Java Arrays

Single Dimensional

Q: Why did the programmer quit his job?

A: Because he didn't get arrays.

(a raise)

What is an Array?

- A Java Class, which extends (IS-A) Object
- Acts like a container with a pre-defined & constant number of buckets
- Holds values of a pre-defined type:
  - Primitives: byte, char, short, int, etc….ArrayList can’t
  - Wrappers - Integer, Long, Double, etc….
  - Objects - Dog, Cat, Jet, FoodTrucks, Cards, etc…

Purpose

- Arrays offer an easy way to store multiple values (in one reference variable)
- Iterate through those values
- Work with (any known number) of values

Creating Arrays

- Type [ ] referenceName = new Type [arrayLength];
- Type referenceName [ ]= new Type [arrayLength];
- Type [ ] referenceName;
  - referenceName = new Type [arrayLength];
    - ⋆ arraySize can be set manually, or by the use of a variable

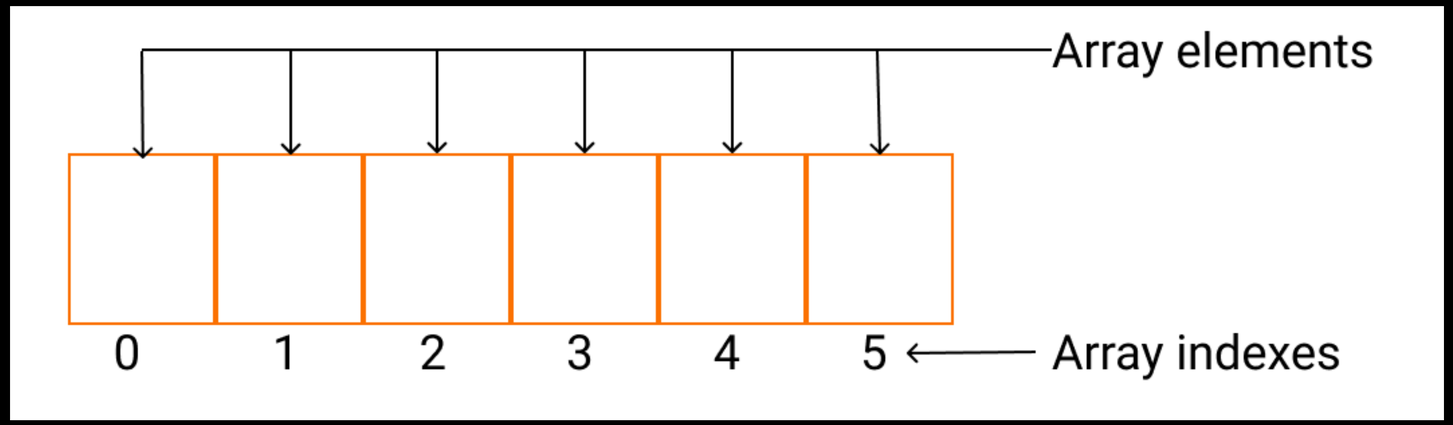
Creating Arrays

Example

```
String [ ] names = new String [2];

int numbers [ ] = new int [13];

double [ ] balance;

    balance = new double [8];
```

Array Indexes/Indices

- An Array's index is the "bucket" that its data is kept in
- Indexes begin at 0 & increment by 1
- They end at the arrayLength you provide minus 1.



Array Indexes/Indices

Example

```
String [ ] names = new String [2];

System.out.println("Length of names Array: " + names.length);

for (int i = 0; i < names.length; i++) {
    System.out.println("Index " + i + " of the names Array");
}
```

Result

```
Length of names Array: 2
Index 0 of the names Array
Index 1 of the names Array
```

Happy Hour

Three programmers walk into a bar...

Filling Arrays:

- At creation/instantiation: - “Anonymous Array”
- Type [ ] referenceName = {comma-delimited values};
  - Type referenceName [ ] = {comma-delimited values};

```
String [ ] names = {"Spock", "Loki"};
```

Filling Arrays Manually:

Example

```
String [ ] names = new String [2];

names[0] = "Spock";
names[1] = "Loki";
```

Filling Arrays using Loops

Example

```
int [ ] numbers = new int [3];

for(int index = 0; index < numbers.length; index ++) {
    numbers[index] = (index + 1); //or any int
    System.out.println("Numbers at index " + index + " = " + numbers[index]);
}
```

Result

```
Numbers at index 0 = 1
Numbers at index 1 = 2
Numbers at index 2 = 3
```

Pulling From Arrays

Example

```
int [ ] numbers = {13, 8, 0};

for(int index = 0; index < numbers.length; index ++) {
    System.out.println(numbers[index]);
}
```

Result

```
13
8
0
```

Pulling From Arrays

Example

```
String [ ] namesArray = {"Spock", "Loki", "Yoda"};

for(String name : namesArray) {
    System.out.println(name);
}
```

Result

```
Spock
Loki
Yoda
```

Useful Stuff

- Package: java.util
- Class: Arrays
- .length
  - Returns the size of the array (last index + 1)
- .clone( );
  - a deep-copy is created of the array
  - New (duplicated) array object is returned - with a new location in memory

Useful Stuff

- Arrays.sort( );
  - int [ ] numbers = {13, 8, 9};
    - Arrays.sort(numbers);
    - Results in 8, 9, 13
  - String [ ] stringNums = {“13”, “8”, “9”};
    - Arrays.sort(stringNums);
    - Results in 13, 8, 9 (sorted alphabetically)

Useful Stuff

- Searching
  - Array must be sorted first to get predictable results
  - Arrays.binarySearch(arrayName, searchTerm)
    - If found - returns the index of the item
    - If not found - returns the index where it would be -1