Set 14: Arrays Part 1

Skill 14.01: Describe how arrays store data
Skill 14.02: Declare, Initialize, and Populate an Array
Skill 14.03: Iterate Over an Array

Skill 14.01: Describe how arrays store data

Skill 14.01 Concepts

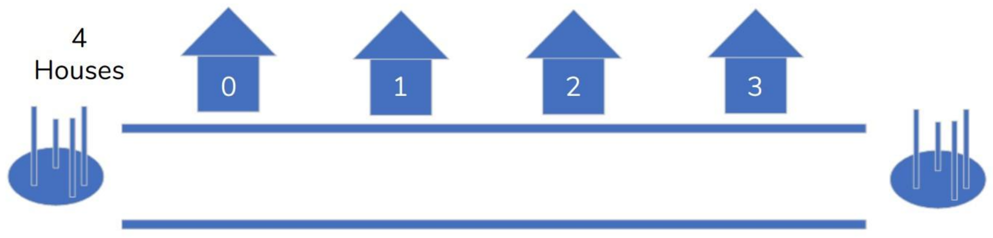
In our day-to-day lives we often encounter the need to make lists. For example, if we are packing for a trip we can make a packing list; if we are going to the grocery store, we may bring a grocery list. In this lesson, we will start looking at how we can use lists in our programs.

Anything that can store data can be called a data structure. Hence, integer, boolean, char, double etc, are all data structures. They are known as Primitive Data Structures. In this lesson we will explore a new type of data structure which will allow us to store data as a list. This type of data structure is referred to as an *array*

An ***array*** is a specific data structure that stores data as a list

One way we can visualize an array, is to imagine a street of houses. Notice on our street below, each house has an address and that the addresses start at "0". Each address in our example is also referred to as an *index*. The contents inside each house are referred to as the *value* associated with the index.

index refers to the location of an element in an array.
value refers to the contents stored at the index.



Skill 14.1: Exercise 1

Skill 14.02: Declare, Initialize, and Populate an Array

Skill 14.02 Concepts

An array in java can be declared, initialized, and populated in at least two ways. Each are described below.

Method 1

The code snippet below illustrates one method for declaring, initializing, and populating an array. This method is only useful if we know the identity of the elements we want to store.

Code

```
String houses[] = {"Bart", "Kyle", "Bugs", "Marvin"};
```

- String - The type of array we want to declare. This can be any type, String, int, double, etc.
- houses[] - The name of the array is *houses* and the brackets, [], following the name, tells java that this is an array.
- {"Bart", "Kyle", "Bugs", "Marvin"} - The String data types we want to store in the array.

Method 2

The code snippet below illustrates another method for declaring an array. In the below example the array is only declared, not initialized.

Code

```
String houses[];
```

- String - The type of array we want to declare. This can be any type, String, int, double, etc.
- houses[] - The name of the array is *houses* and the brackets, [], following the name, tells java that this is an array.

To initialize the array we use the *new* keyword to tell java this is a new array. Additionally, we need to tell java the datatype the array will store, along with how many items the array can hold. The code snippet below illustrates how to initialize the array we declared above to hold up to 4 items.

Code

```
houses = new String[4];
```

While the array above has been declared and initialized, it currently does not hold any values. The below code snippet illustrates how to populate each location in the array with a value. If no value is specified, the location will be assigned the value *null*.

Code

```
houses[0] = "Bart";
houses[1] = "Kyle";
houses[2] = "Bugs";
houses[3] = "Marvin";
```

- houses - The name of the array in which we want to store the item.
- [0] - The *index* in which we want store the item.
- "Bart" - The *value* of the item we want to store

While the above examples how to declare, initialize, and populate an array of String datatypes. Arrays for double, int, char, and boolean types are also possible. They are declared, initialized, and populated in exactly the same way. We can even make arrays of objects - but, more on that later!

[Skill 14.2: Exercise 1](#)

Skill 14.03 Iterate over an array

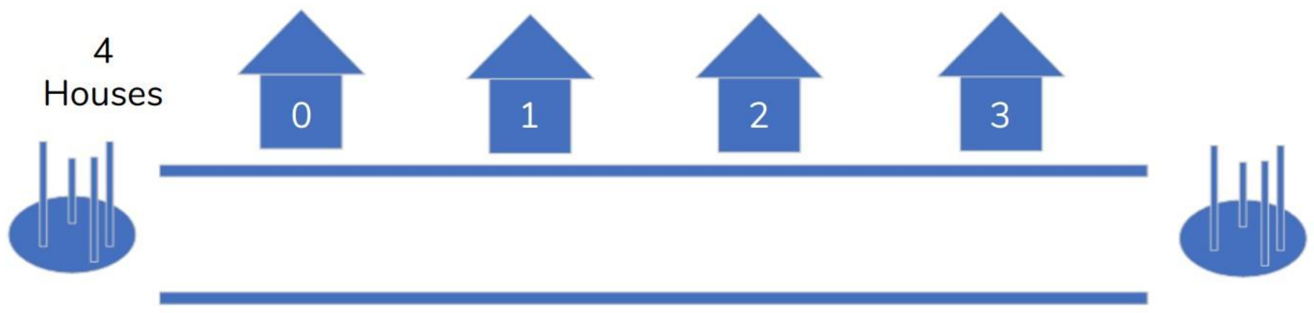
Skill 14.03 Concepts

Iterating over an array is the process of accessing each element of array one by one. Accessing each element in an array requires that we know both the address of the first element and the address of the last element in the array. The first element in an array is always located at index = 0, the last element of an array can be found if the length of the array is known. The syntax for finding the length of an array is illustrated below,

Code	Output

`String houses = new String[4];`
`int addresses = houses.length;`
`System.out.println(addresses);`
	4

In the illustration below, we see that the address of the last house in our array is one less than the length.



Consider the following code snippet,

Code	Output
`houses[houses.length] = "Wirt";`	`Array index out of bounds error`
`houses[houses.length-1] = "Wirt";`	`Nothing`

Now that we know the address of the first and last element, we can apply the loops we've learned previously to iterate over any array. In each example, we will iterate over the array below,

Code

```
String houses = new String[4];

houses[0] = "Bart";
houses[1] = "Kyle";
houses[2] = "Bugs";
houses[3] = "Marvin";
```

for-loops

Code

```
for(int a = 0; a < houses.length; a++){
    System.out.println(houses[a] + " lives in house " + a);
}
```

Output

```
Bart lives in house 0
Kyle lives in house 1
Bugs lives in house 2
Marven lives in house 3
```

While-loops

Code

```
int address = 0;

while(address < houses.length){
     System.out.println(houses[address] + " lives in house " +
address);
     address++;
}
```

Output

```
Bart lives in house 0
Kyle lives in house 1
Bugs lives in house 2
Marven lives in house 3
```

do-while-loops

Code

```
int address = 0;

do{
     System.out.println(houses[address] + " lives in house " +
address);
     address++;
}while(address < houses.length);
```

Output

```
Bart lives in house 0
Kyle lives in house 1
Bugs lives in house 2
Marven lives in house 3
```

for-each

A *for-each* loop is another type of loop that allows us to access each element in an array. The syntax for a *for-each* loop is illustrated below,

Code

```
for(String a:houses){
     System.out.println(a);
}
```

Output

```
Bart lives in house 0
Kyle
Bugs
Marven
```

The syntax above can be translated as follows: *for each String "a" in houses* and is described below,

- String a - String refers to the type of data stored and *a* is the variable to which each element is assigned. You can think of this portion as reading *for each String "a"*
- : - the colon associates the element we are accessing to the array in which it is stored. You can think of this portion as reading *in*
- houses - The name of the array in which each element is stored.

Skill 14.3: Exercises 1 and 2