



# Programming Fundamentals 1

---

Produced by Mr. Dave Drohan ([david.drohan@setu.ie](mailto:david.drohan@setu.ie))  
Dr. Siobhán Drohan  
Ms. Mairead Meagher

Department of Computing & Mathematics  
South East Technological University  
Waterford, Ireland

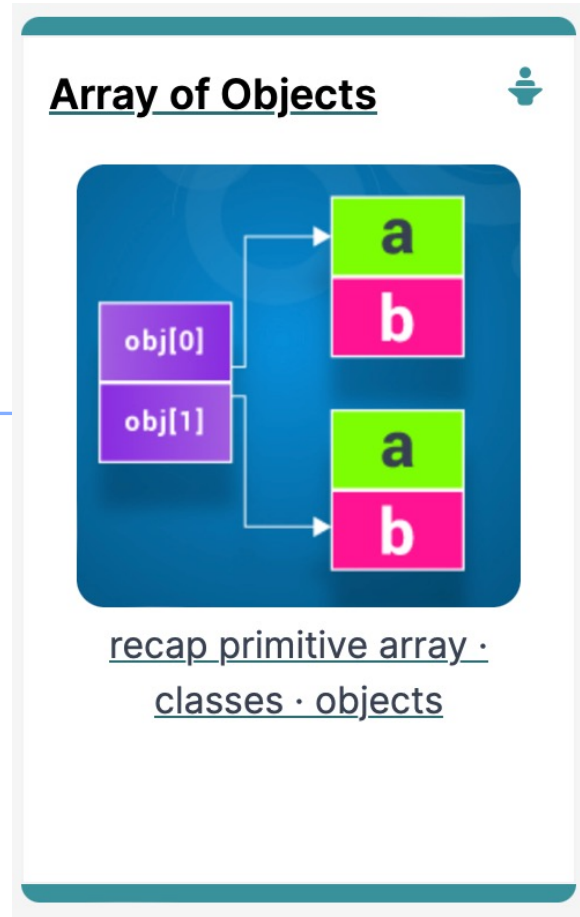
setu.ie





# Arrays, Classes & Objects

A look at arrays of objects





# Agenda

---

- ❑ RECAP : Primitive Arrays
- ❑ String Arrays
- ❑ Arrays of Objects - Product



---

# RECAP : Primitive Arrays





An array can store any type of data.

### Primitive Types

```
int numbers[] = new int[10];
```

```
byte smallNumbers[] = new byte[4];
```

```
char characters[] = new char[26];
```

### Object Types

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

```
Product products[] = new Product[10];
```



# Structure of an **int** primitive array

---

`int[] numbers;`

**numbers**

null



# Structure of an **int** primitive array

---

```
int[] numbers;
```

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

**numbers**



0	0
1	0
2	0
3	0

# Structure of an `int` primitive array

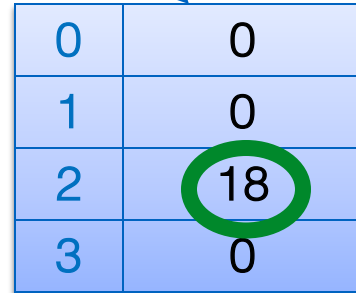
```
int[] numbers;
```

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

```
numbers[2] = 18;
```

We are directly accessing the element at index **2** and setting it to a value of **18**.

`numbers`

A diagram illustrating array access. A blue arrow points from the variable reference box to the third row of a table. The table has two columns: the first column contains indices 0, 1, 2, and 3; the second column contains values 0, 0, 18, and 0. The value 18 is circled in green.

0	0
1	0
2	18
3	0



# Structure of an `int` primitive array

```
int[] numbers;
```

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

```
numbers[2] = 18;
```

```
numbers[0] = 12;
```

We are setting the element at index 0 to a value of 12.

**numbers**



0	12
1	0
2	18
3	0



# Structure of an `int` primitive array

```
int[] numbers;
```

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

```
numbers[2] = 18;
```

```
numbers[0] = 12;
```

```
print(numbers[2]);
```

**numbers**



0	12
1	0
2	18
3	0



Here we are printing the contents of index location 2  
i.e. 18 will be printed to the console.



---

# String Arrays





An array can store any type of data.

### Primitive Types

```
int numbers[] = new int[10];
```

```
byte smallNumbers[] = new byte[4];
```

```
char characters[] = new char[26];
```

### Object Types

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

```
Product products[] = new Product[10];
```





# Structure of a **String** object array

---

```
String[] words;
```

**words**

```
null
```

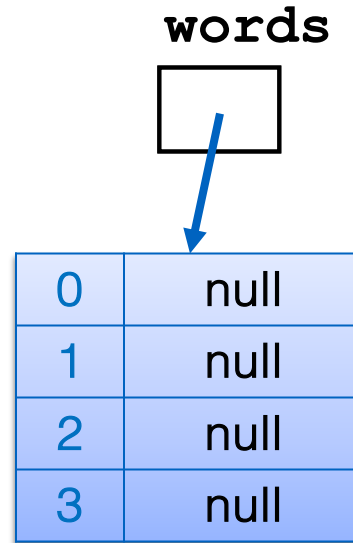


# Structure of a **String** object array

---

```
String[] words;
```

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



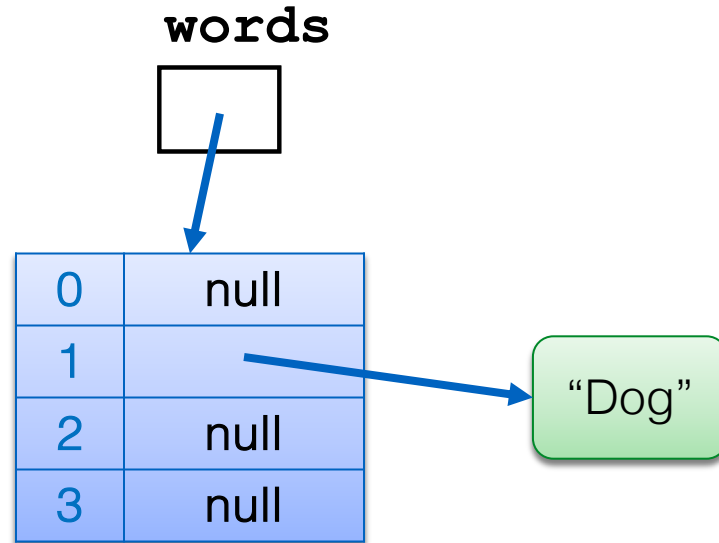


# Structure of a **String** object array

```
String[] words;
```

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

```
words[1] = "Dog";
```



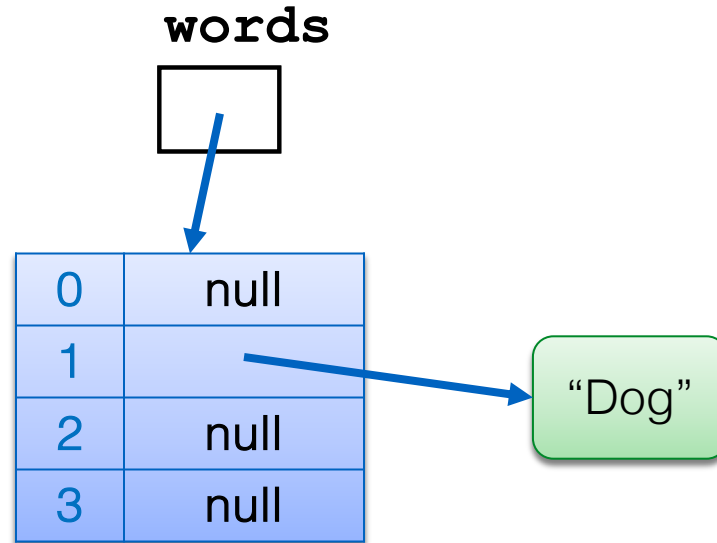
# Structure of a **String** object array

```
String[] words;
```

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

```
words[1] = "Dog";
```

We are directly accessing the element at index 1 and setting it to a value of "Dog".





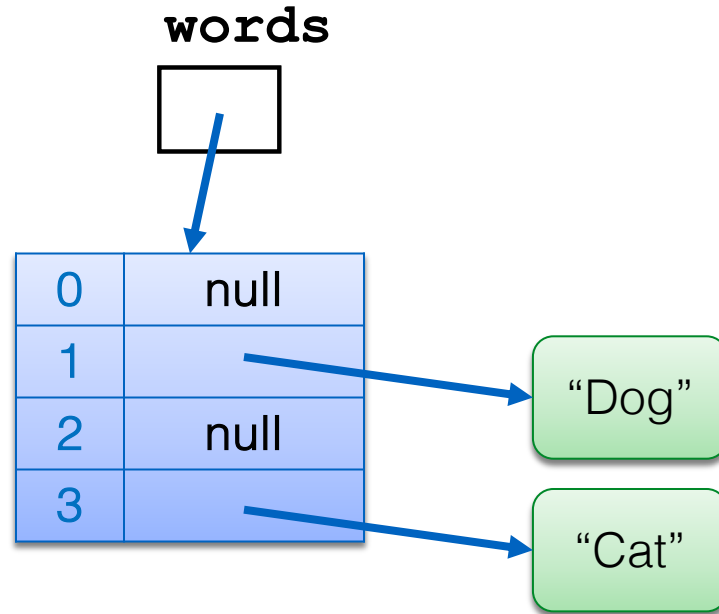
# Structure of a **String** object array

```
String[] words;
```

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

```
words[1] = "Dog";
```

```
words[3] = "Cat";
```



# Structure of a **String** object array

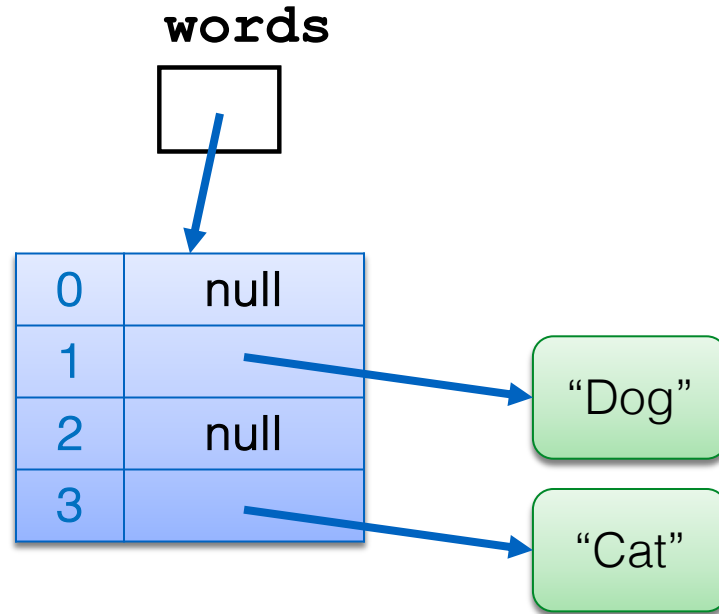
```
String[] words;
```

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

```
words[1] = "Dog";
```

```
words[3] = "Cat";
```

The element at index  
3 is set to "Cat".

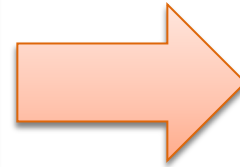




# Structure of a **String** object array

---

```
String words[];  
  
words = new String[4];  
  
words[1] = "Dog";  
words[3] = "Cat";  
  
for (int i=0; i < words.length; i++)  
{  
    System.out.println(words[i]);  
}
```



```
null  
Dog  
null  
Cat
```



---

# Arrays of Objects - Product





An array can store any type of data.

### Primitive Types

```
int numbers[] = new int[10];
```

```
byte smallNumbers[] = new byte[4];
```

```
char characters[] = new char[26];
```

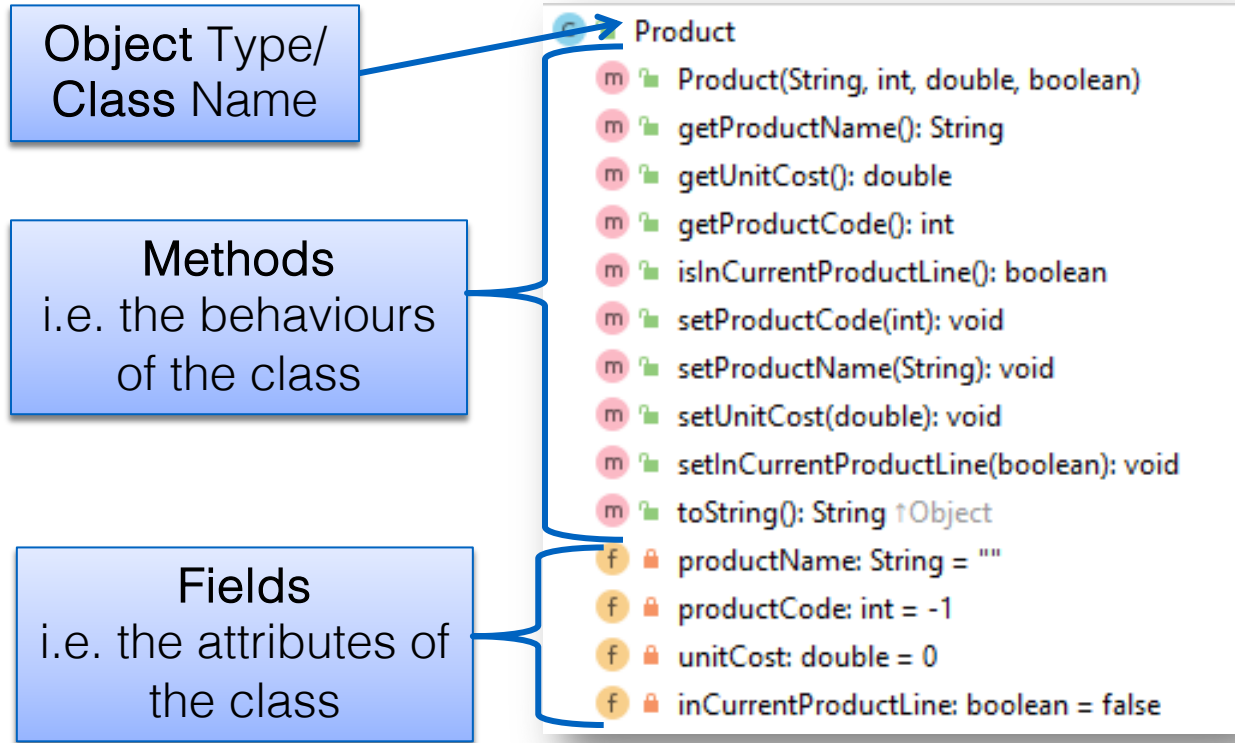
### Object Types

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

```
Product products[] = new Product[10];
```



# Product Class





# Structure of a **Product** Object array

---

```
Product[] products;
```

```
products
```

```
null
```



# Structure of a **Product** Object array

```
Product[] products;
```

```
products = new Product[4];
```

**products**



0	null
1	null
2	null
3	null

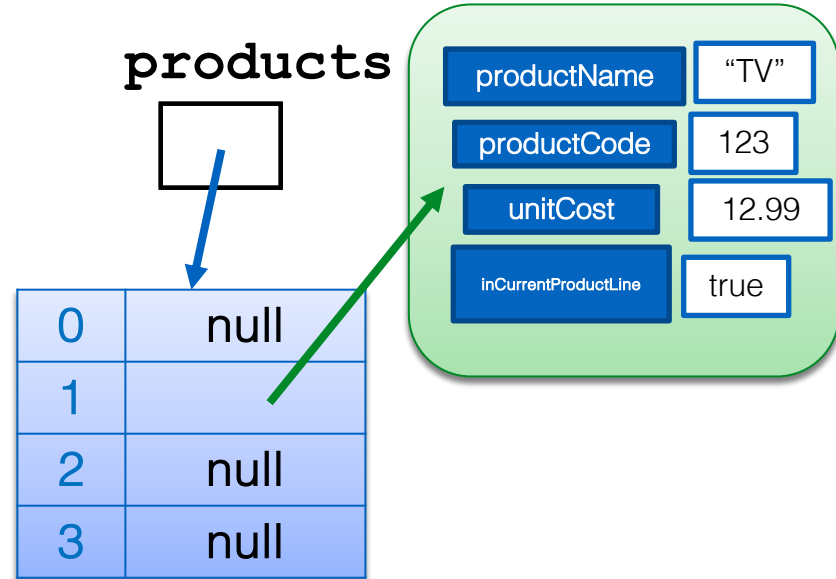




# Structure of a **Product** Object array

```
Product[] products;
```

```
products = new Product[4];
```



```
products[1] = new Product("TV", 123, 12.99, true);
```



# Example using a **Product** Object array

```
public String listProducts() {  
  
    String listOfProducts = "";  
  
    for (int i = 0; i < total; i++) {  
        listOfProducts += i + ": " + products[i].toString() + "\n";  
    }  
  
    return listOfProducts;  
}
```

Returns a String containing all the products stored in the primitive array.

# Questions?

---



Thanks.

