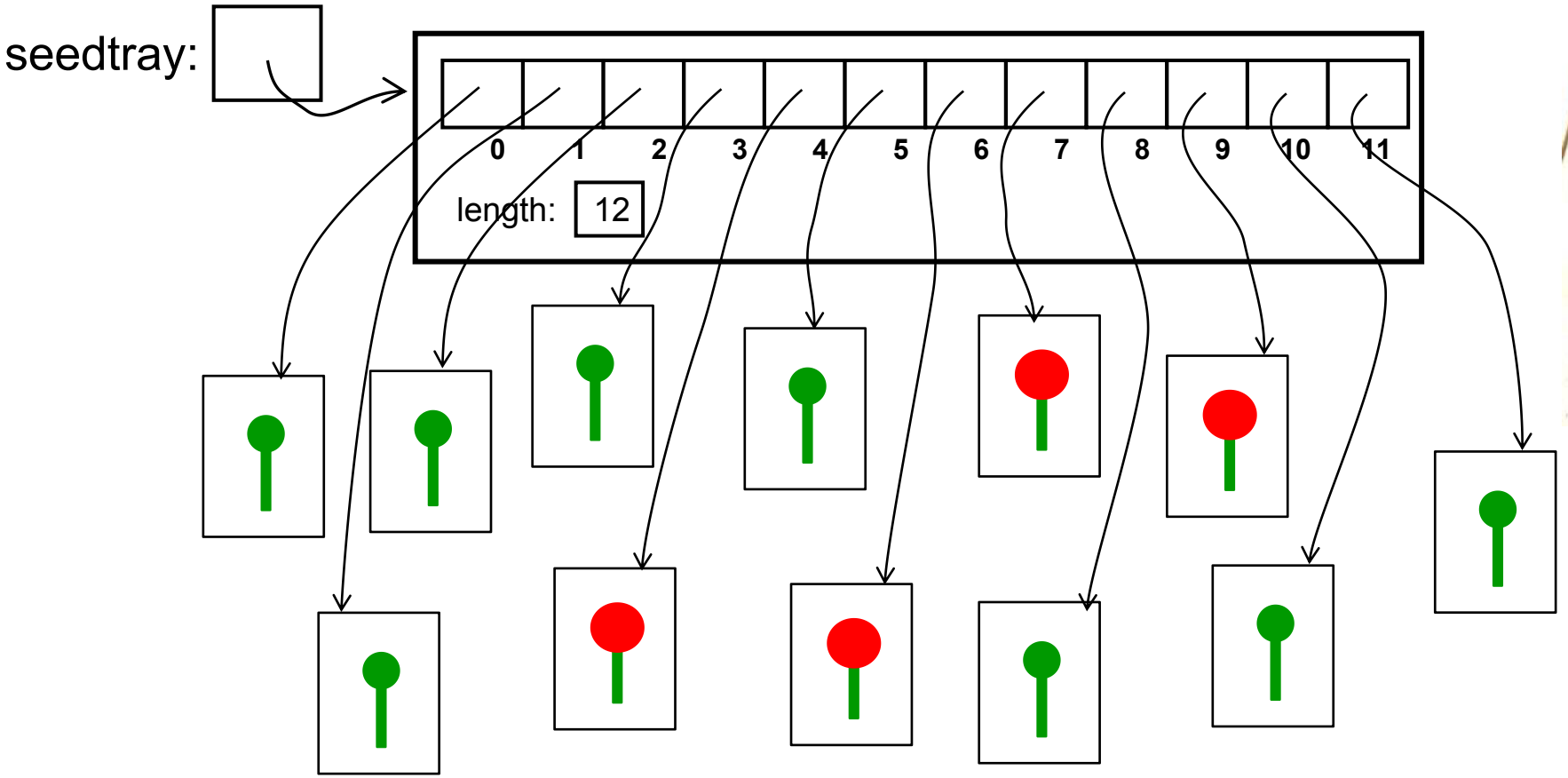
Arrays vs ArrayLists

- Some lists have a fixed number of places:

- The places may be empty

- Arrays may be sometimes more convenient than using ArrayLists

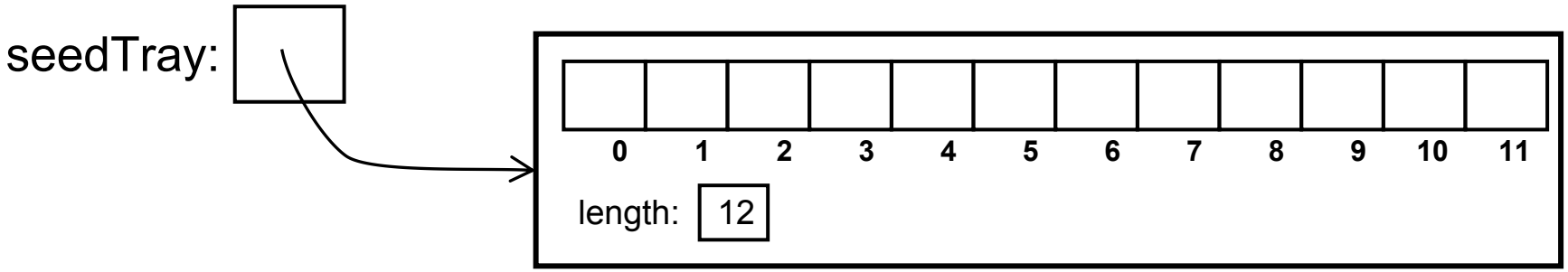
SeedTray Program: just 12 flowers



from http://bifurcatedcarrots.eu/wp-content/uploads/2010/04/tps_seedlings.jpg

Arrays

- An array is an object with a fixed number of places
  - Length determined when array is created
  - All elements are of the same type
  - Can have arrays of int, double, boolean
  - Special syntax, no methods



  - Each element specified by its *index* (an int expression)

    seedTray[4] ← the element in seedTray specified by index 4

    seedTray[n-3]

    - Counting from 0, just like ArrayLists!

  - Array knows its length: seedtray.length

Confusion:
names.size() ← ArrayList
name.length() ← String
tray.length ← Array

Declaring and Creating Arrays

- Declare a variable to hold an array object by putting [ ] after the type of the elements:

```
Flower[ ]  seedtray;
String[ ] keywords;
private double[ ] marks;
```

Creates a place that can hold an array
Doesn't create the array itself

- Create an array object with **new** and the length of the array in square brackets:

```
new Flower[12];
new String[50];
new double[200];
```

NO ROUND BRACKETS !!!

Creates an array object, but nothing in it

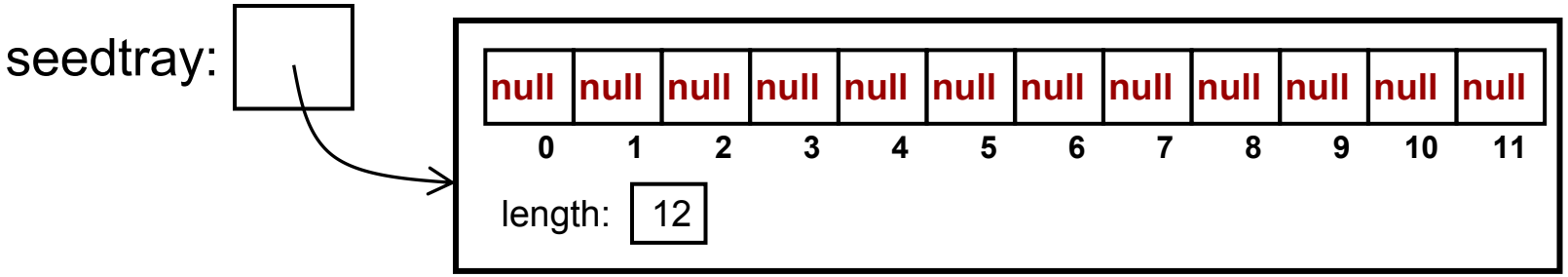
Can have an array of double or int (unlike ArrayLists)

- As usual, can combine declaration and initialisation:
  - `String [ ] keywords = new String [50];`

What does the new array contain?

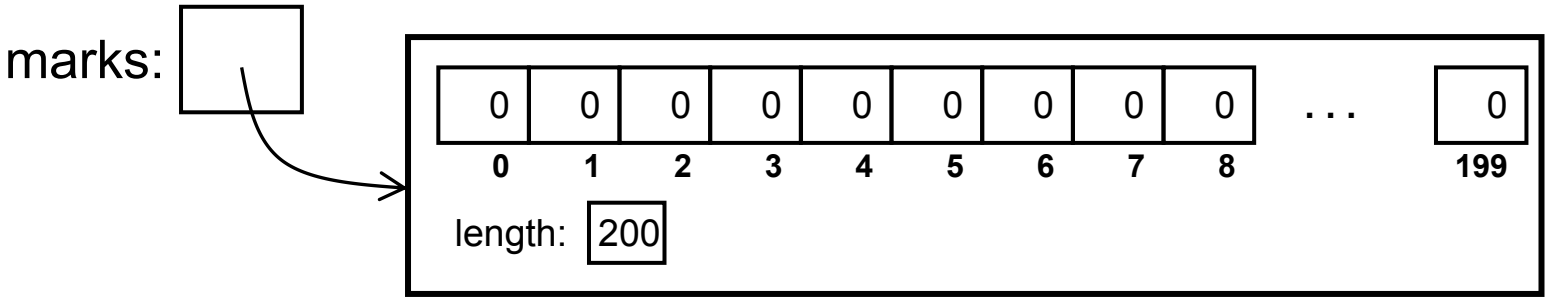
Initial values in a new array

```
Flower[ ]  seedtray =  new Flower[12];
```



Arrays of objects initialised with **null** (the "no object here" value)

```
double[ ]  marks =  new double[200];
```

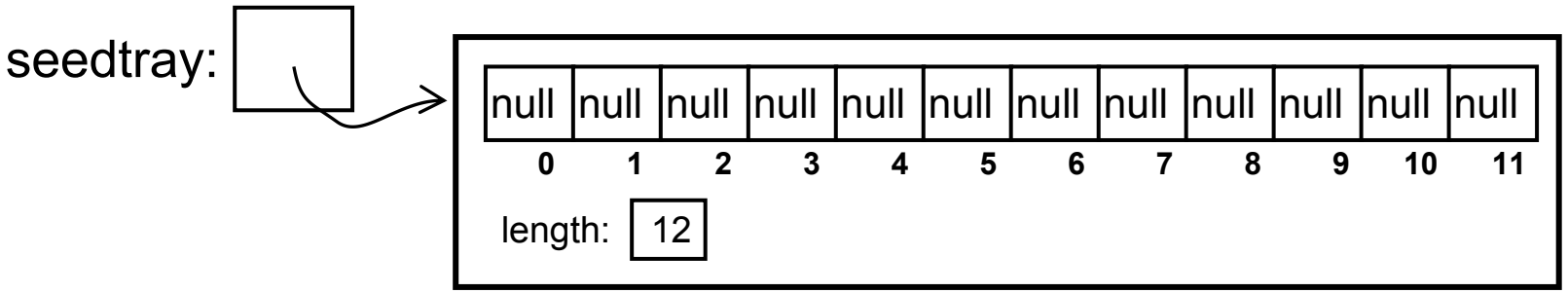


Arrays of numbers initialised to 0.

SeedTray Program

```
public class SeedTray {

    private  Flower[ ] seedtray = new  Flower[12];
```

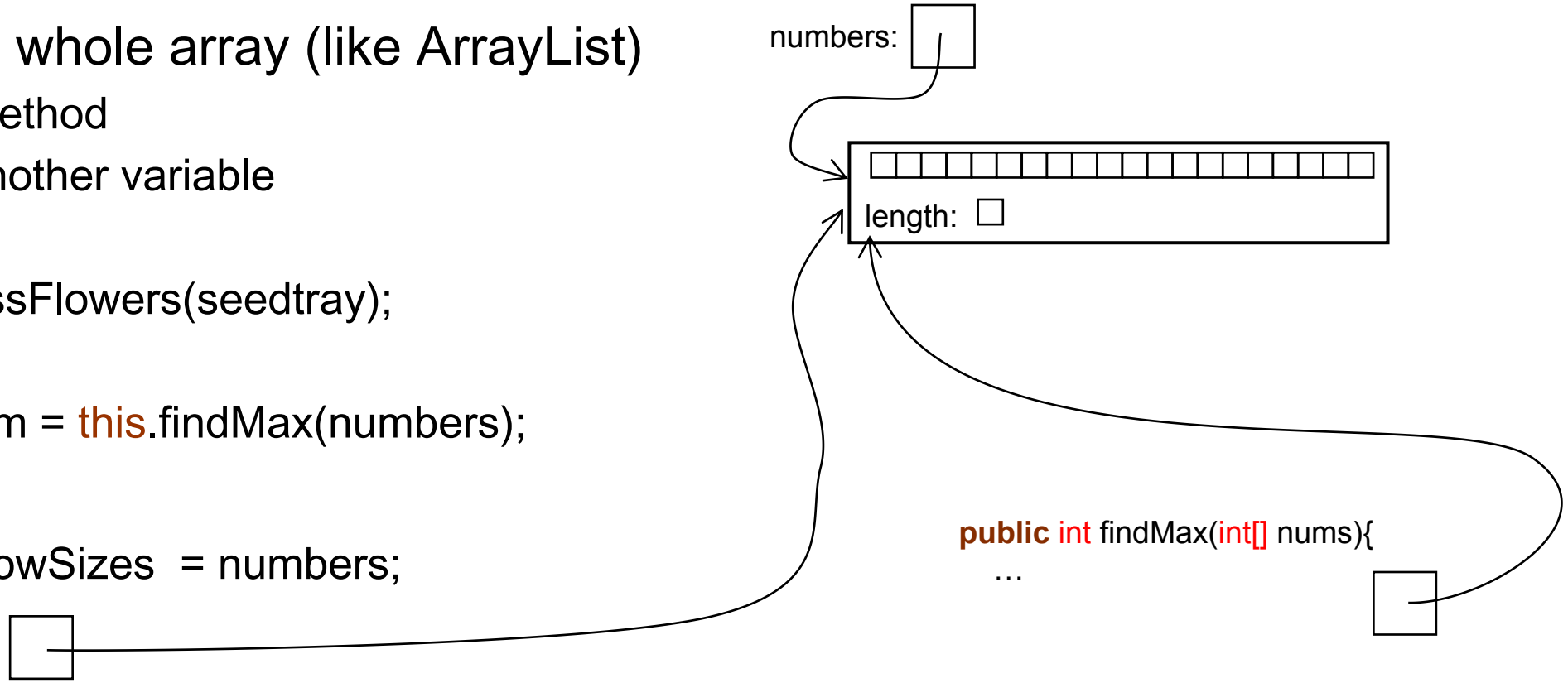
No 'Array' in declaration!



Using an array

- Can act on the whole array (like ArrayList)
  - to pass to a method
  - to assign to another variable

```
        :
this.processFlowers(seedtray);

int maxNum = this.findMax(numbers);

int [ ] windowSizes  = numbers;
```



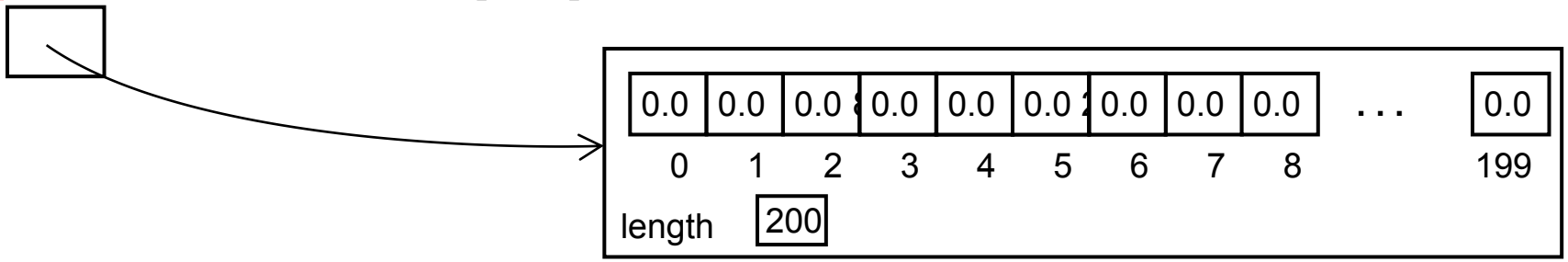
- Note, passing as argument and assignment do **not** copy the array! (just the reference/ID of the object)
- Just the same as with ArrayList.

Using an Array

- Use [..] to refer to an individual place in the array
  - to access the value in that place
  - to put a value in that place (using assignment: =)

Not get() and set()

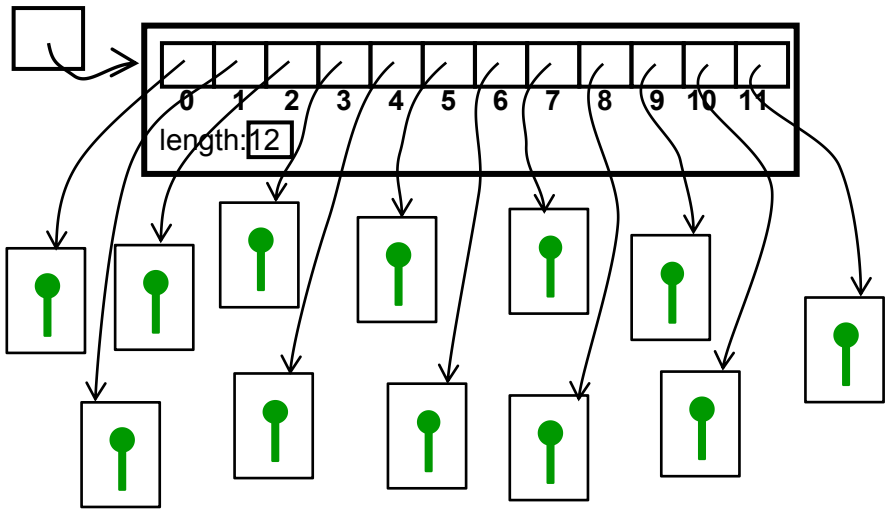
```
double [ ] marks = new double [200];

int n=4;
   :

marks[5] = 45.6;
marks[6] = ( marks[5] + marks[7] ) / 2;
marks[n-1] = 80.0;
marks[n] = marks[n-1];
if (marks[ i ] == marks[ i+1 ]) {…
```



Index can be any int valued expression

SeedTray Program

```
public class  SeedTray{
    private  Flower[ ] seedtray = new  Flower[12];
        :
    public void replant(){
        for (int i = 0; i < this.seedtray.length;  i++) {
            this.seedtray[ i ] = new  Flower(70+i*50, 400);
        }
    }
    public  void growAll(){
        for (int i = 0; i < this.seedtray.length; i++) {
            this.seedtray[ i ].grow();
        }          }
}
```



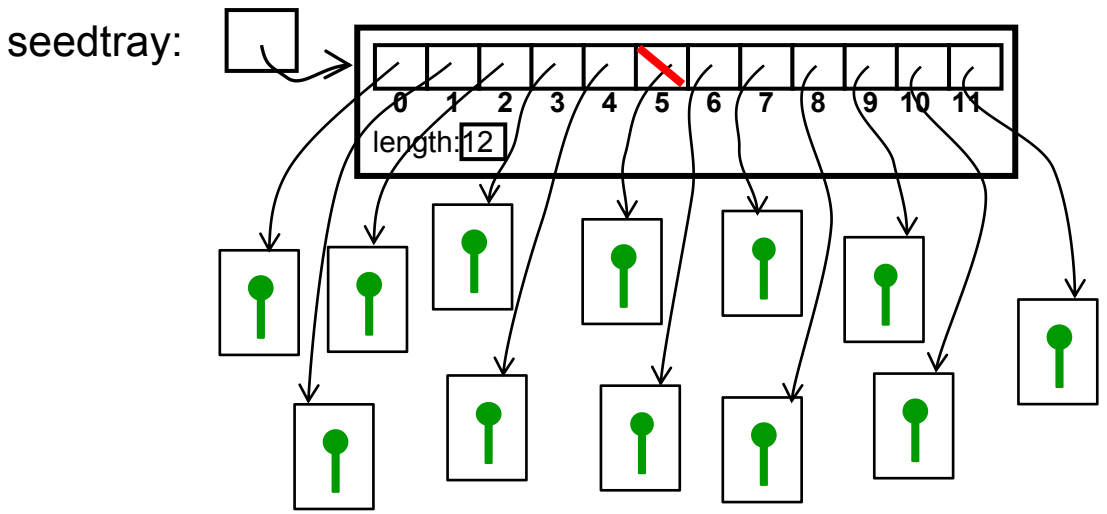

```
public  void growAll(){
    for (Flower flower : this.seedtray){
        flower.grow();

}
```

For each loop works on arrays, just like ArrayLists

Arrays of Objects can contain null

```
public void pick(int index){
    this.seedtray[ index ] = null;
}
```

If the array may have null, must check items before acting on them



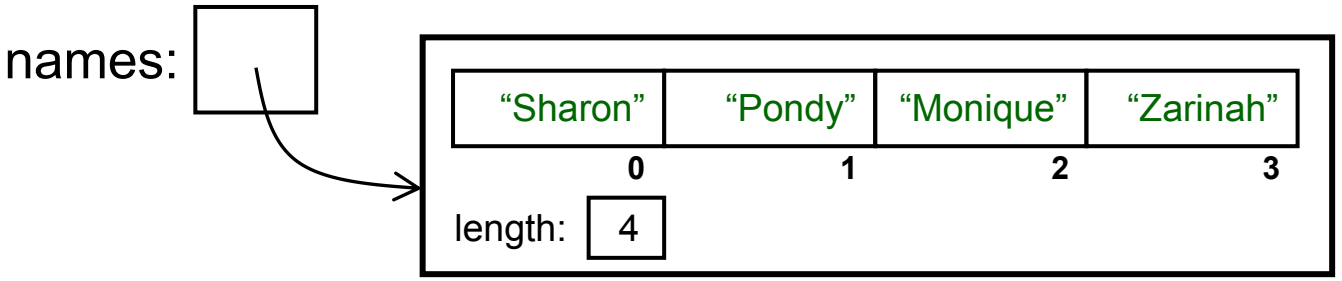

```
public  void growAll(){
    for (int i = 0; i < this.seedtray.length; i++) {
        if (this.seedtray[ i ] != null){
            this.seedtray[ i ].grow();
        }
    }
}
```

```
public  void growAll(){
    for (Flower flower : this.seedtray){
        if (flower != null){
            flower.grow();
        }
    }
}
```

Initialising the contents of an array

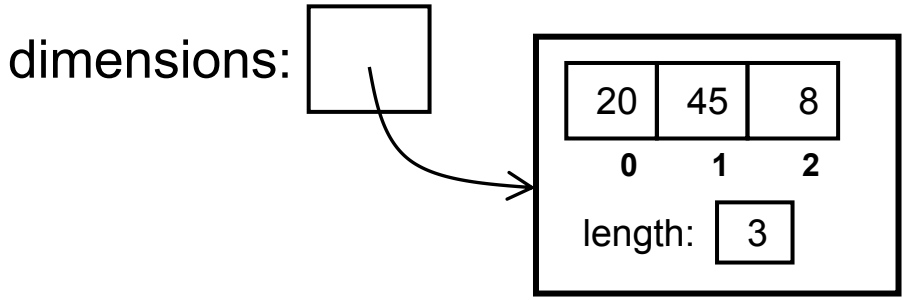
- Can specify the initial values (and size) of an array by listing the values in {.. , .. , ..} :

```
String [ ] names = new String [ ] { "Sharon", "Pondy", "Monique" , "Zarinah"  };
```



Can't do this with ArrayLists!

```
int [ ] dimensions = new int [ ] { 20, 45, 8 };
```



Arrays vs ArrayList

- Use an array if
  - it will never change size, and
  - you know how big it will need to be, at the point you need to create it.
  - speed is important to you.

- Use an ArrayList if
  - the size will change, or
  - you don't know how big it will need to be.

- Arrays have convenient syntax [ ]
- ArrayLists have convenient methods.

Comparing arrays.

- Be careful when comparing arrays (as with all objects)

```
int[ ] a = new int[ ]{ 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 43, 47};
int[ ] b = new int[ ]{ 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 43, 47};
int[ ] c = b;
```

```
if (a == b) ..          ➔ ??  no
if (b == c) ..          ➔ ??  yes
if (a.equals(b) ) ..    ➔ ??  no
if  (Arrays.equals(a, b) ) .. ➔  ??  yes
if  (this.myIntArrayEquals(a, b) ) .. ➔  ??  yes
```

```
public boolean myIntArrayEquals(int[ ] a, int[ ] b) {
    if (a==null && b==null ) { return true; }
    if (a==null || b==null ) { return false; }
    if ((a.length != b.length ) { return false; }
    for (int i = 0; i < a.length; i++) { if ( a[i] != b[i] ) { return false; } }
    return true;
}
```