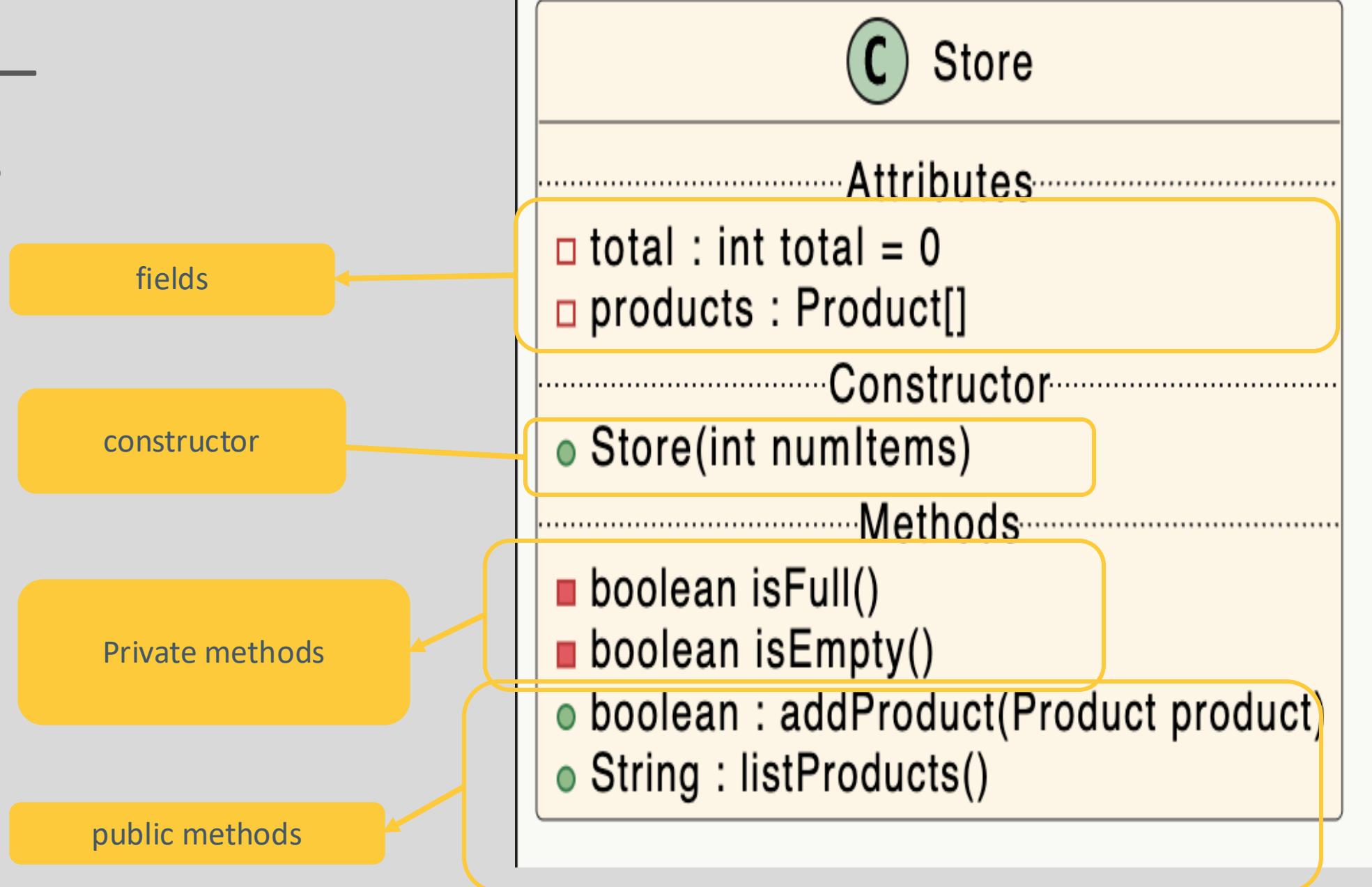


# Shop v2.1 – Store Add more methods

Produced    Ms Siobhan Roche  
by:         Ms Mairead Meagher

# Shop V2.1 – Store Class



# Shop V2.1 – Store – Fields and Constructor

```
public class Store {  
  
    private Product[] products;  
    private int total = 0; //dual purpose.  
                        // 1) number of items stored in array,  
                        // 2) next available index location  
  
    public Store(int numberItems){  
        products = new Product[numberItems];  
    }  
}
```

# Shop V2.1 – Store (private methods)

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

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

## getters

- `isFull()` & `isEmpty()` return state of fields.
- They are private member methods.

# Shop V2.1 – Store (addProduct() Method)

```
public boolean addProduct(Produkt produkt){  
    if (isFull()){  
        return false;  
    }  
    else{  
        products[total] = produkt;  
        total++;  
        return true;  
    }  
}
```

- addProduct() makes use of private method isFull()

# Shop V2.1 – Store (listProducts())

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

list

- `toString` 'type' method  
`listProducts()` makes use of private  
method `isEmpty()`

Product Class

```
public String toString()
{
    String yesOrNo;
    if (inCurrentProductLine)
        yesOrNo = "T";
    else yesOrNo = "F";
    return "Product description: " + productName
           + ", product code: " + productCode
           + ", unit cost: " + unitCost
           + ", currently in product line: " + yesOrNo;
}
```

# Shop V2.1 – Store (listProductsAbovePrice(..))

```
public String listProductsAboveAPrice(double price) {  
    if (isEmpty()) {  
        return "No Products in the store";  
    } else {  
        String str = "";  
        for (int i = 0; i < total; i++) {  
            if (products[i].getUnitCost() > price)  
                str += i + ": " + products[i] + "\n";  
        }  
        if (str.equals("")) {  
            return "No products are more expensive than: " + price;  
        } else {  
            return str;  
        }  
    }  
}
```

Product Class

```
public double getUnitCost(){  
    return unitCost;  
}
```

# Shop V2.1 – Store (cheapestProduct())

```
public Product cheapestProduct() {  
    if (!isEmpty()) {  
        Product cheapestProduct = products[0];  
        for (int i = 1; i < total; i++) {  
            if (products[i].getUnitCost() < cheapestProduct.getUnitCost())  
                cheapestProduct = products[i];  
        }  
        return cheapestProduct;  
    } else {  
        return null;  
    }  
}
```

Product Class

```
~,  
public double getUnitCost(){  
    return unitCost;  
}
```

# Shop V2.1 – Store (listCurrentProducts())

```
public String listCurrentProducts() {  
    if (isEmpty()) {  
        return "No Products in the store";  
    } else {  
        String listOfProducts = "";  
        for (int i = 0; i < total; i++) {  
            if (products[i].isInCurrentProductLine())  
                listOfProducts += i + ": " + products[i] + "\n";  
        }  
        if (listOfProducts.equals("")){  
            return "No Products are in our current product line";  
        }  
        else{  
            return listOfProducts;  
        }  
    }  
}
```

Product Class

```
public boolean isInCurrentProductLine() {  
    return inCurrentProductLine;  
}
```

# Shop V2.1 – Store (averageProductPrice())

```
public double averageProductPrice() {  
    if (!isEmpty()) {  
        double totalPrice = 0;  
        for (int i = 0; i < total; i++) {  
            totalPrice += products[i].getUnitCost();  
        }  
        return totalPrice / total;  
    } else {  
        return -1;  
    }  
}
```

Product Class

```
public double getUnitCost(){  
    return unitCost;  
}
```

# Shop V2.1 – Driver

## C Driver

### Attributes

- store : Store
- input : Scanner = new Scanner(System.int)

### Constructor

- Store(int numItems)

### Methods

- void : main(String[] )
- void : processOrder()
- void : addProduct()
- void : printProducts()
- void : printCurrentProducts()
- void : printAverageProductPrice()
- void : printCheapestProduct()
- void : printProductsAboveAPrice()

# Questions?



