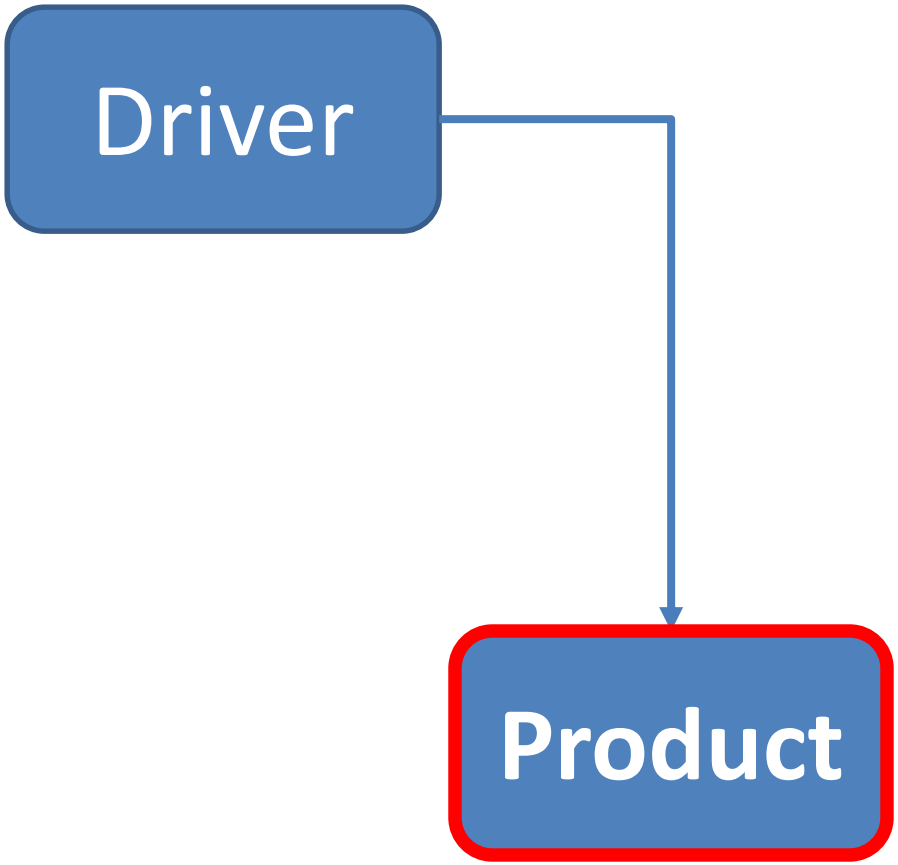
Shop V2.0 - An Array of Product

Produced by: Ms. Maireád Meagher
Dr. Siobhán Drohan

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

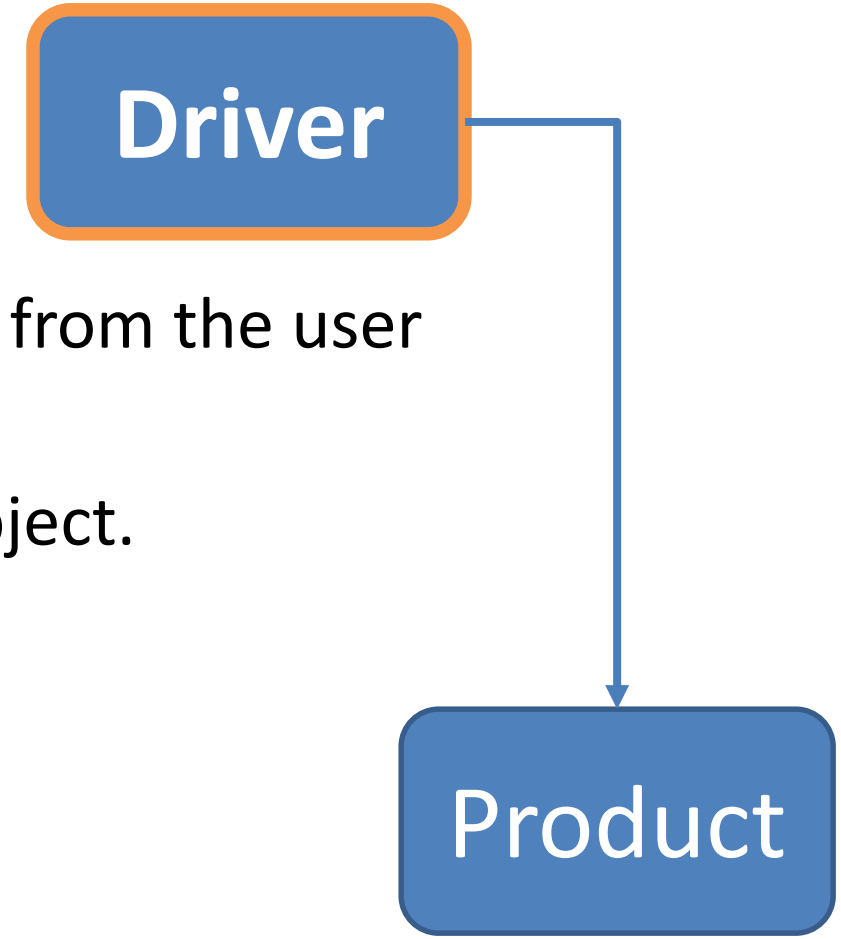
Recap: Shop V1.0 - **Product**



- The **Product** class stores **details** about a product
  - name
  - code
  - unit cost
  - in the current product line or not?

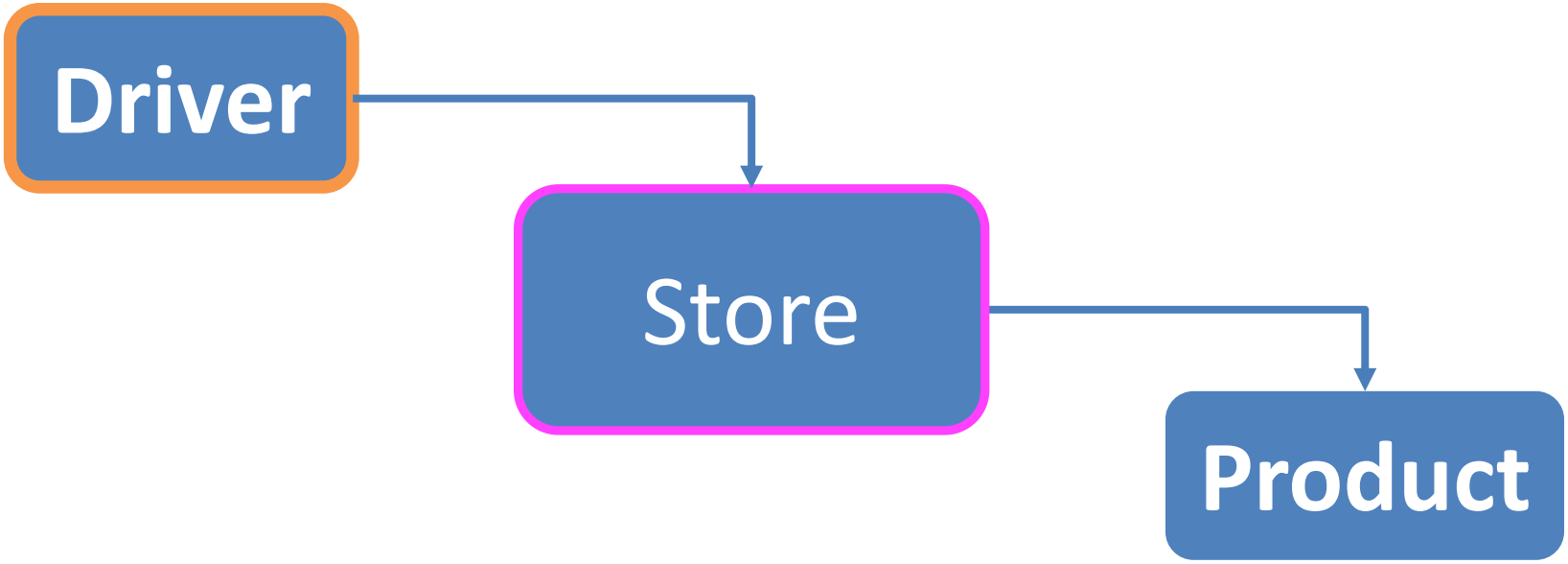
Recap: Shop V1.0 - **Driver**

- The Driver class
  - has the **main()** method.
  - **reads** the product details from the user (via the console)
  - **creates** a new Product object.
  - **prints** the product object (to the console)

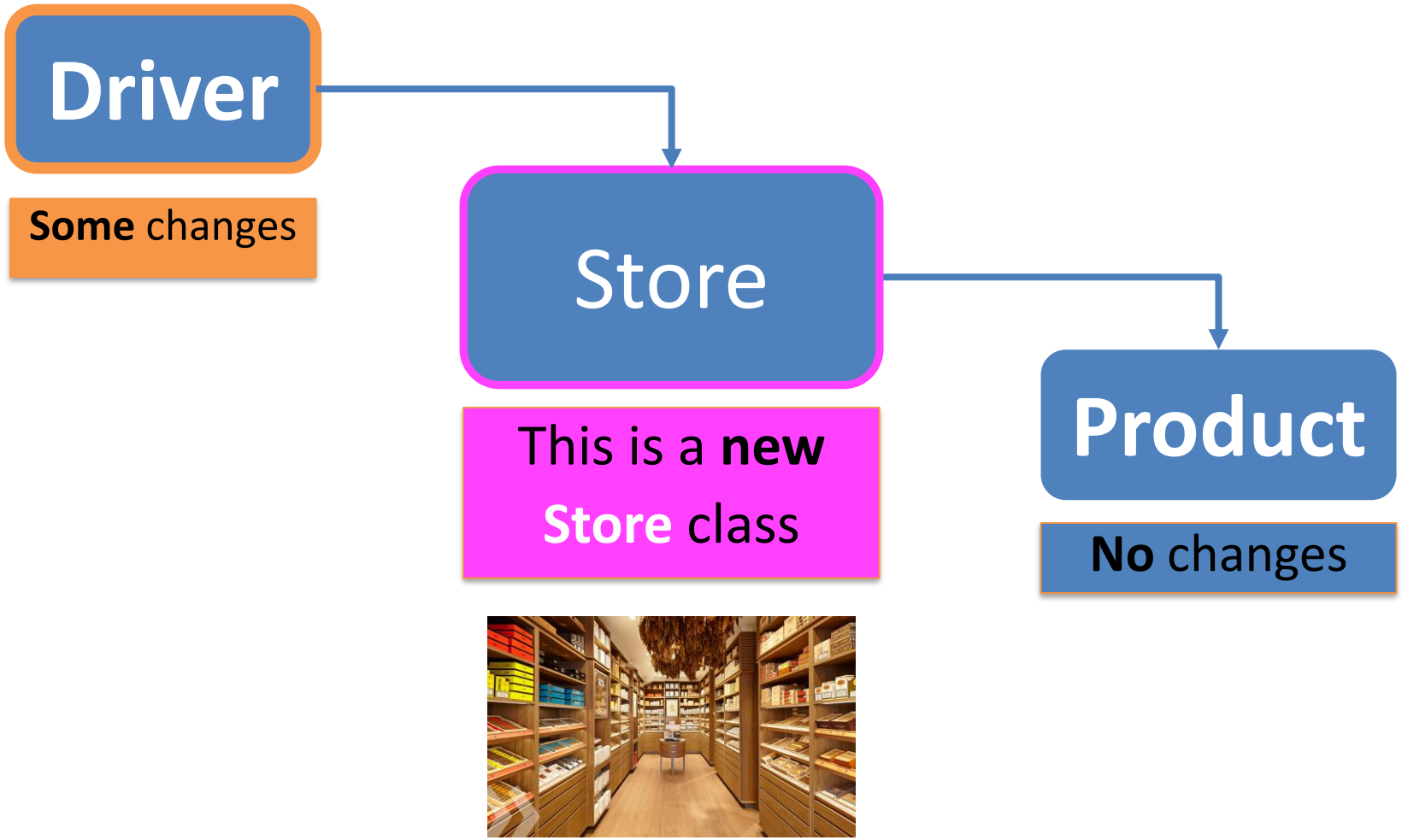


Shop **V2.0**

- New **Store** class is responsible for maintaining a collection of Products
  - i.e. an **array of Products**.
- **Driver** will now allow the user to decide **how many product** details they want to store.



Shop **V2.0** – **changes** to classes



Store – new class

Store

constructor

Store(int)

isFull(): boolean

isEmpty(): boolean

add(Product): boolean

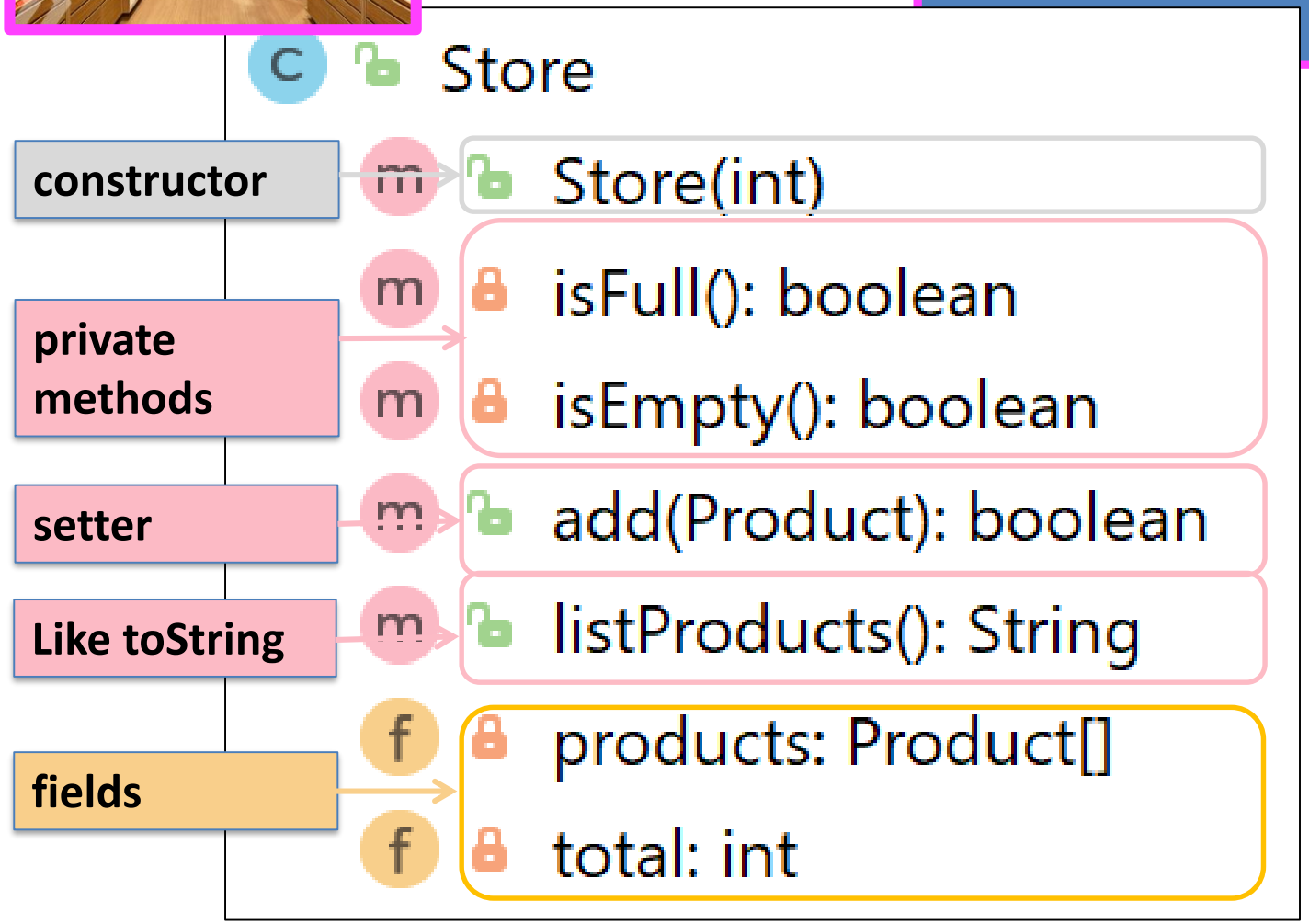
listProducts(): String

fields

products: Product[]

total: int

Store – new class



- Store
  - Store(int)
  - isFull(): boolean
  - isEmpty(): boolean
  - add(Product): boolean
  - listProducts(): String
  - products: Product[]
  - total: int

```
public class Store {

    private Product[] products;
    private int total;

    public Store(int numberItems){
        products = new Product[numberItems];
        total = 0;
    }

    //other methods
}
```

fields

Why private?

- Store
  - Store(int)
  - isFull(): boolean
  - isEmpty(): boolean
  - add(Product): boolean
  - listProducts(): String
  - products: Product[]
  - total: int

```
public class Store {

    private Product[] products;
    private int total;

    public Store(int numberItems){
        products = new Product[numberItems];
        total = 0;
    }

    //other methods
}
```

fields

Why private?

constructor

```
private boolean isFull(){
    return (total == products.length);
}

private boolean isEmpty(){
    return (total == 0);
}

public boolean add(Product product){
    if (isFull()){
        return false;
    }
    else{
        products[total] = product;
        total++;
        return true;
    }
}
```

- Store
  - Store(int)
  - isFull(): boolean
  - isEmpty(): boolean
  - add(Product): boolean
  - listProducts(): String
  - products: Product[]
  - total: int

getters
isFull() & **isEmpty()**
return state of fields.

They are private
member methods

setter
add() makes use of
private method **isFull()**

- Store
  - Store(int)
  - isFull(): boolean
  - isEmpty(): boolean
  - add(Product): boolean
  - listProducts(): String
  - products: Product[]
  - total: int

```java
public String listProducts(){
    if (isEmpty()){
        return "No products";
    }
    else{
        String listOfProducts = "";
        for (int i = 0; i < total; i++){
            listOfProducts += i + ": " + products[i] + "\n";
        }
        return listOfProducts;
    }
}
```

toString type method **listProducts()** makes use of private method **isEmpty()**

Driver

5 changes

4) main() changed to call **processOrder()**

2) addProduct() changed to add the entered product to the array.

5) printProduct() changed to print out all products in the array.

3) New method, **processOrder()**, reads in products from the user.

Driver

- main(String[]): void
- addProduct(): void
- printProduct(): void
- processOrder(): void
- input: Scanner = new Scanner(...)
- store: Store

1) Product object removed and replaced with **Store** object.

Driver

- main(String[]): void
- addProduct(): void
- printProduct(): void
- processOrder(): void
- input: Scanner = new Scanner(...)
- store: Store

Driver

Change - 1

```
import java.util.Scanner;

public class Driver{

    private Scanner input = new Scanner(System.in);
    private Store store;

    //code omitted
}
```

1) Product object removed and replaced with **Store** object.

2) New method, **processOrder()**, reads in products from the user.

Driver

Change - 2

- Driver
  - main(String[]): void
  - addProduct(): void
  - printProduct(): void
  - processOrder(): void
  - input: Scanner = new Scanner(...)
  - store: Store

```
private void processOrder(){
    //find out from the user how many products they would like to order
    System.out.print("How many Products would you like to have in your Store? ");
    int numberProducts = input.nextInt();

    store = new Store(numberProducts);

    //ask the user for the details of the products and add them to the order
    for (int i = 0; i < numberProducts; i++){
        addProduct();
    }
}
```

- Asks how many?
- Pass into Store constructor to initialise an array to that size
- Calls addProduct() for each one

3) main() changed to call **processOrder()**

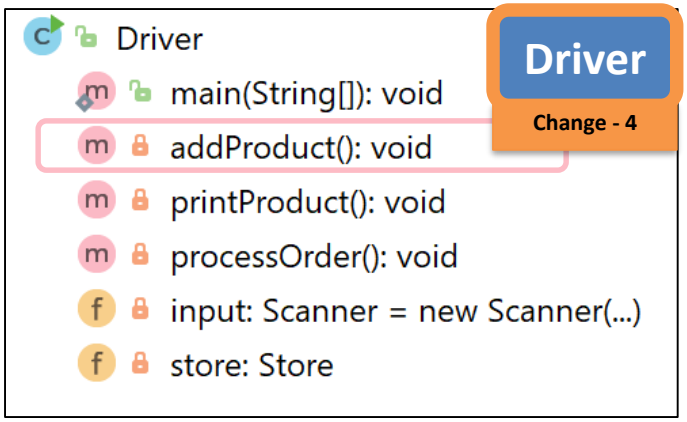
Driver

Change - 3

- Driver
  - main(String[]): void
  - addProduct(): void
  - printProduct(): void
  - processOrder(): void
  - input: Scanner = new Scanner(...)
  - store: Store

```
public static void main(String[] args) {
    Driver c = new Driver();
    c.processOrder();
    c.printProduct();
}
```

4) addProduct() changed to add the entered product to the array.




```java
//gather the product data from the user and create a new product.
private void addProduct(){
    //dummy read of String to clear the buffer - bug in Scanner class.
    input.nextLine();
    System.out.print("Enter the Product Name:  ");
    String productName = input.nextLine();
    System.out.print("Enter the Product Code:  ");
    int productCode = input.nextInt();
    System.out.print("Enter the Unit Cost:  ");
    double unitCost = input.nextDouble();
    System.out.print("Is this product in your current line (y/n): ");
    char currentProduct = input.next().charAt(0);
    boolean inCurrentProductLine = false;
    if ((currentProduct == 'y') || (currentProduct == 'Y'))
        inCurrentProductLine = true;

    store.add(new Product(productName, productCode, unitCost, inCurrentProductLine));
}
```

Read in a **string**

Read in an **int**

Read in an **double**

Read in an **char**

Set **boolean** based on char value

Driver

Change - 5

- Driver
  - main(String[]): void
  - addProduct(): void
  - printProduct(): void
  - processOrder(): void
  - input: Scanner = new Scanner(...)
  - store: Store

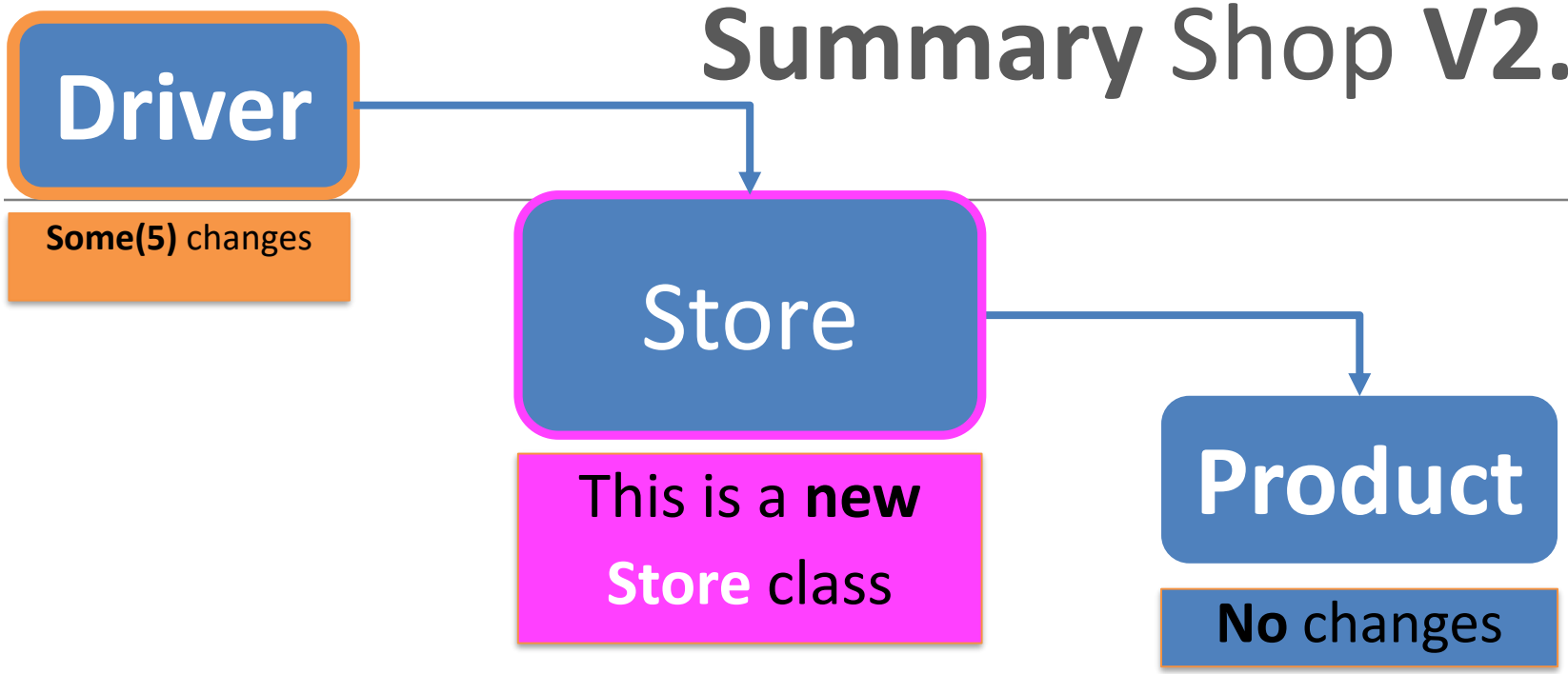
5) printProduct() changed to print out all products in the array.

```
private void printProduct(){

    System.out.println(store.listProducts());

}
```

Summary Shop **V2.0**



- **Store** class maintains a collection of Products i.e. an **array of Products**; store.Products[]

- **Driver** allows the user to decide **how many product** details they want to store. Methods updated to work with this new **store.Products[]** array

Questions?