# Arrays

- Objects on the heap
- Array Bounds
- Initialization
- Algorithms
- Multiple dimensions

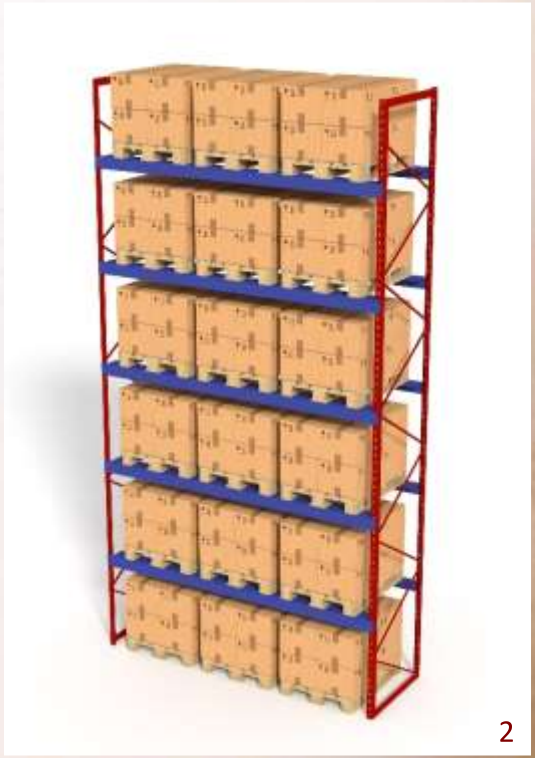
Introduction to Java

## See Also

http://codingbat.com/doc/java-array-loops.html

http://www.javatpoint.com/array-in-java

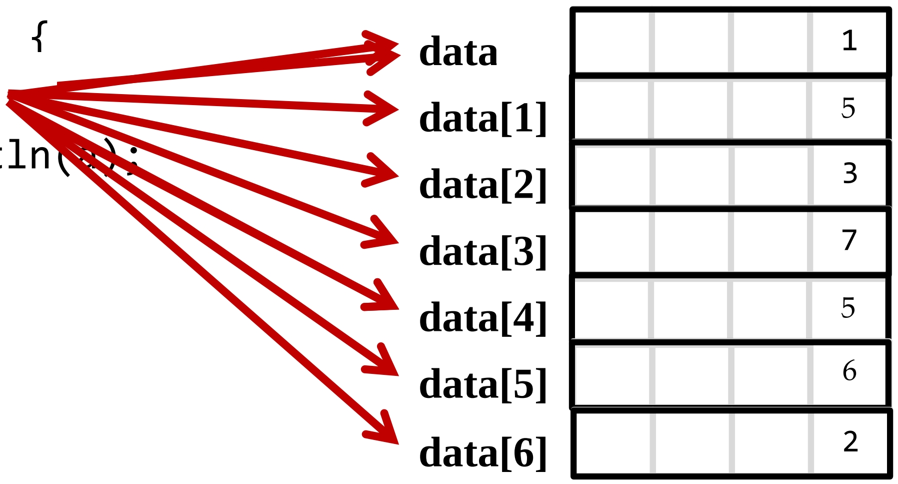
https://www.youtube.com/watch?v=dZb5ofv0twk



# Indexing

- Until now your code has been very explicit when accessing data.
- An array is a list of items under the same name. Each item has an index.
- You can use a variable for the index and visit each item in the array.

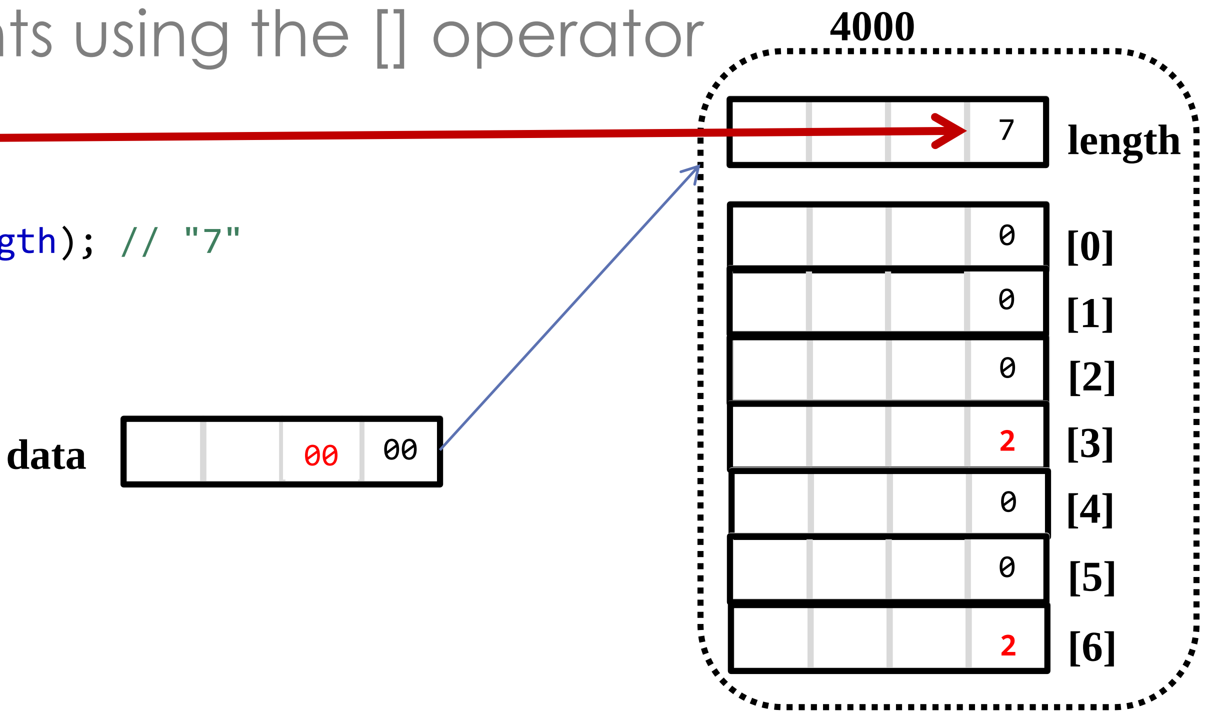
```
for(int x=0;x<7;++x) {
    int a = data[x];
    System.out.println(a);
}
```



# Arrays are Objects (heap)

- New built-in type … the "pointer to array"
- Arrays are always on the heap. You "new" them.
- The ".length" has the size
- Initialized to all 0s
- Access elements using the [] operator

```
int [] data = new int[7];

System.out.println(data.length); // "7"

data[3] = 2;

data[6] = data[3];

data = null;
```



# Array Bounds

- Array access is always checked at runtime
- Remember … 0 to (length-1)
- The “.length” is read-only

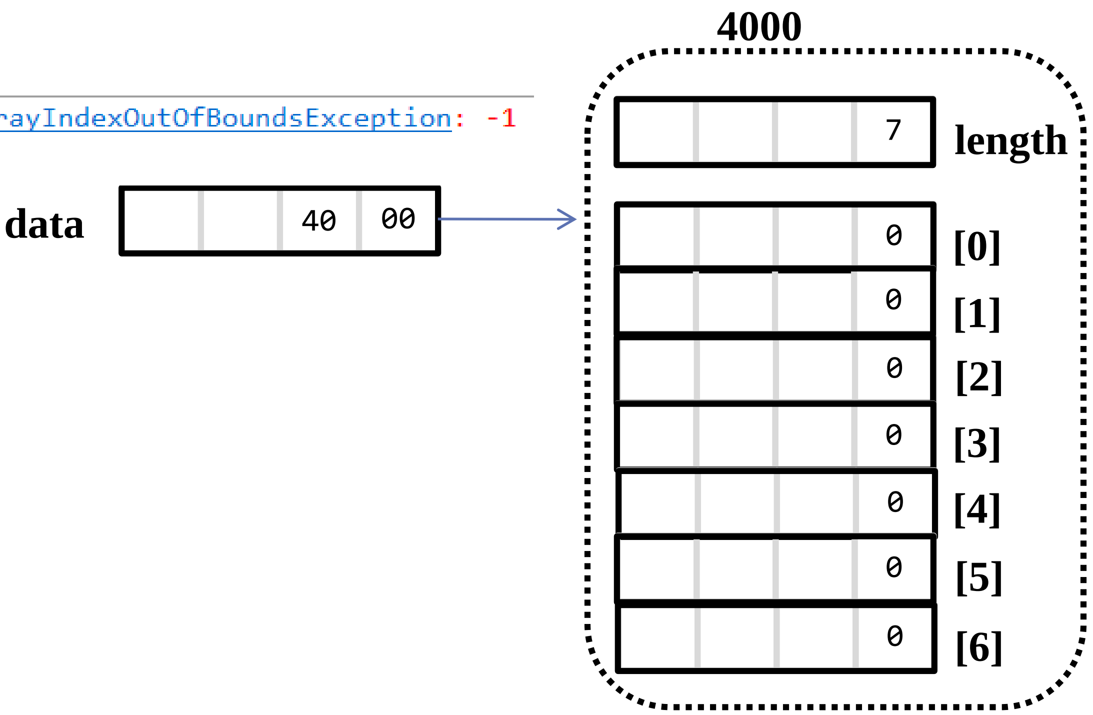
```
int [] data = new int[7];

data[-1] = 10;
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: -1
        at Tinker.main(Tinker.java:8)
```

```
data[7] = 10;

data.length = 100;
```

The final field array.length cannot be assigned

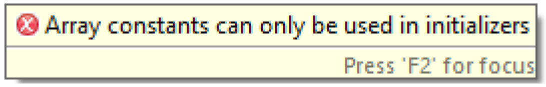
Press 'F2' for focus



## Initialization Lists

- Size can be determined at runtime
- Once set, it can never be changed
- You can list items with brackets and commas
- This only works when with “new”

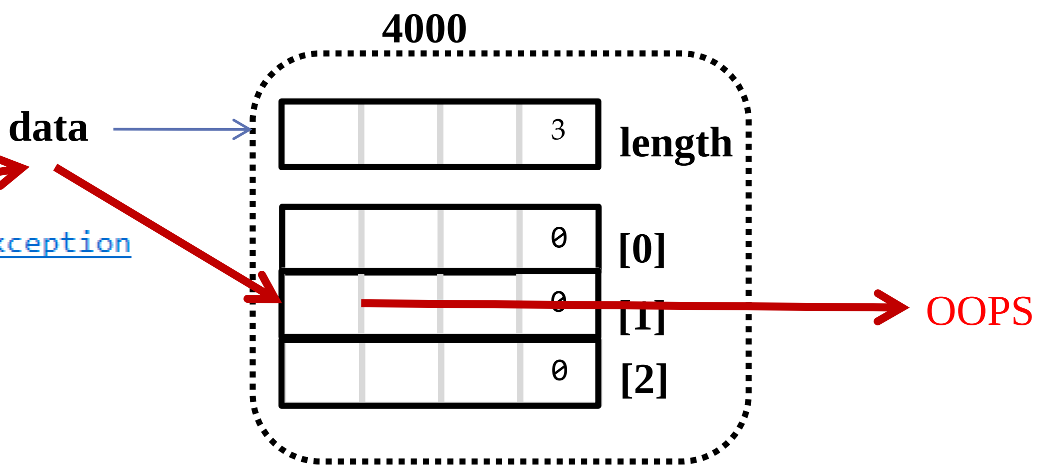
```
int x = askUserForANumber();
int [] data = new int[x];

int [] data = {1,5,3,7,5,6,2};

data = {2,3,4};
```


Array constants can only be used in initializers
Press 'F2' for focus

# Arrays of Pointers

- Arrays of objects are arrays of pointers-to-objects
- The array is initialized to all null pointers (0)
- You can combine [] and "."

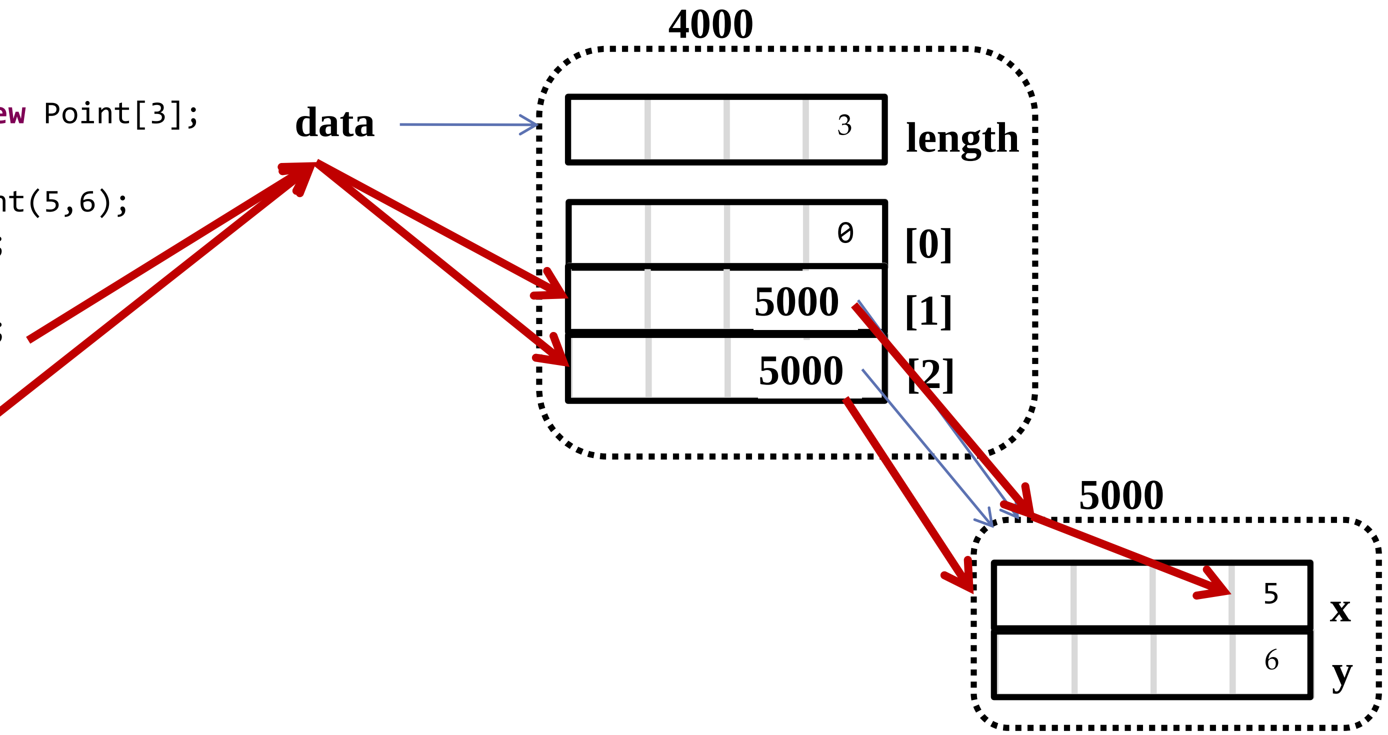
```
Point [] data = new Point[3];

int a = data[1].x;
          java.lang.NullPointerException

data[2].print();
```



# Arrays of Pointers

- You have to "new" the elements
- Remember: One "new" means one object

```
Point [] data = new Point[3];

data[1] = new Point(5,6);
data[2] = data[1];

int a = data[1].x;

data[2].print();
```



# Arrays as Parameters

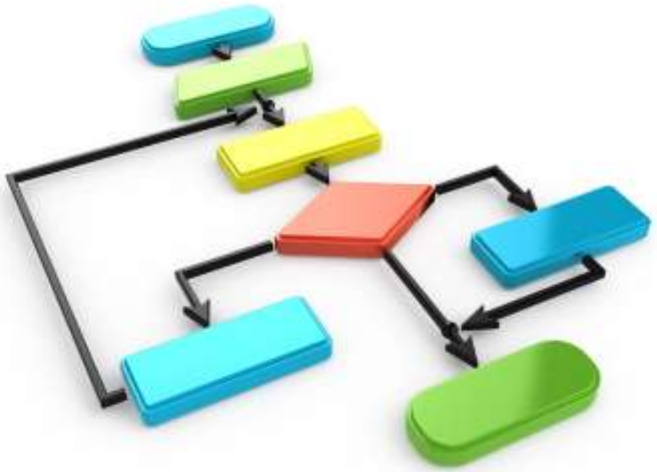
- Arrays are objects. You pass pointers.

```
public class Tinker {

    public static double average(int [] data) {
        double ret = 0.0;
        for(int x=0;x<data.length;x=x+1) {
            ret = ret + data[x];
        }
        ret = ret / data.length;
        return ret;
    }

    public static void main(String [] args) {
        int [] data = {1,5,3,7,5,6,2};
        double av = average(data);
        System.out.println(av); // 4.142857...
    }

}
```

## Algorithms

- Sort an array "in place"

```
public static void main(String [] args) {

    int [] data = {2,1,100,7,5,6,2};

    sort(data);

    for(int x=0;x<data.length;++x) {
        System.out.println(data[x]);
    }

}
```



# Algorithms

- Swap adjacent if needed

```
public static void sort(int [] data) {

    int a = data[0];
    int b = data[1];

    if(b<a) {
        data[0] = b;
        data[1] = a;
    }

}
```

# Algorithms

- Loop through array

```
public static void sort(int [] data) {
    for(int x=0;x<data.length-1;++x) {
        int a = data[x];
        int b = data[x+1];

        if(b<a) {
            data[x] = b;
            data[x+1] = a;
        }
    }
}
```

# Algorithms

- Many loops

```
public static void sort(int [] data) {
    for(int y=0;y<1000;++y) {
        for(int x=0;x<data.length-1;++x) {
            int a = data[x];
            int b = data[x+1];

            if(b<a) {
                data[x] = b;
                data[x+1] = a;
            }
        }
    }
}
```

# Algorithms

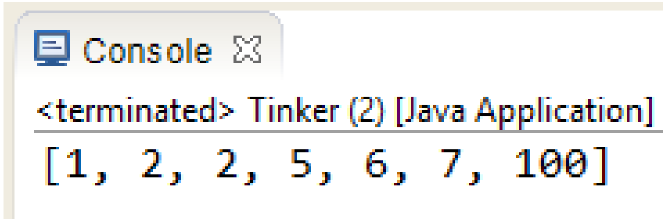
- Loop when needed

```
public static void sort(int [] data) {
        boolean changed = true;
        while(changed) {
            changed = false;
            for(int x=0;x<data.length-1;++x) {
                int a = data[x];
                int b = data[x+1];

                if(b<a) {
                    data[x] = b;
                    data[x+1] = a;
                    changed = true;
                }
            }
        }
    }
```

# Array Tools

- The "Arrays" class has many useful static methods

```
import java.util.Arrays;

public class Tinker {

    public static void main(String [] args) {

        int [] data = {1,2,100,7,5,6,2};

        Arrays.sort(data);

        System.out.println(Arrays.toString(data))

    }

}
```

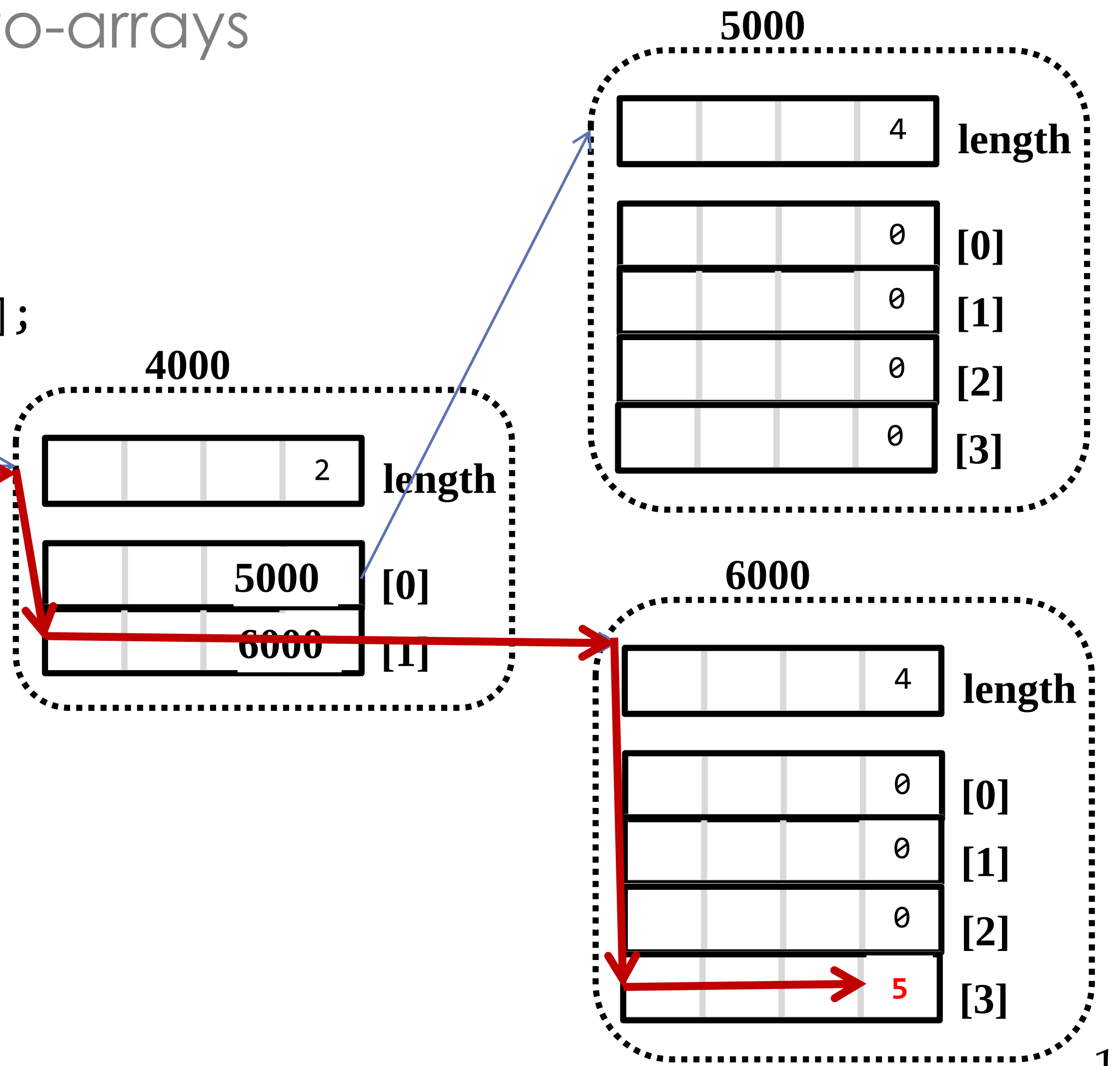


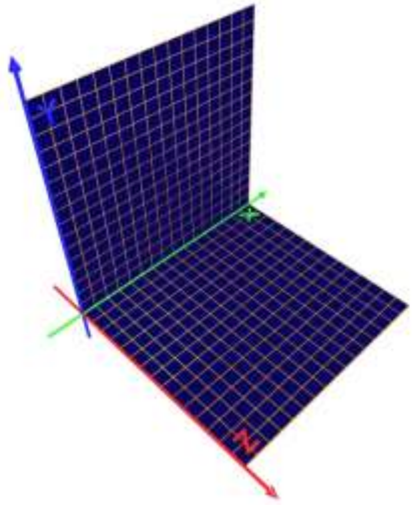
```
<terminated> Tinker (2) [Java Application]
[1, 2, 2, 5, 6, 7, 100]
```

# Multidimensional Arrays

- Arrays of pointers-to-arrays
- Can be ragged

```
int [][] data = new int[2][4];

data[1][3] = 5;
```

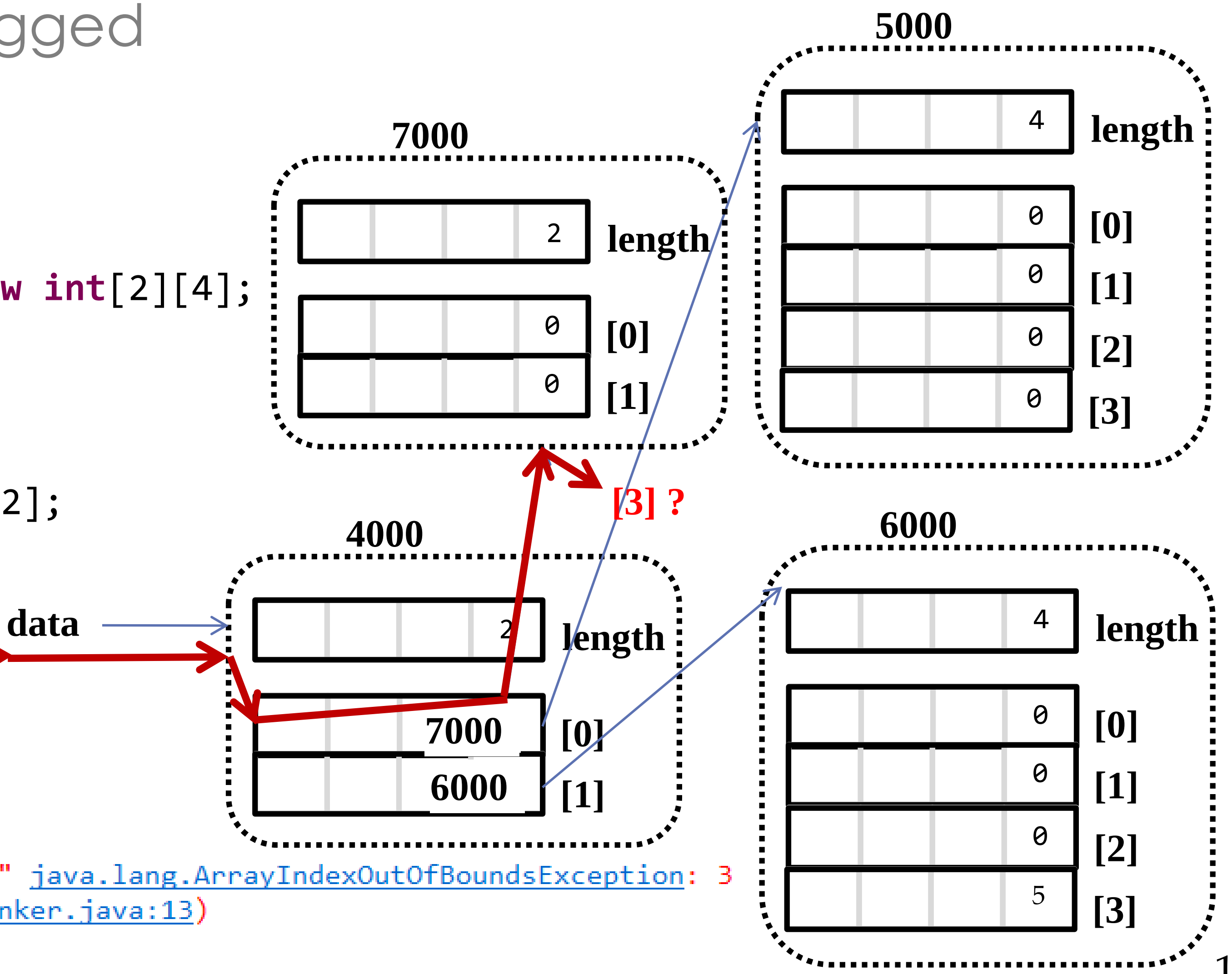




# Multidimensional Arrays

- Can be ragged

```
int [][] data = new int[2][4];

data[1][3] = 5;

data[0] = new int[2];

data[0][3] = 5;
```

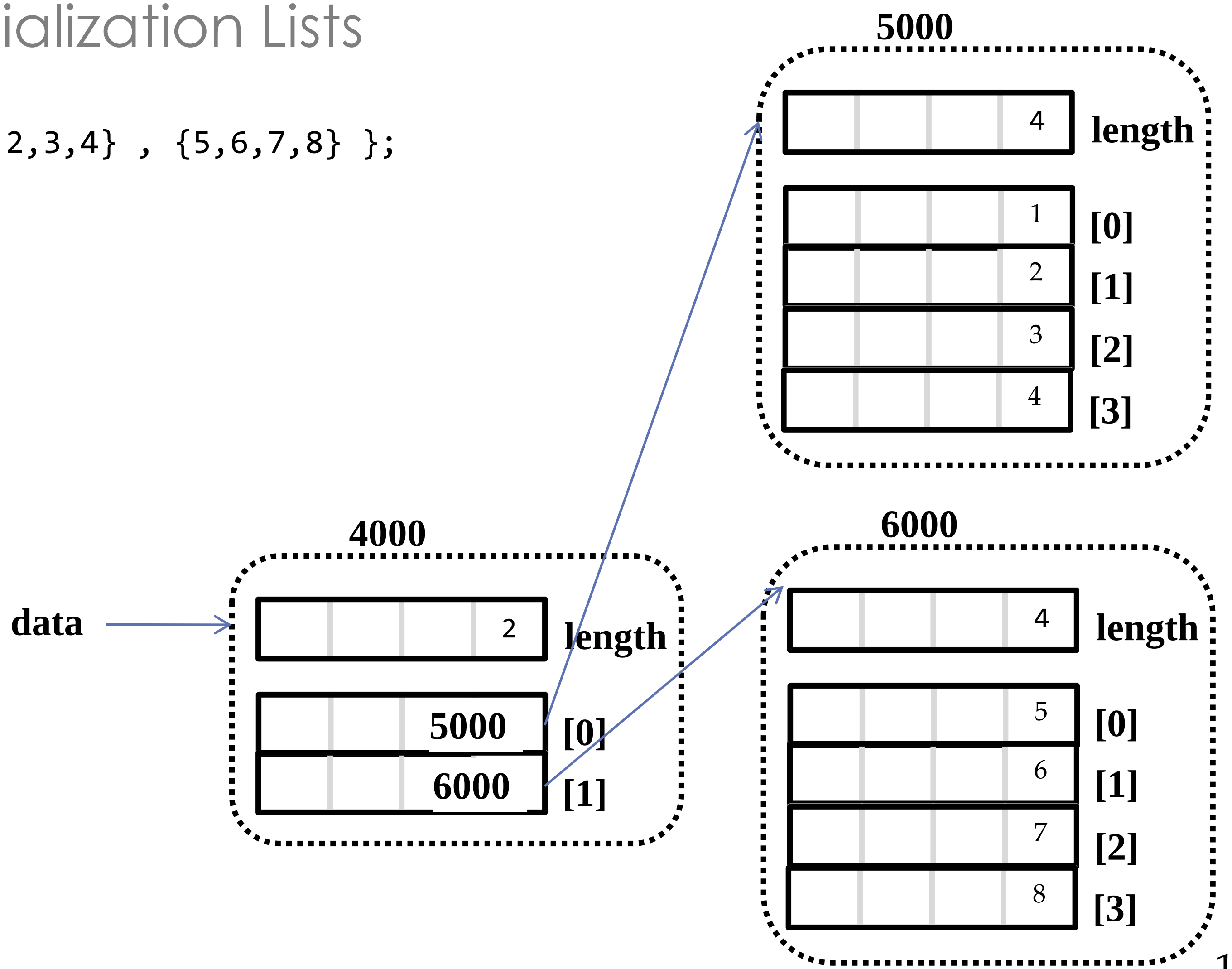


```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3
	at Tinker.main(Tinker.java:13)
```

# Multidimensional Arrays

- Nested Initialization Lists

```
int [][] data = { {1,2,3,4} , {5,6,7,8} };

int [][] data2 =
    {
        {1,2,3,4} ,
        {5,6,7,8} ,
        {1,2}
    };
```

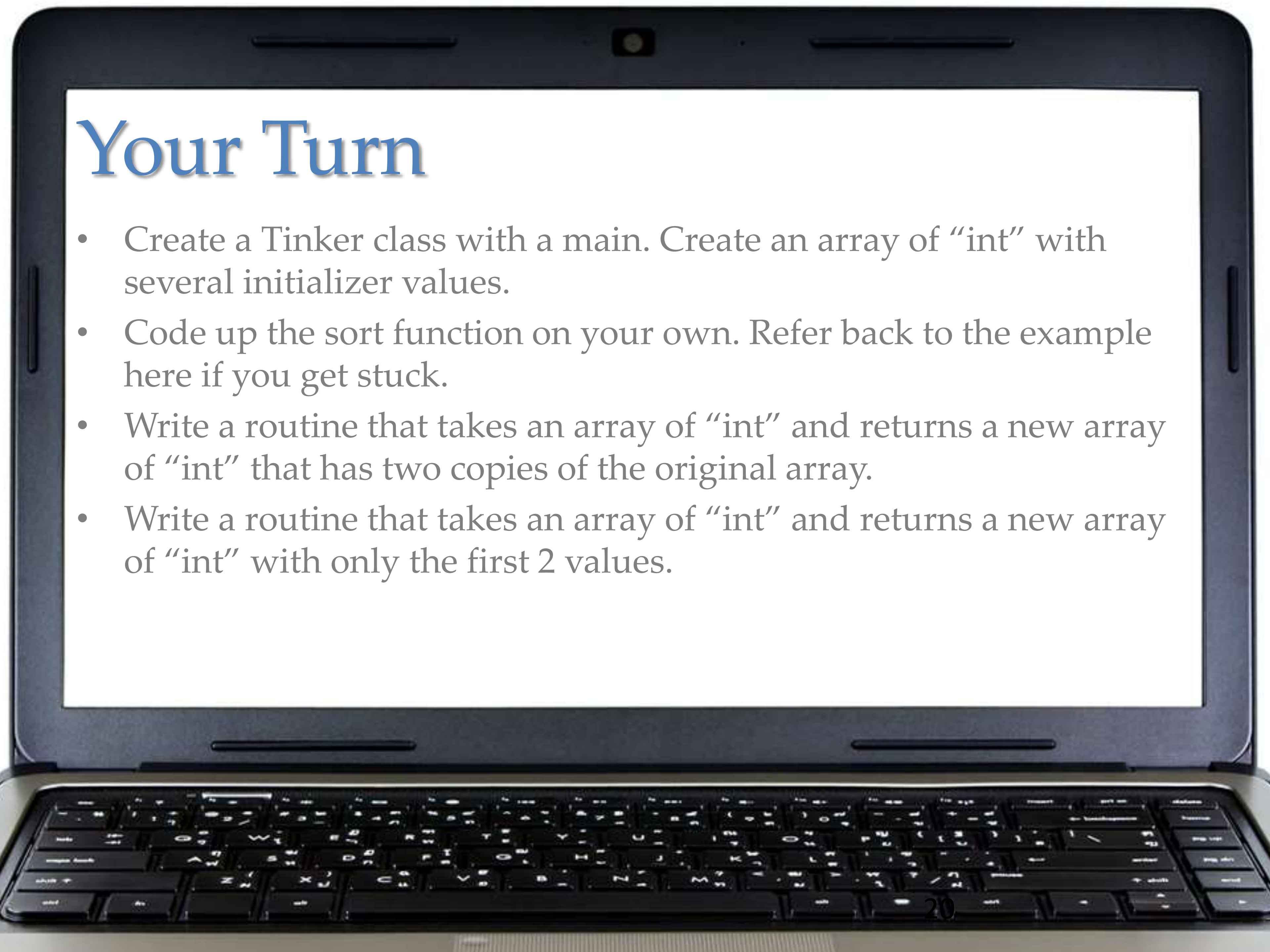


# In Passing

- Flexible bracket placement
- Initialization lists are processed at runtime

```
int []data1 = {1,2,3};

int[] data2 = {1,2,3};

int data3[] = {1,2,3};

int[] data4[] = {{1,2,3},{4,5,6}};
```

```
int a = 12;

int [] g = {1,a,doStuff()*4,rand(50)};

int [] g = new int[4];
g[0] = 1;
g[1] = a;
g[2] = doStuff()*4;
g[3] = rand(50);
```



## Your Turn

- Create a Tinker class with a main. Create an array of “int” with several initializer values.
- Code up the sort function on your own. Refer back to the example here if you get stuck.
- Write a routine that takes an array of “int” and returns a new array of “int” that has two copies of the original array.
- Write a routine that takes an array of “int” and returns a new array of “int” with only the first 2 values.