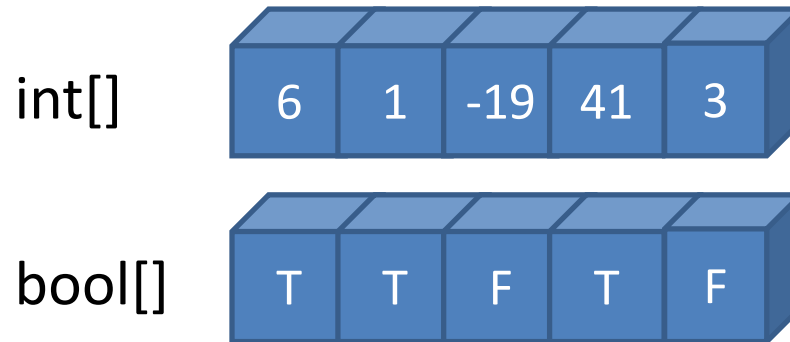
- if key pressed, check for u,d,l,r and move

Techniques with Arrays

partially filled arrays

partially filled arrays

note: when you first create an array, the default data is generally useless to you: you need to put data into the array

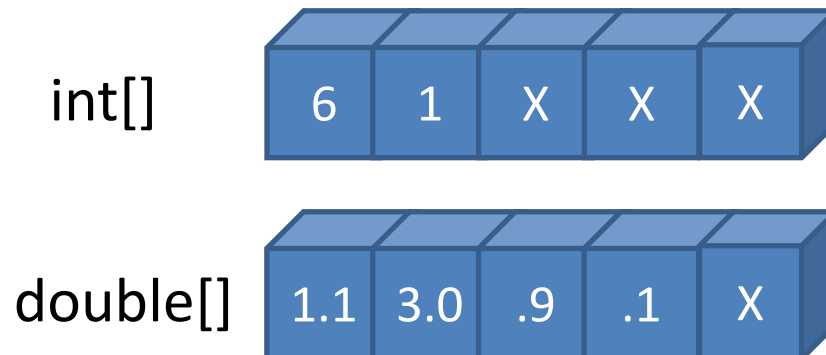


until now, all of our arrays were fully populated immediately

partially filled arrays:

note: a **partially filled array** is an array that has data in some bins but not in others.

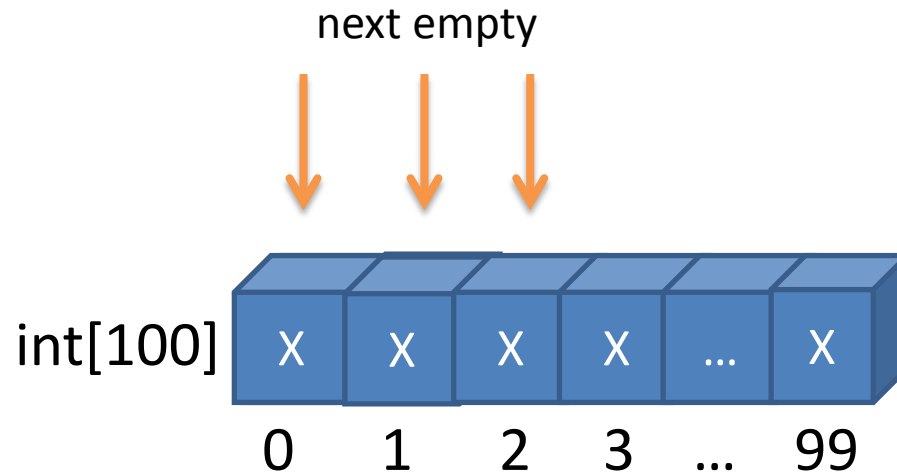
the challenge becomes identifying which bins have data, and which do not.



simple technique for filling up an array

go left to right,

keep a pointer to the next empty spot



use pointer each time to decide where to put the next element