Arrays

Dr. Krishnendu Guha (kguha@ucc.ie)

Teaching Assistants

Zarrar Haider (123120583@umail.ucc.ie)

Rheena Blas (120347046@umail.ucc.ie)

Python Lists

- **Lists** are **built into Python** (along with sets, dictionaries, tuples)
- They are **easy to create**.
- They can **even contain different types of data** (probably not a good idea).
- Python **has operators, functions and methods to access elements of the list, to grow the list, to shrink the list, etc**.


```
groceries = ["eggs", "bread", "milk"]
my_fave_things = []
stupid_list = ["Charlywawa", -23, [1, 2.6, 3], True]
print( groceries[1] )
groceries[2] = "fat-free milk"
groceries = groceries + ["bacon"]
print( len( my_fave_things ) )
my_fave_things.append( "Java" )
my_fave_things.insert( "Python", 0 )
groceries.remove("eggs")
stupid_list.pop(2)
```

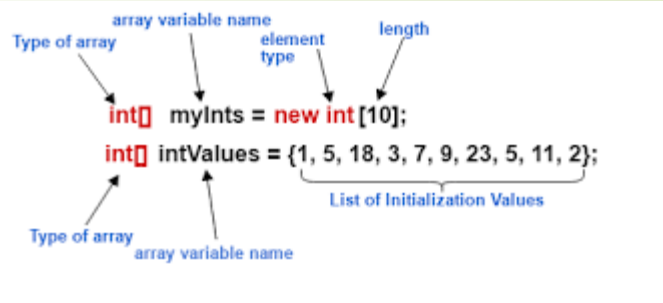
Java Arrays

- Java **does not have lists** (or sets, tuples or dictionaries) built in
- We can **write our own class definitions for them**, or **use class definitions from the Java SE class library**
- Java has **arrays built into the language** (an **indexed list of values**)

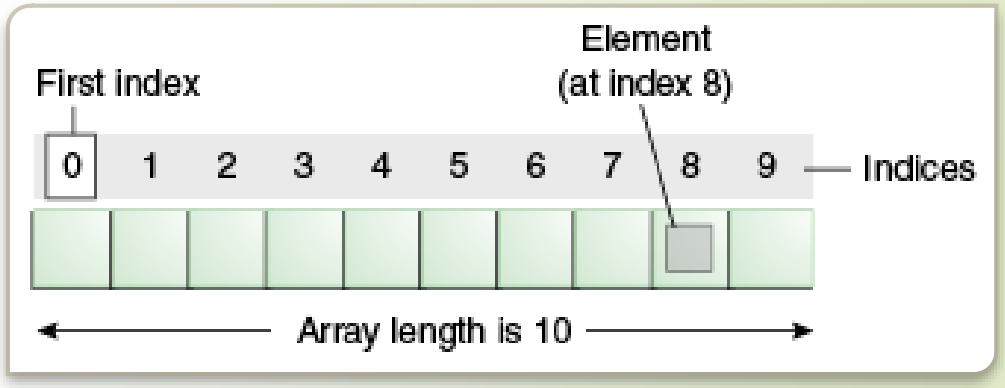
```
public class Main {
public static void main(String[] args) {
int my_array[] = {1, 2, 3, 4};
int mul = 1;

for (int i : my_array)
    mul *= i;
System.out.println("The product is " + mul);
}
}
```

- They **may not seem very flexible**:
- They are **fixed length**, which you must choose when you create them
- All **data in an array must be of the same type** (but see polymorphism later!) – but arrays, in general, can be of any type.
- All you can do to them is access the elements and find out the length
- But **accessing elements of an array is efficient**
- (In fact, underneath, Python lists are usually implemented using arrays in C)



- In computer science, an array is **a data structure consisting of a collection of elements (values or variables), each identified by at least one array index or key**.
- An *array* is a **container** object that holds a **fixed number of values of a single type.**
- The **length of an array is established when the array is created**. After creation, its length is fixed.



- **Each item in an array is called an *element***, and each element is accessed by its **numerical *index***.
- Numbering begins with 0.
- The 9th element, for example, would therefore be accessed at index 8.

Creating arrays

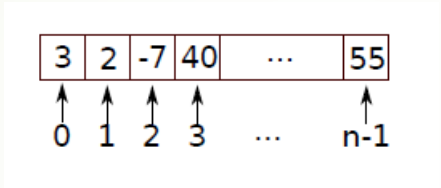
- Suppose you want an array that can hold 4 integers, referenced from a **variable** called **times**.
- **Declare the variable** with its type: int[] times;
- *Note: it will be a mistake at this stage to think that your array is created!*

- **Create** the array: times = new int[4];

- Of course, you can instead **do both** in one statement: int[] times = new int[4];
- The elements of the array are initialized to the usual default values

Accessing Arrays

int[] arr;



- Arrays are indexed from 0
- **Storing data** in the array: times[0] = 3;
- **To print all elements**, you need a loop because the following won't print them:

  System.out.println(times);
- That's how you find **the length (and print):** System.out.println(times.length);
- **Accessing data stored** in the array: System.out.println(times[1]);

Question: Is length a variable or method?

Declaring, creating and initializing arrays

Summary: The following declares a variable, creates an array (with default initial values), references it from the variable, and then stores data in the array:

```
int[] times = new int[4];

times[0] = 154;

times[1] = 176;

times[2] = 154;

times[3] = 135;
```

You can also do this!

```
int[] times = {154, 176, 154, 135};
```

Array of Objects

- Arrays can contain ints, doubles, booleans (primitive types)
- But they can also contain objects, e.g. Strings, Scanners, Randoms, Dogs (reference types)
- E.g. an array of Strings

```
String suits = {"Clubs", "Diamonds", "Hearts", "Spades"};
```

- E.g. an array of Dogs

```
Dog[] my_sled_team = new Dog[3];
my_sled_team[0] = new Dog("Charles", 2, "Charlywawa");
my_sled_team[1] = new Dog("Charlene", 7, "Charlywawa");
my_sled_team[2] = new Dog("Charmaine", 4, "Charlywawa");
```

Multidimensional Array

To access the elements of the **myNumbers** array, we need to **specify two indexes**:
one for the array, and
one for the element inside that array.

This example accesses the third element (2) in the second array (1) of myNumbers:

```
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
int x = myNumbers[1][2];
System.out.println(x);
```

Question: What is the output?

It prints the second row's third column, so output is 7.

Change Element Values

You can also change the value of an element:

Example

```
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
myNumbers[1][2] = 9;
System.out.println(myNumbers[1][2]);

// Outputs 9 instead of 7
```

Loop through an Array

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (int i = 0; i < cars.length; i++) {
System.out.println(cars[i]);
}
```

Loop through a Multidimensional Array

We can also use a for loop inside another for loop to get the elements of a two-dimensional array (we still have to point to the two indexes):

Example

```
public class Main
{
		public static void main(String[] args)
		{
			int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
			for (int i = 0; i < myNumbers.length; ++i)
			{
				for(int j = 0; j < myNumbers[i].length; ++j)
				{
					System.out.println(myNumbers[i][j]);
				}
			}
		}
}
```

Loop through an array using *for-each*

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
System.out.println(i);
}
```

Question: Does this work for int[] days?

You can’t use for (int i : days), but you can use for (Integer i : days).

Homework: Tell why?


Thank you