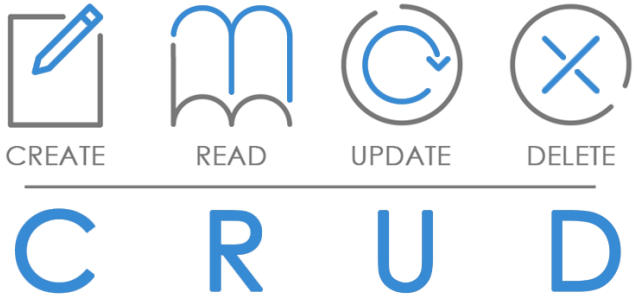


Persistence

An Introduction to the CRUD Process

Produced by:

Dr. Siobhán Drohan
Ms. Maireád Meagher
Ms Siobhan Roche

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

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

Topic List

1. What is CRUD?
2. Recap of Shop V3.0
3. Shop V4.0 (Driver.java):
   - revised menu (making it CRUD compliant)

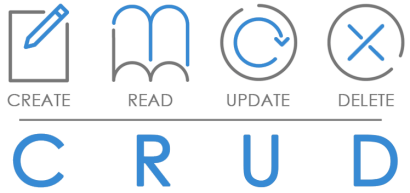
recap of case 1 (add a product)

recap of case 2 (list a product)

coding case 4 (delete a product)

coding case 3 (update a product)

CRUD



The four basic functions of **persistent storage**:

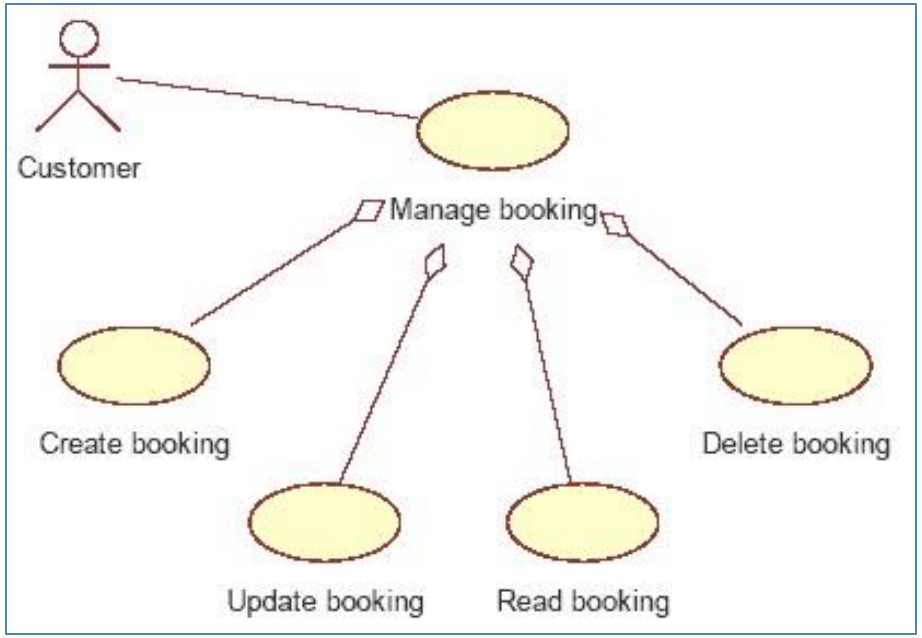
- Create or add new objects

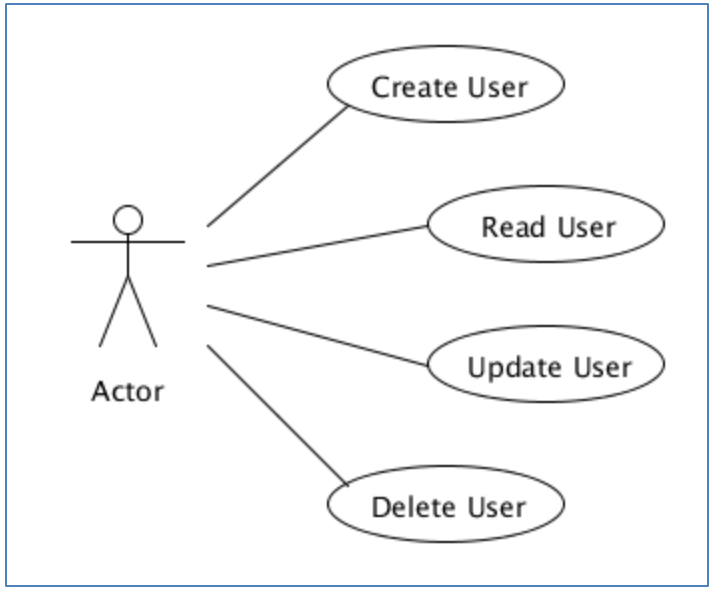
- Read, retrieve or search for existing objects

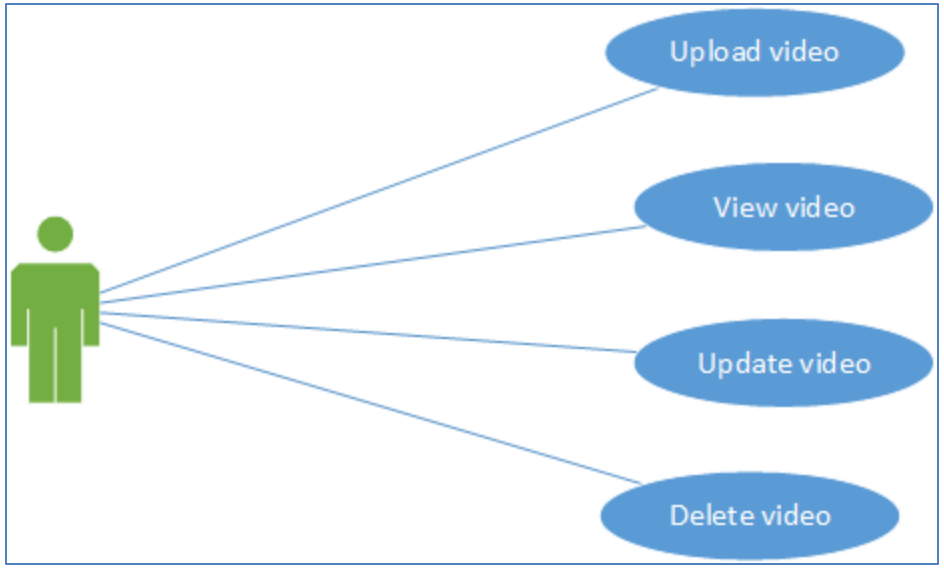
- Update or edit existing objects

- Delete existing objects

CRUD Examples







Topic List

1. What is CRUD?
2. Recap of Shop V3.0
3. Shop V4.0 (Driver.java):
   – revised menu (making it CRUD compliant)

recap of case 1 (add a product)

recap of case 2 (list a product)

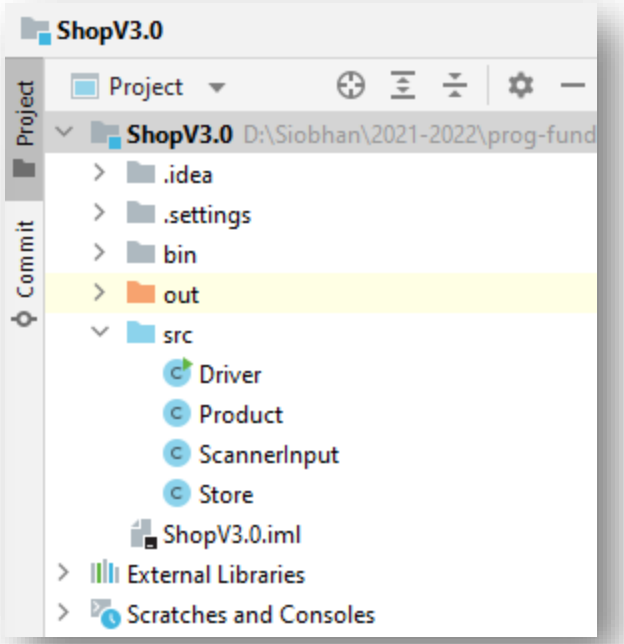
coding case 4 (delete a product)

coding case 3 (update a product)

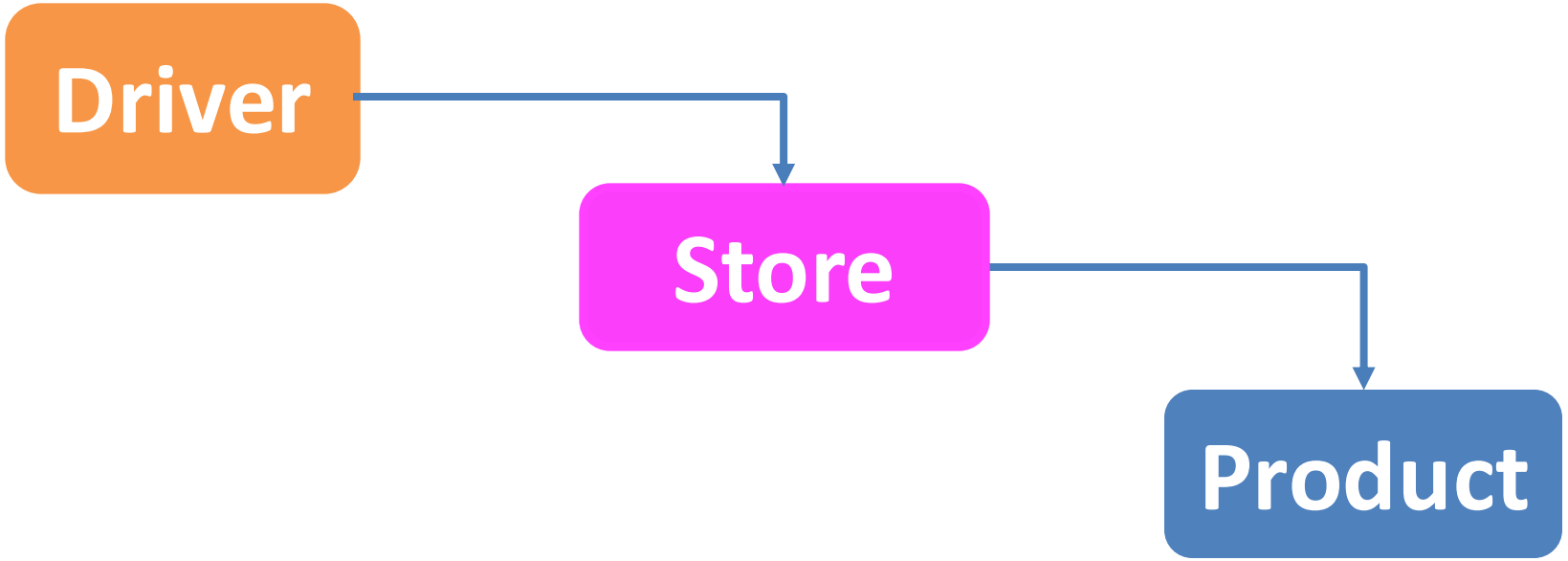
Shop

Uses an **ArrayList of Products** to store the details.

```
Shop Menu
---------
   1) Add a product
   2) List the Products
   ----------------------------
   3) List the current products
   4) Display average product unit cost
   5) Display cheapest product
   6) List products that are more expensive than a given price
   ----------------------------
   0) Exit
==>>
```



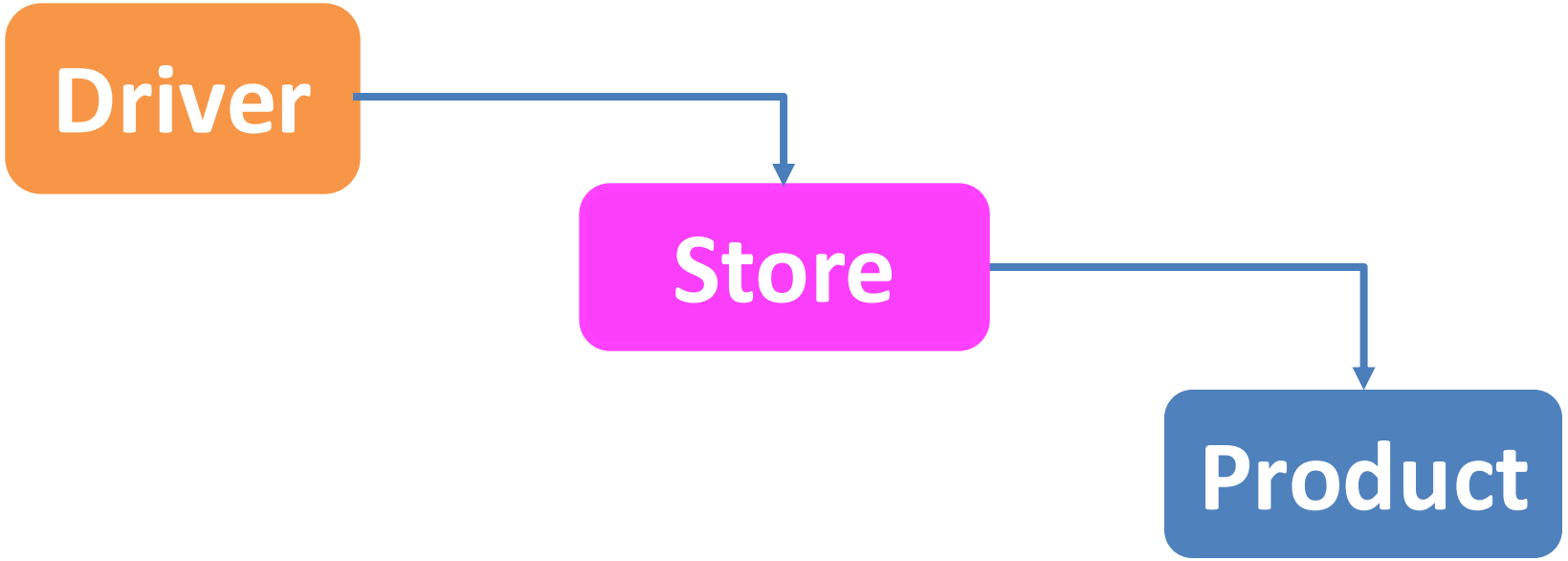
RECAP: Shop **V3.0**



Product class

- Four instance fields
  - product's name, code, unit cost, is in the current product line or not.
- Basic class with Constructors, Getters, Setters and toString methods

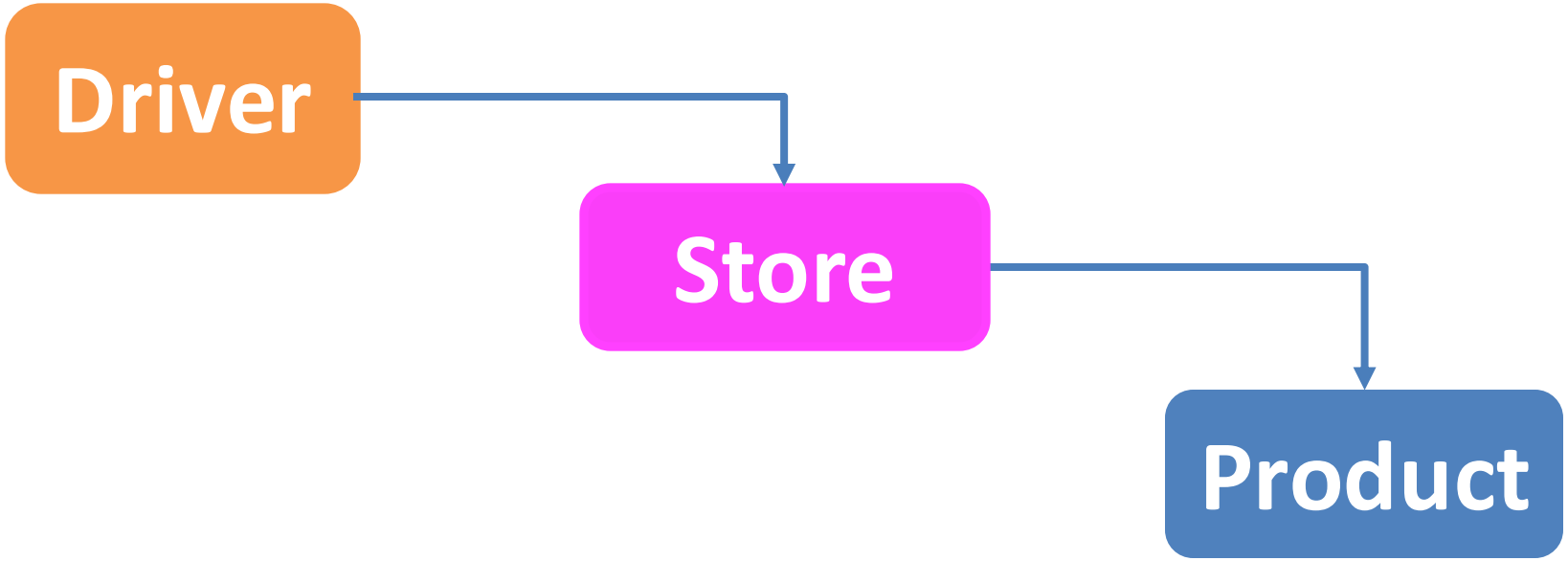
RECAP: Shop **V3.0**



Store class

- One instance field, **products** (an ***ArrayList of Product***).
- Many additional methods
  - listProducts(), cheapestProduct(), listCurrentProducts(), etc.

RECAP: Shop **V3.0**



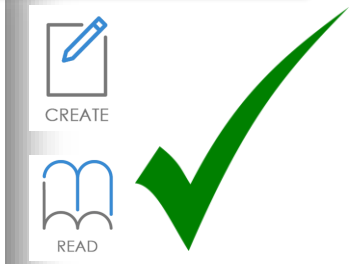
Driver

- Contains the **main()** method
- Runs the **menu**
- Negotiates with the user (i.e. handles **I/O** using the ScannerInput class**)**

RECAP: Shop **V3.0**

Add Product: Menu Option 1.
Read a Product(s): Menu Options 2 - 6.

```
Shop Menu
---------
   1) Add a product
   2) List the Products
   ----------------------------
   3) List the current products
   4) Display average product unit cost
   5) Display cheapest product
   6) List products that are more expensive than a given price
   ----------------------------
   0) Exit
==>>
```



The menu has NO Update or Delete!

Topic List

1. What is CRUD?
2. Recap of Shop V3.0
3. Shop V4.0 (Driver.java):
   – revised menu (making it CRUD compliant)

recap of case 1 (add a product)

recap of case 2 (list a product)

coding case 4 (delete a product)

coding case 3 (update a product)

Shop V4.0 – Revised Menu

```
Shop Menu
---------
   1) Add a product
   2) List the Products
   3) Update a Product
   4) Delete a Product
   ----------------------------
   5) List the current products
   6) Display average product unit cost
   7) Display cheapest product
   8) List products that are more expensive than a given price
   ----------------------------
   0) Exit
==>>
```

Option 1 – Create a Product
Option 2 – Read products
Option 3 – Update a product
Option 4 – Delete a product

Shop V4.0 – Revised Menu

```
private int mainMenu(){
    int option = ScannerInput.readNextInt( prompt: """
            Shop Menu
            ---------
               1) Add a product
               2) List the Products
               3) Update a Product
               4) Delete a Product
               ---------------------------
               5) List the current products
               6) Display average product unit cost
               7) Display cheapest product
               8) List products that are more expensive than a given price
               ---------------------------
               0) Exit
            ==>> """);
    return option;
}
```

Now we need to update the **switch** to:

- add code for case 3 (update) and 4 (delete) to Driver.java
- move the current options for 3-6 to be 5-8.

Shop V4.0 – Revised Menu

```
switch (option){
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> printCurrentProducts();
    case 6 -> printAverageProductPrice();
    case 7 -> printCheapestProduct();
    case 8 -> printProductsAboveAPrice();
    default -> System.out.println("Invalid option entered: " + option);
}
```

In Driver.java, we have provided a case 3 and 4, but we still need to write the actual methods:

- updateProduct
- deleteProduct

We have moved the options for 3-6 to 5-8.

Topic List

1. What is CRUD?
2. Recap of Shop V3.0
3. Shop V4.0 (Driver.java):
   – revised menu (making it CRUD compliant)

recap of case 1 (add a product)

recap of case 2 (list a product)

coding case 4 (delete a product)

coding case 3 (update a product)

Driver.java code

```java
switch (option){
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> printCurrentProducts();
    case 6 -> printAverageProductPrice();
    case 7 -> printCheapestProduct();
    case 8 -> printProductsAboveAPrice();
```

```java
//gather the product data from the user and create a new product object - add it to the co
private void addProduct(){

    String productName = ScannerInput.readNextLine( prompt: "Enter the Product Name:  ");
    int productCode = ScannerInput.readNextInt( prompt: "Enter the Product Code:  ");
    double unitCost = ScannerInput.readNextDouble( prompt: "Enter the Unit Cost:  ");

    //Ask the user to type in either a Y or an N.  This is then
    //converted to either a True or a False (i.e. a boolean value).
    char currentProduct = ScannerInput.readNextChar( prompt: "Is this product in your current line (y/n): ");
    boolean inCurrentProductLine = false;
    if ((currentProduct == 'y') || (currentProduct == 'Y'))
        inCurrentProductLine = true;

    boolean isAdded = store.add(new Product(productName, productCode, unitCost, inCurrentProductLine));
    if (isAdded){
        System.out.println("Product Added Successfully");
    }
    else{
        System.out.println("No Product Added");
    }
}
```

Driver.java code

```java
switch (option){
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> printCurrentProducts();
    case 6 -> printAverageProductPrice();
    case 7 -> printCheapestProduct();
    case 8 -> printProductsAboveAPrice();
```

```java
//gather the product data from the user and create a new product object - add it to the co
private void addProduct(){

    String productName = ScannerInput.readNextLine( prompt: "Enter the Product Name:  ");
    int productCode = ScannerInput.readNextInt( prompt: "Enter the Product Code:  ");
    double unitCost = ScannerInput.readNextDouble( prompt: "Enter the Unit Cost:  ");

    //Ask the user to type in either a Y or an N.  This is then
    //converted to either a True or a False (i.e. a boolean value).
    char currentProduct = ScannerInput.readNextChar( prompt: "Is this product in your current line (y/n): ");
    boolean inCurrentProductLine = false;
    if ((currentProduct == 'y') || (currentProduct == 'Y'))
        inCurrentProductLine = true;

    boolean isAdded = store.add(new Product(productName, productCode, unitCost, inCurrentProductLine));
    if (isAdded){
        System.out.println("Product Added Successfully");
    }
    else{
        System.out.println("No Product Added");
    }
}
```

Store.java code

```java
public boolean add (Product product){
    return products.add (product);
}
```

Topic List

1. What is CRUD?
2. Recap of Shop V3.0
3. Shop V4.0 (Driver.java):
   – revised menu (making it CRUD compliant)

recap of case 1 (add a product)

recap of case 2 (list a product)

coding case 4 (delete a product)

coding case 3 (update a product)

Driver.java code

```java
//print the products stored in the collection
private void printProducts(){
    System.out.println("List of Products are:");
    System.out.println(store.listProducts());
}
```

```java
switch (option){
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> printCurrentProducts();
    case 6 -> printAverageProductPrice();
    case 7 -> printCheapestProduct();
    case 8 -> printProductsAboveAPrice();
```

Store.java code

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

Driver.java code

```java
//print the products stored in the collection
private void printProducts(){
    System.out.println("List of Products are:");
    System.out.println(store.listProducts());
}
```

```java
switch (option){
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> printCurrentProducts();
    case 6 -> printAverageProductPrice();
    case 7 -> printCheapestProduct();
    case 8 -> printProductsAboveAPrice();
```

Store.java code

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

Sample Output:

```
==>> 2
List of Products are:
0: Product description: tv, product code: 1234, unit cost: 349.99, currently in product line: true
1: Product description: phone, product code: 2345, unit cost: 299.99, currently in product line: true
2: Product description: amazon echo, product code: 4543, unit cost: 89.0, currently in product line: false
```

Topic List

1. What is CRUD?
2. Recap of Shop V3.0
3. Shop V4.0 (Driver.java):
   – revised menu (making it CRUD compliant)

recap of case 1 (add a product)

recap of case 2 (list a product)

coding case 4 (delete a product)

coding case 3 (update a product)

```java
switch (option){
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> printCurrentProducts();
    case 6 -> printAverageProductPrice();
    case 7 -> printCheapestProduct();
    case 8 -> printProductsAboveAPrice();
```

```java
//ask the user to enter the index of the object to delete, and assuming it's valid, delete it.
private void deleteProduct(){
    printProducts();
    if (store.numberOfProducts() > 0){
        //only ask the user to choose the product to delete if products exist
        int indexToDelete = ScannerInput.readNextInt( prompt: "Enter the index of the product to delete ==> ");
        //pass the index of the product to Store for deleting and check for success.
        Product productToDelete = store.deleteProduct(indexToDelete);
        if (productToDelete != null){
            System.out.println("Delete Successful! Deleted product: " + productToDelete.getProductName());
        }
        else{
            System.out.println("Delete NOT Successful");
        }
    }
}
```

Driver.java code

Validation: Checks products are in the ArrayList before asking the user for an index number.

```
switch (option){
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> printCurrentProducts();
    case 6 -> printAverageProductPrice();
    case 7 -> printCheapestProduct();
    case 8 -> printProductsAboveAPrice();
```

```
//ask the user to enter the index of the object to delete, and assuming it's valid, delete it.
private void deleteProduct(){
    printProducts();
    if (store.numberOfProducts() > 0){
        //only ask the user to choose the product to delete if products exist
        int indexToDelete = ScannerInput.readNextInt( prompt: "Enter the index of the product to delete ==> ");
        //pass the index of the product to Store for deleting and check for success.
        Product productToDelete = store.deleteProduct(indexToDelete);
        if (productToDelete != null){
            System.out.println("Delete Successful! Deleted product: " + productToDelete.getProductName());
        }
        else{
            System.out.println("Delete NOT Successful");
        }
    }
}
```

Driver.java code

Validation: Checks It checks whether the delete was successful or not.

```
switch (option){
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> printCurrentProducts();
    case 6 -> printAverageProductPrice();
    case 7 -> printCheapestProduct();
    case 8 -> printProductsAboveAPrice();
```

```
//ask the user to enter the index of the object to delete, and assuming it's valid, delete it.
private void deleteProduct(){
    printProducts();
    if (store.numberOfProducts() > 0){
        //only ask the user to choose the product to delete if products exist
        int indexToDelete = ScannerInput.readNextInt( prompt: "Enter the index of the product to delete ==> ");
        //pass the index of the product to Store for deleting and check for success.
        Product productToDelete = store.deleteProduct(indexToDelete);
        if (productToDelete != null){
            System.out.println("Delete Successful! Deleted product: " + productToDelete.getProductName());
        }
        else{
            System.out.println("Delete NOT Successful");
        }
    }
}
```

Driver.java code

The isValidIndex method checks to see if the index (passed as a parameter) is valid i.e. it is greater than zero and less than the size of the products ArrayList.

```java
public boolean isValidIndex(int index) {
    return (index >= 0) && (index < products.size());
}
```

Store.java code

```java
switch (option){
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> printCurrentProducts();
    case 6 -> printAverageProductPrice();
    case 7 -> printCheapestProduct();
    case 8 -> printProductsAboveAPrice();
```

If indexToDelete is a valid index in the products ArrayList, this method removes the product at that location from the ArrayList. The removed product is returned to Driver so it can be printed to the user.

Null is returned if indexToDelete is invalid.

```java
public Product deleteProduct(int indexToDelete) {
    if (isValidIndex(indexToDelete)) {
        return products.remove(indexToDelete);
    }
    return null;
}
```

Store.java code

Topic List

1. What is CRUD?
2. Recap of Shop V3.0
3. Shop V4.0 (Driver.java):
   – revised menu (making it CRUD compliant)

recap of case 1 (add a product)

recap of case 2 (list a product)

coding case 4 (delete a product)

coding case 3 (update a product)

Driver.java

```java
private void updateProduct(){
    printProducts();
    if (store.numberOfProducts() > 0){
        //only ask the user to choose the product to update if products exist
        int indexToUpdate = ScannerInput.readNextInt( prompt: "Enter the index of the product to update ==> ");
        if (store.isValidIndex(indexToUpdate)){
            String productName = ScannerInput.readNextLine( prompt: "Enter the Product Name:  ");
            int productCode = ScannerInput.readNextInt( prompt: "Enter the Product Code:  ");
            double unitCost = ScannerInput.readNextDouble( prompt: "Enter the Unit Cost:  ");

            //Ask the user to type in either a Y or an N.  This is then
            //converted to either a True or a False (i.e. a boolean value).
            char currentProduct = ScannerInput.readNextChar( prompt: "Is this product in your current line (y/n): ");
            boolean inCurrentProductLine = false;
            if ((currentProduct == 'y') || (currentProduct == 'Y'))
                inCurrentProductLine = true;

            //pass the index of the product and the new product details to Store for updating and check for success.
            if (store.updateProduct(indexToUpdate, new Product(productName, productCode, unitCost, inCurrentProductLine))){
                System.out.println("Update Successful");
            }
            else{
                System.out.println("Update NOT Successful");
            }
        }
        else{
            System.out.println("There are no products for this index number");
        }
    }
}
```

```java
switch (option){
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> printCurrentProducts();
    case 6 -> printAverageProductPrice();
    case 7 -> printCheapestProduct();
    case 8 -> printProductsAboveAPrice();
```

Driver.java

```java
private void updateProduct(){
    printProducts();
    if (store.numberOfProducts() > 0){
        //only ask the user to choose the product to update if products exist
        int indexToUpdate = ScannerInput.readNextInt( prompt: "Enter the index of the product to update ==> ");
        if (store.isValidIndex(indexToUpdate)){
            String productName = ScannerInput.readNextLine( prompt: "Enter the Product Name:  ");
            int productCode = ScannerInput.readNextInt( prompt: "Enter the Product Code:  ");
            double unitCost = ScannerInput.readNextDouble( prompt: "Enter the Unit Cost:  ");

            //Ask the user to type in either a Y or an N.  This is then
            //converted to either a True or a False (i.e. a boolean value).
            char currentProduct = ScannerInput.readNextChar( prompt: "Is this product in your current line (y/n): ");
            boolean inCurrentProductLine = false;
            if ((currentProduct == 'y') || (currentProduct == 'Y'))
                inCurrentProductLine = true;

            //pass the index of the product and the new product details to Store for updating and check for success.
            if (store.updateProduct(indexToUpdate, new Product(productName, productCode, unitCost, inCurrentProductLine))){
                System.out.println("Update Successful");
            }
            else{
                System.out.println("Update NOT Successful");
            }
        }
        else{
            System.out.println("There are no products for this index number");
        }
    }
}
```

Validation: Makes sure products exist before asking user for any update details.

```java
switch (option){
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> printCurrentProducts();
    case 6 -> printAverageProductPrice();
    case 7 -> printCheapestProduct();
    case 8 -> printProductsAboveAPrice();
```

Driver.java

```java
private void updateProduct(){
    printProducts();
    if (store.numberOfProducts() > 0){
        //only ask the user to choose the product to update if products exist
        int indexToUpdate = ScannerInput.readNextInt( prompt: "Enter the index of the product to update ==> ");
        if (store.isValidIndex(indexToUpdate)){
            String productName = ScannerInput.readNextLine( prompt: "Enter the Product Name:  ");
            int productCode = ScannerInput.readNextInt( prompt: "Enter the Product Code:  ");
            double unitCost = ScannerInput.readNextDouble( prompt: "Enter the Unit Cost:  ");

            //Ask the user to type in either a Y or an N.  This is then
            //converted to either a True or a False (i.e. a boolean value).
            char currentProduct = ScannerInput.readNextChar( prompt: "Is this product in your current line (y/n): "
            boolean inCurrentProductLine = false;
            if ((currentProduct == 'y') || (currentProduct == 'Y'))
                inCurrentProductLine = true;

            //pass the index of the product and the new product details to Store for updating and check for success.
            if (store.updateProduct(indexToUpdate, new Product(productName, productCode, unitCost, inCurrentProductLine))){
                System.out.println("Update Successful");
            }
            else{
                System.out.println("Update NOT Successful");
            }
        }
        else{
            System.out.println("There are no products for this index number");
        }
    }
}
```

Validation: Makes sure the index number entered by the user is an index in the products ArrayList…

```java
switch (option){
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> printCurrentProducts();
    case 6 -> printAverageProductPrice();
    case 7 -> printCheapestProduct();
    case 8 -> printProductsAboveAPrice();
```

Driver.java

```java
private void updateProduct(){
    printProducts();
    if (store.numberOfProducts() > 0){
        //only ask the user to choose the product to update if products exist
        int indexToUpdate = ScannerInput.readNextInt( prompt: "Enter the index of the product to update ==> ");
        if (store.isValidIndex(indexToUpdate)){
            String productName = ScannerInput.readNextLine( prompt: "Enter the Product Name:  ");
            int productCode = ScannerInput.readNextInt( prompt: "Enter the Product Code:  ");
            double unitCost = ScannerInput.readNextDouble( prompt: "Enter the Unit Cost:  ");

            //Ask the user to type in either a Y or an N.  This is then
            //converted to either a True or a False (i.e. a boolean value).
            char currentProduct = ScannerInput.readNextChar( prompt: "Is this product in your current line (y/n): "
            boolean inCurrentProductLine = false;
            if ((currentProduct == 'y') || (currentProduct == 'Y'))
                inCurrentProductLine = true;

            //pass the index of the product and the new product details to Store for updating and check for success.
            if (store.updateProduct(indexToUpdate, new Product(productName, productCode, unitCost, inCurrentProductLine))){
                System.out.println("Update Successful");
            }
            else{
                System.out.println("Update NOT Successful");
            }
        }
        else{
            System.out.println("There are no products for this index number");
        }
    }
}
```

Validation: If the update failed, the user is informed of it, otherwise they get a successful message.

```java
switch (option){
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> printCurrentProducts();
    case 6 -> printAverageProductPrice();
    case 7 -> printCheapestProduct();
    case 8 -> printProductsAboveAPrice();
```

Store.java

```java
public boolean updateProduct(int indexToUpdate, Product updateDetails) {
    //find the product object by the index number
    Product foundProduct = findProduct(indexToUpdate);

    //if the product exists, use the details passed in the updateDetails parameter to
    //update the found product in the ArrayList.
    if (foundProduct != null) {
        foundProduct.setProductName(updateDetails.getProductName());
        foundProduct.setProductCode(updateDetails.getProductCode());
        foundProduct.setUnitCost(updateDetails.getUnitCost());
        foundProduct.setInCurrentProductLine(updateDetails.isInCurrentProductLine());
        return true;
    }

    //if the product was not found, return false, indicating that the update was not successful
    return false;
}
```

Store.java

```java
public Product findProduct(int index) {
    if (isValidIndex(index)) {
        return products.get(index);
    }
    return null;
}
```

```java
switch (option){
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> printCurrentProducts();
    case 6 -> printAverageProductPrice();
    case 7 -> printCheapestProduct();
    case 8 -> printProductsAboveAPrice();
```

Any Questions?