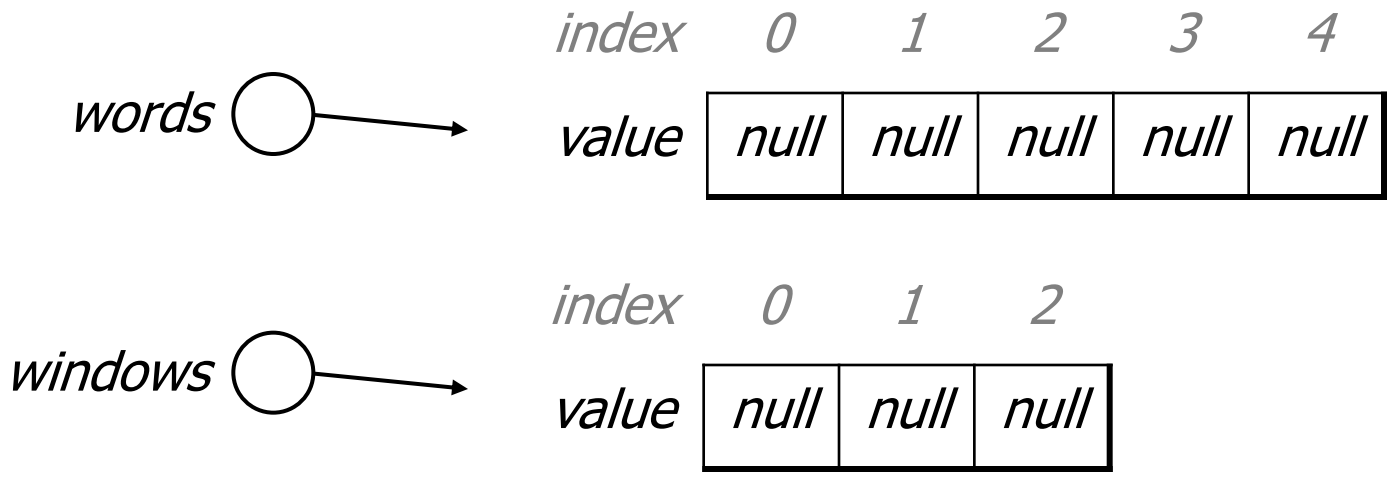
AP CS:
Arrays Of Objects
& `null`

Subset of the Supplement Lesson slides from: Building Java Programs, Chapter 8 somewhere by Stuart Reges and Marty Stepp (http://www.buildingjavaprograms.com/) & thanks to Ms Martin.

Arrays of objects

- **`null`:** A value that does not refer to any object.
  - The elements of an array of objects are initialized to `null`.

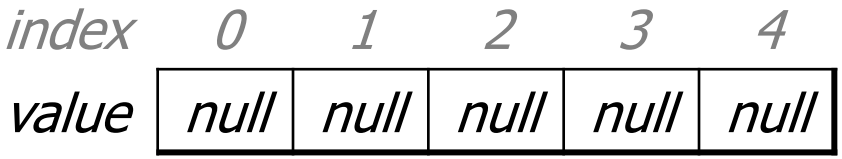
```
String[] words = new String[5];
DrawingPanel[] windows = new DrawingPanel[3];
```



Null pointer exception

- **dereference**: To access data or methods of an object with the dot notation, such as `s.length()` .
  - It is illegal to dereference `null` (causes an exception).
  - `null` is not any object, so it has no methods or data.

```
String[] words = new String[5];
System.out.println("word is: " + words[0]);
words[0] = words[0].toUpperCase();   // ERROR
```



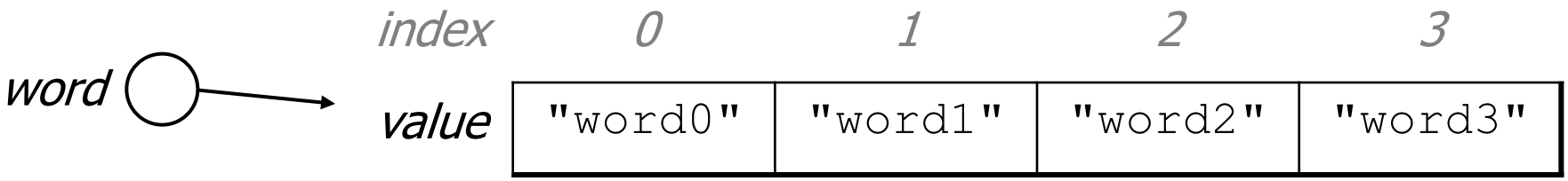
Output:

```
word is: null
Exception in thread "main"
java.lang.NullPointerException
        at Example.main(Example.java:8)
```

Two-phase initialization

1) initialize the array itself (each element is initially `null`)
2) initialize each element of the array to be a new object

```
String[] words = new String[4];              // phase 1
for (int i = 0; i < words.length; i++) {
    coords[i] = "word" + i;                  // phase 2
}
```

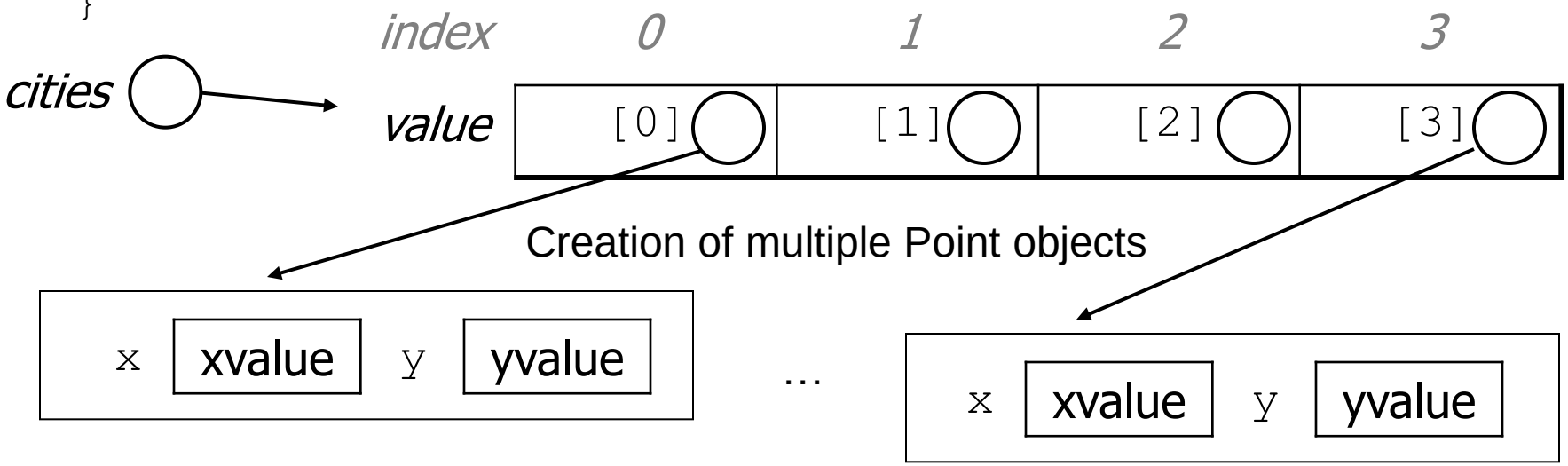


Two-phase initialization

1) initialize the array itself (each element is initially `null`)

2) initialize each element of the array to be a **`new`** object

```
int numCities= scan.nextInt();
Point[] cities = new Point[numCities];      // phase 1
for (int i = 0; i < cities.length; i++) {
    int xvalue = scan.nextInt();
    int yvalue = scan.nextInt();
    Point[i] = new Point(xvalue, yvalue); // phase 2
}
```



Arrays as Fields

- What if you don't know the length of an Object's array till you Construct it?
- Then simply first declare it like you would any variable:

```
// declare our fields (data)
      private int numClasses;
      private int [] scores; // Note: No Size
```

- And then "Instantiate" it in the constructor when you have the length available as a parameter.

```
// Constructor for Object GradeBook
      public GradeBook (int numClasses) {
         this.numClasses = numClasses;
// We now "Instantiate" the array to its size
         scores = new int [numClasses];
   }
```

FYI: Things you can do w/ `null`

- store `null` in a variable or an array element

```
String s = null;
words[2] = null;
```

- print a `null` reference

```
System.out.println(s);        // null
```

- ask whether a variable or array element is `null`

```
if (words[2] == null) { ...
```

- pass `null` as a parameter to a method

```
System.out.println(null);     // null
```

- return `null` from a method (often to indicate failure)

```
return null;
```

Looking before you leap

- You can check for `null` before calling an object's methods.

```
String[] words = new String[5];
words[0] = "hello";
words[2] = "goodbye";    // words[1], [3], [4] are null

for (int i = 0; i < words.length; i++) {
    if (words[i] != null) {
        words[i] = words[i].toUpperCase();
    }
}
```