

Introducing groups of classes

Produced Dr. Siobhán Drohan
by: Ms. Maireád Meagher
 Ms. Siobhan Roche

A Shop with one Product isn't really that great, we need to be able to have multiple products



How about an *Array* of Products?

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

**Object Type/
Class Name**

Methods
i.e. the behaviours of
the class

Fields
i.e. the attributes of
the class

```
Product
m Product(String, int, double, boolean)
m getProductName(): String
m getUnitCost(): double
m getProductCode(): int
m isInCurrentProductLine(): boolean
m setProductCode(int): void
m setProductName(String): void
m setUnitCost(double): void
m setInCurrentProductLine(boolean): void
m toString(): String ↑Object
f productName: String = ""
f productCode: int = -1
f unitCost: double = 0
f inCurrentProductLine: boolean = false
```

Structure of a **Product** primitive array

Product[] products;

products

null

Structure of a **Product** primitive array

```
Product[] products;
```

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

products



0	null
1	null
2	null
3	null

Structure of a **Product** primitive array

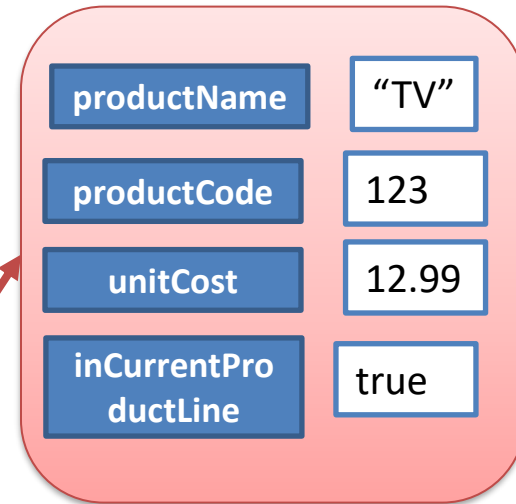
```
Product[] products;
```

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

products



0	null
1	
2	null
3	null



```
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;  
}
```

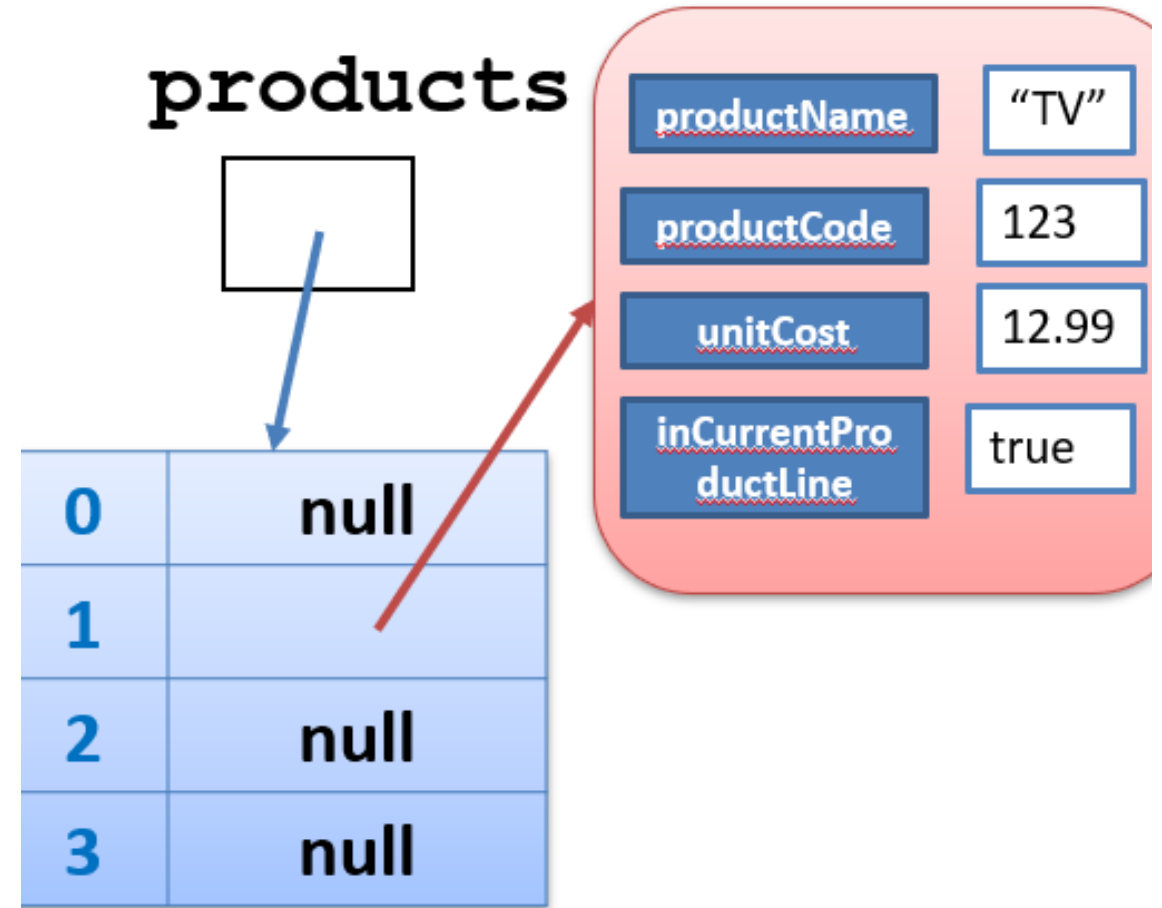
Returns a String containing all the products stored in the primitive array.

Arrays in Java

- Fixed size — must know size in advance
- Stored sequentially in memory
- Useful for simple, fixed collections of data
- Limitations: no automatic resizing, limited built-in methods

Why Not Always Arrays?

- Adding/removing elements is difficult
- No built-in search, sort, or utilities
- Errors occur when guessing array size
- Collections solve these limitations



When to Use an ArrayList

- When the number of items changes over time
- When convenient operations matter
- When modelling real-world lists of objects



Shop Program Evolution

- Current: Driver → Product
- Next: Driver → Store → Product
- Store will manage a collection of Products
- `ArrayList<Product>` is ideal because the number of products can grow

Why ArrayList for Store?

- Store needs to:
 - add products
 - remove products
 - search products
 - list products
- ArrayList provides simple, clean methods to support all of this

Questions?

