

# Persistence

## An Introduction to the CRUD Process

---

Produced by: Dr. Siobhán Drohan



Waterford Institute *of* Technology  
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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

# Topic List

---

- What is CRUD?
- Shop V2.0 – a recap on **CRUD**
- Shop V3.0 – adding Update (CR**U**D)
- DRY (Don't Repeat Yourself)

# CRUD

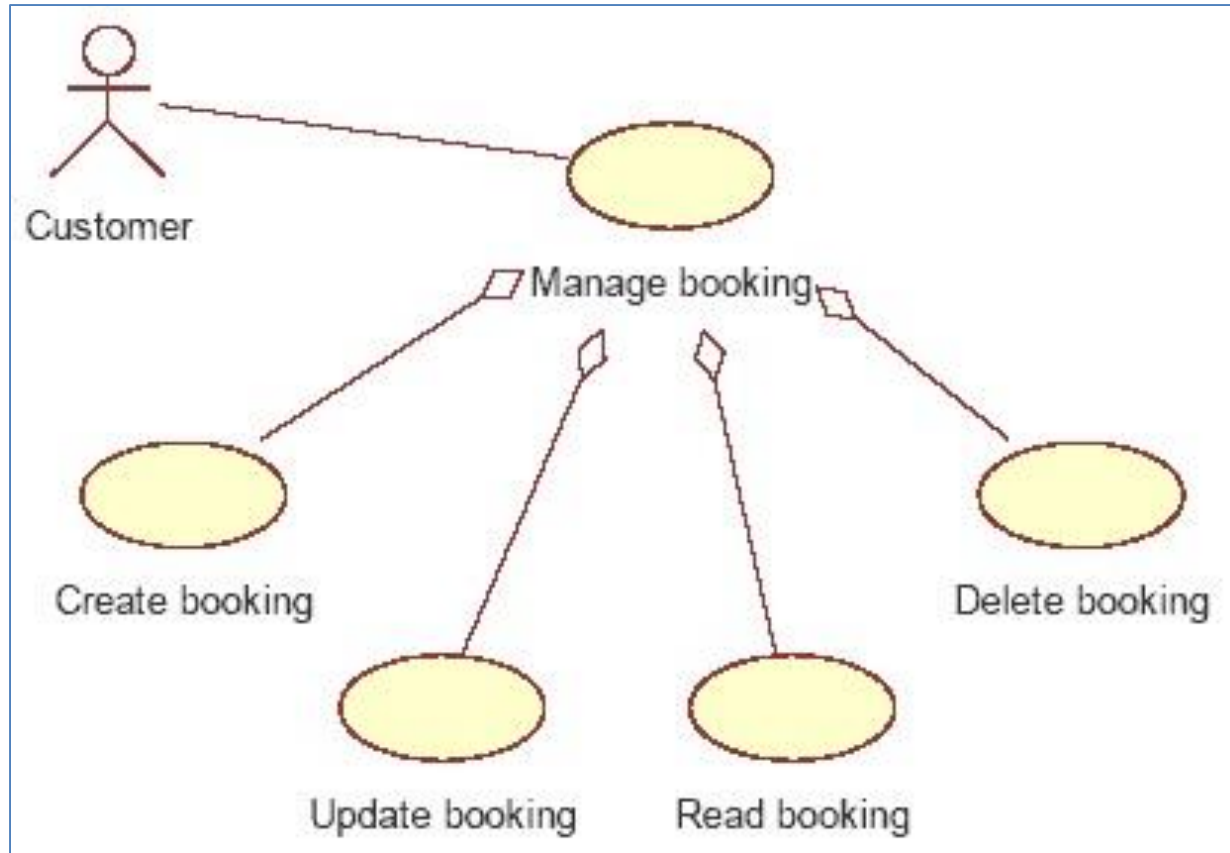
---

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

- **C**reate or add new objects
- **R**ead, retrieve or search for existing objects
- **U**ppdate or edit existing objects
- **D**delete existing objects

# CRUD – Example

---

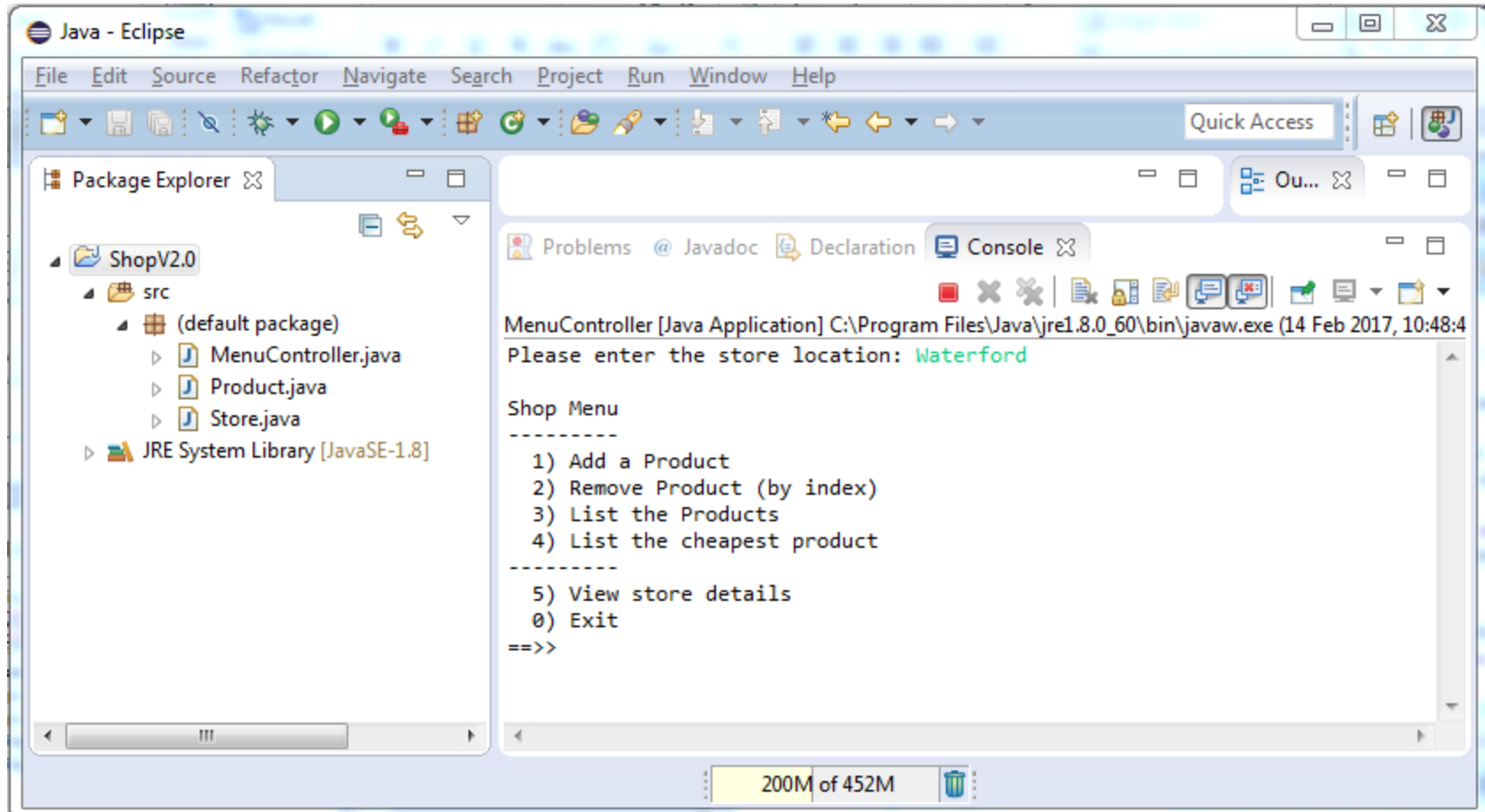


# Topic List

---

- What is CRUD?
- Shop V2.0 – a recap on **CRUD**
- Shop V3.0 – adding Update (CR**U**D)
- DRY (Don't Repeat Yourself)

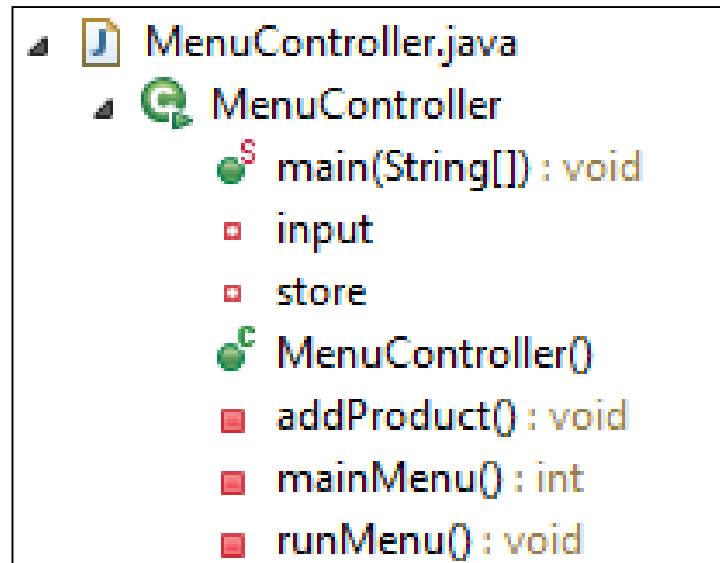
# ShopV2.0 – a recap



# ShopV2.0 – a recap

---

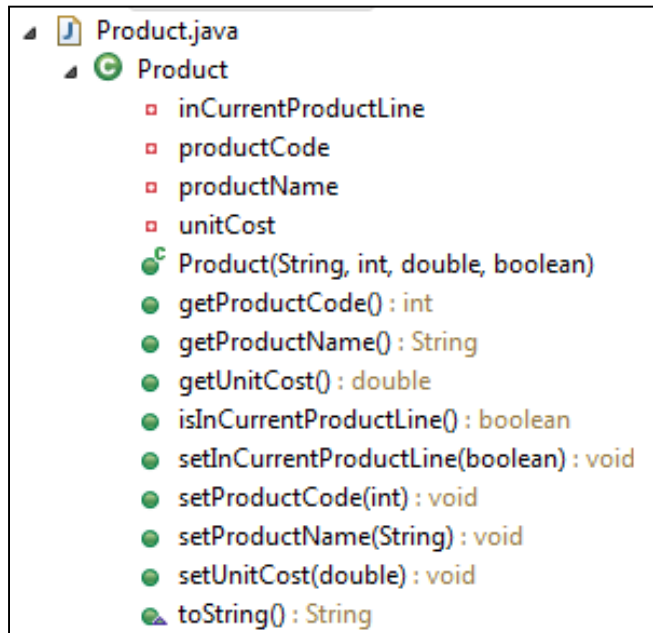
- MenuController
  - Controls I/O via menu driven console.
  - Contains the main method.
  - Contains a series of private methods and fields.



# ShopV2.0 – a recap

---

- Product
  - Four private instance fields (with validation).
  - Basic template class with public Constructors, Getters, Setters and a toString method.



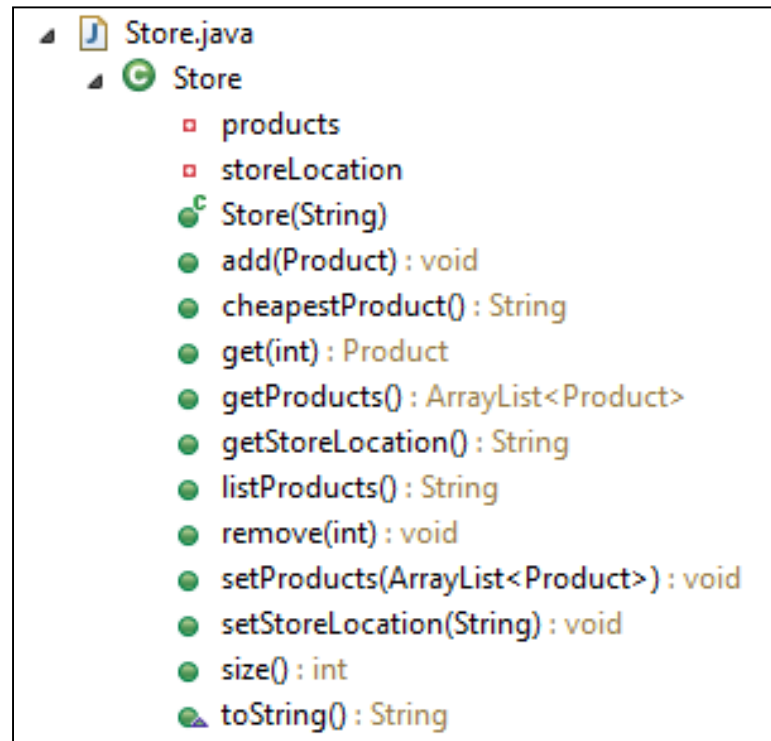
```
Product.java
├── Product
│   ├── inCurrentProductLine
│   ├── productCode
│   ├── productName
│   └── unitCost
│   ├── Product(String, int, double, boolean)
│   ├── getProductCode() : int
│   ├── getProductName() : String
│   ├── getUnitCost() : double
│   ├── isInCurrentProductLine() : boolean
│   ├── setInCurrentProductLine(boolean) : void
│   ├── setProductCode(int) : void
│   ├── setProductName(String) : void
│   ├── setUnitCost(double) : void
│   └── toString() : String
```



# ShopV2.0 – a recap

---

- Store
  - Two private instance fields.
  - Has many public methods that handle the ArrayList of product.



```
Store.java
├── Store
│   ├── products
│   ├── storeLocation
│   ├── Store(String)
│   ├── add(Product) : void
│   ├── cheapestProduct() : String
│   ├── get(int) : Product
│   ├── getProducts() : ArrayList<Product>
│   ├── getStoreLocation() : String
│   ├── listProducts() : String
│   ├── remove(int) : void
│   ├── setProducts(ArrayList<Product>) : void
│   ├── setStoreLocation(String) : void
│   ├── size() : int
│   └── toString() : String
```

# ShopV2.0 – a recap

---

- **C**reate a Product: Menu Option 1.
- **R**ead a Product(s): Menu Options 3 & 4.
- **U**ppdate a Product: The menu has NO **U**ppdate!
- **D**elate a Product: Menu Option 2.

```
Please enter the store location: Waterford

Shop Menu
-----
 1) Add a Product
 2) Remove Product (by index)
 3) List the Products
 4) List the cheapest product
-----
 5) View store details
 0) Exit
==>>
```

# ShopV2.0 – a recap

```
switch (option){
  case 1:    addProduct();
  break;
  case 2:    System.out.println(store.listProducts());
  if (store.size() > 0){
    System.out.print("Please enter the index for the product you wish to delete: ");
    int index = input.nextInt();
    store.remove(index);
  }
  break;
  case 3:    System.out.println(store.listProducts());
  break;
  case 4:    System.out.println(store.cheapestProduct());
  break;
  case 5:    System.out.println(store.toString());
  break;
  default:   System.out.println("Invalid option entered: " + option);
  break;
}
```

Create a Product:  
Menu Option 1.

MenuController  
Class


```
//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.println("Enter the Product details...");
  System.out.print("\tName: ");
  String productName = input.nextLine();
  System.out.print("\tCode (between 1000 and 9999): ");
  int productCode = input.nextInt();
  System.out.print("\tUnit Cost: ");
  double unitCost = input.nextDouble();
  System.out.print("\tIs 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));
}
```

# ShopV2.0 – a recap

## MenuController Class

```
switch (option){
    case 1:    addProduct();
    break;
    case 2:    System.out.println(store.listProducts());
    if (store.size() > 0){
        System.out.print("Please enter the index for the product you wish to delete: ");
        int index = input.nextInt();
        store.remove(index);
    }
    break;
    case 3:    System.out.println(store.listProducts());
    break;
    case 4:    System.out.println(store.cheapestProduct());
    break;
    case 5:    System.out.println(store.toString());
    break;
    default:   System.out.println("Invalid option entered: " + option);
    break;
}
```



Read a Product(s):  
Menu Option 3


## Store Class

```
public String listProducts()
{
    String listOfProducts = "";
    for (int index = 0; index < products.size(); index++)
    {
        listOfProducts = listOfProducts + index + ": " + products.get(index).toString() + "\n";
    }
    if (listOfProducts.equals(""))
    {
        return "No products";
    }
    else
    {
        return listOfProducts;
    }
}
```

# ShopV2.0 – a recap

## MenuController Class

```
switch (option){
    case 1:    addProduct();
              break;
    case 2:    System.out.println(store.listProducts());
              if (store.size() > 0){
                  System.out.print("Please enter the index for the product you wish to delete: ");
                  int index = input.nextInt();
                  store.remove(index);
              }
              break;
    case 3:    System.out.println(store.listProducts());
              break;
    case 4:    System.out.println(store.cheapestProduct());
              break;
    case 5:    System.out.println(store.toString());
              break;
    default:   System.out.println("Invalid option entered: " + option);
              break;
}
```



Read a Product(s):  
Menu Option 4


## Store Class

```
public String cheapestProduct()
{
    if (products.size() > 0)
    {
        Product cheapestProduct = products.get(0);
        for (Product product : products)
        {
            if (product.getUnitCost() < cheapestProduct.getUnitCost() )
                cheapestProduct = product;
        }
        return cheapestProduct.getProductName();
    }
    else
        return "No Products";
}
```

# ShopV2.0 – a recap

## MenuController Class

```
switch (option){
    case 1:    addProduct();
              break;
    case 2:    System.out.println(store.listProducts());
              if (store.size() > 0){
                  System.out.print("Please enter the index for the product you wish to delete: ");
                  int index = input.nextInt();
                  store.remove(index);
              }
              break;
    case 3:    System.out.println(store.listProducts());
              break;
    case 4:    System.out.println(store.cheapestProduct());
              break;
    case 5:    System.out.println(store.toString());
              break;
    default:   System.out.println("Invalid option entered: " + option);
              break;
}
```



Delete a Product:  
Menu Option 2

## Store Class

```
public void remove(int index)
{
    if ((index >= 0) && (index < products.size()))
    {
        products.remove(index);
    }
}
```

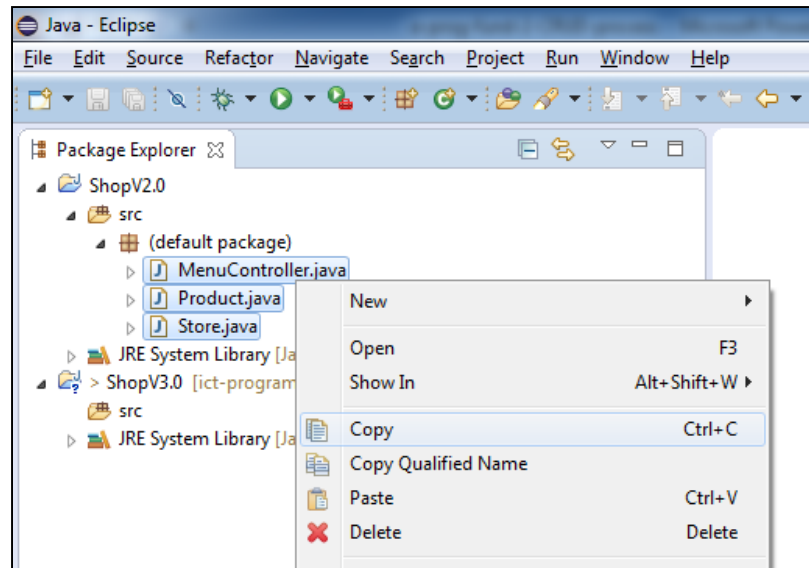
# Topic List

---

- What is CRUD?
- Shop V2.0 – a recap on **CRUD**
- Shop V3.0 – adding Update (CR**U**D)
- DRY (Don't Repeat Yourself)

# ShopV3.0 - adding Update

- In Eclipse:
  - create a new Java Project and call it ShopV3.0
  - Select the three java files in ShopV2.0.
  - Right click on the selection and choose **Copy**.

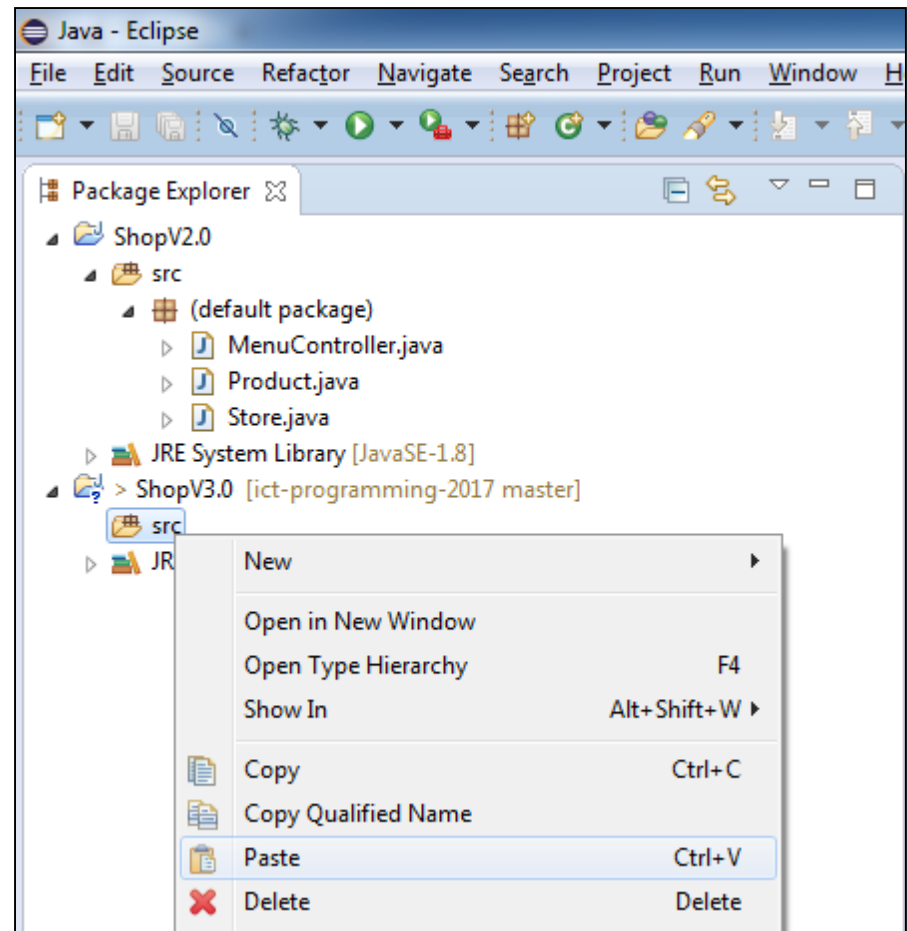




# ShopV3.0 - adding Update

---

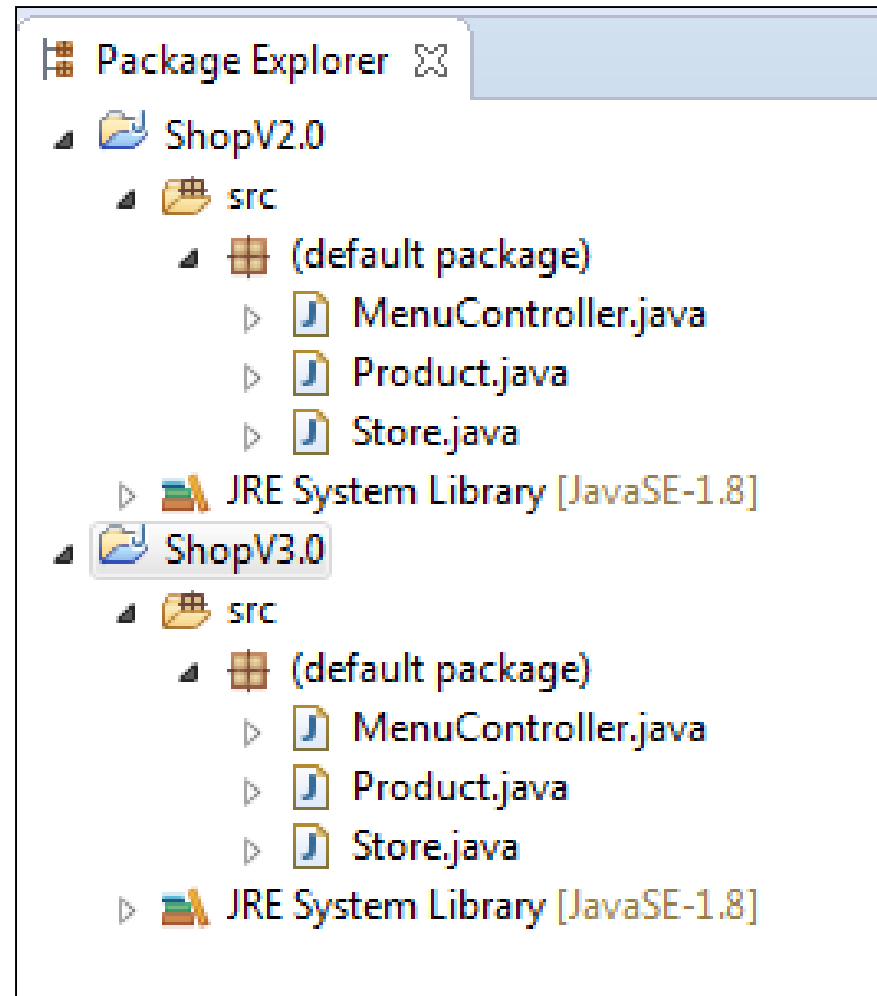
- Right click on the src folder in ShopV3.0 and choose **Paste**.



# ShopV3.0 - adding Update

---

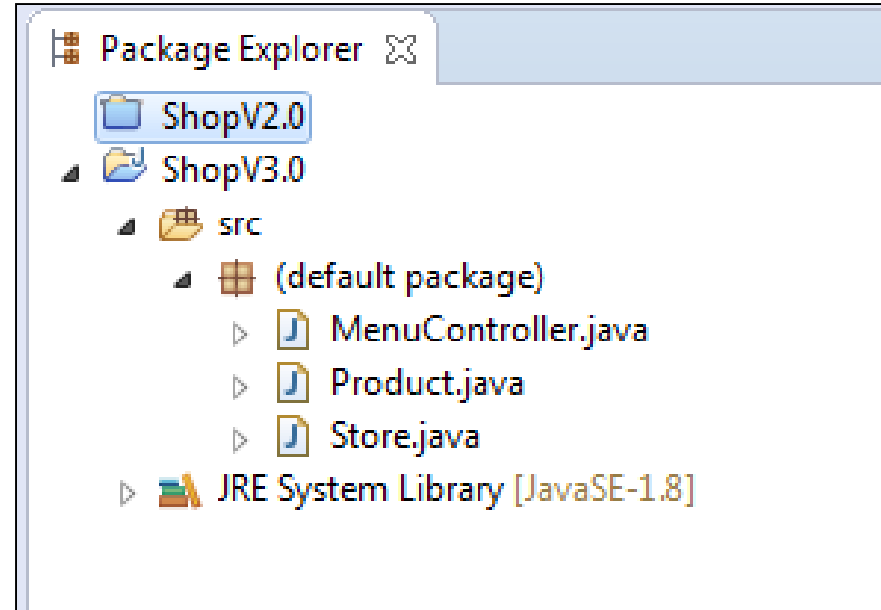
- Having refreshed (F5) your package explorer, ShopV3.0 should now look like this:



# ShopV3.0 - adding Update

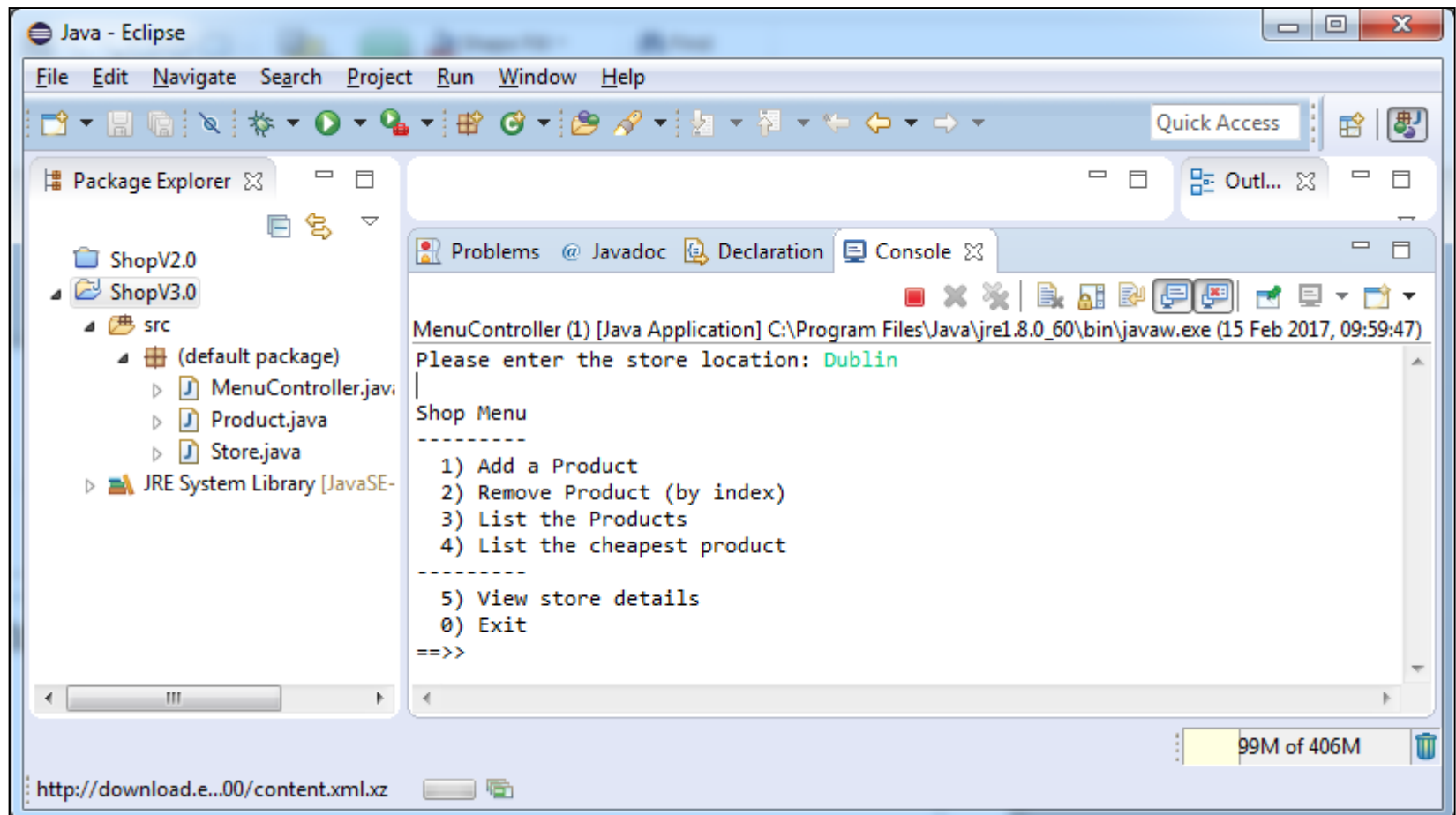
---

- You can now close ShopV2.0 by right clicking and selecting **Close project**.



# ShopV3.0 - adding Update

- Run ShopV3.0 to ensure all is ok before we start coding:



# ShopV3.0 - adding Update

---

```
Please enter the store location: Dublin

Shop Menu
-----
 1) Add a Product
 2) Remove Product (by index)
 3) List the Products
 4) List the cheapest product
-----
 5) View store details
 0) Exit
==>>
```



```
Please enter the store location: Dublin
|
Shop Menu
-----
 1) Add a Product
 2) List the Products
 3) Update a Product
 4) Remove Product (by index)
-----
 5) List the cheapest product
-----
 6) View store details
 0) Exit
==>>
```

# ShopV3.0 - adding Update

---

```
private int mainMenu()  
{  
    System.out.println("\fShop Menu");  
    System.out.println("-----");  
    System.out.println(" 1) Add a Product");  
    System.out.println(" 2) List the Products");  
    System.out.println(" 3) Update a Product");  
    System.out.println(" 4) Remove Product (by index)");  
    System.out.println("-----");  
    System.out.println(" 5) List the cheapest product");  
    System.out.println("-----");  
    System.out.println(" 6) View store details");  
    System.out.println(" 0) Exit");  
    System.out.print("==>> ");  
    int option = input.nextInt();  
    return option;  
}
```

```
Please enter the store location: Dublin  
|  
Shop Menu  
-----  
 1) Add a Product  
 2) List the Products  
 3) Update a Product  
 4) Remove Product (by index)  
-----  
 5) List the cheapest product  
-----  
 6) View store details  
 0) Exit  
==>>
```

# ShopV3.0 - adding Update

```
private int mainMenu()  
{  
    System.out.println("\fShop Menu");  
    System.out.println("-----");  
    System.out.println(" 1) Add a Product");  
    System.out.println(" 2) List the Products");  
    System.out.println(" 3) Update a Product");  
    System.out.println(" 4) Remove Product (by index)");  
    System.out.println("-----");  
    System.out.println(" 5) List the cheapest product");  
    System.out.println("-----");  
    System.out.println(" 6) View store details");  
    System.out.println(" 0) Exit");  
    System.out.print("==>> ");  
    int option = input.nextInt();  
    return option;  
}
```

MenuController.java

```
switch (option){  
    case 1:    addProduct();  
              break;  
    case 2:    System.out.println(store.listProducts());  
              break;  
    case 3:    updateProduct();  
              break;  
    case 4:    System.out.println(store.listProducts());  
              if (store.size() > 0){  
                  System.out.print("Please enter the index for the product you wish to delete: ");  
                  int index = input.nextInt();  
                  store.remove(index);  
              }  
              break;  
    case 5:    System.out.println(store.cheapestProduct());  
              break;  
    case 6:    System.out.println(store.toString());  
              break;  
    default:   System.out.println("Invalid option entered: " + option);  
              break;  
}
```

# ShopV3.0 - adding Update, #1

```
private void updateProduct() {  
  
    //retrieve the index for the product for update  
    System.out.println(store.listProducts());  
    int index = -1;  
    if (store.size() > 0){  
        System.out.print("Please enter the index for the product you wish to update: ");  
        index = input.nextInt();  
    }  
  
    input.nextLine(); //dummy read - bug in Scanner class.  
  
    //gather details to update the selected product with  
    System.out.println("Enter the Product details...");  
    System.out.print("\tName: ");  
    String productName = input.nextLine();  
    System.out.print("\tCode (between 1000 and 9999): ");  
    int productCode = input.nextInt();  
    System.out.print("\tUnit Cost: ");  
    double unitCost = input.nextDouble();  
    System.out.print("\tIs this product in your current line (y/n): ");  
    char currentProduct = input.next().charAt(0);  
    boolean inCurrentProductLine = false;  
    if ((currentProduct == 'y') || (currentProduct == 'Y'))  
        inCurrentProductLine = true;  
  
    //retrieve the product and update it  
    Product product = store.get(index);  
    product.setInCurrentProductLine(inCurrentProductLine);  
    product.setProductCode(productCode);  
    product.setProductName(productName);  
    product.setUnitCost(unitCost);  
}
```

Will this  
work?

What  
happens if  
we have **no**  
**products** in  
the  
ArrayList?

MenuController.java



# ShopV3.0 - adding Update, #2

```
private void updateProduct() {  
  
    //retrieve the index for the product for update  
    System.out.println(store.listProducts());  
    if (store.size() > 0){  
        System.out.print("Please enter the index for the product you wish to update: ");  
        int index = input.nextInt();  
  
        input.nextLine(); //dummy read - bug in Scanner class.  
  
        //gather details to update the selected product with  
        System.out.println("Enter the Product details...");  
        System.out.print("\tName: ");  
        String productName = input.nextLine();  
        System.out.print("\tCode (between 1000 and 9999): ");  
        int productCode = input.nextInt();  
        System.out.print("\tUnit Cost: ");  
        double unitCost = input.nextDouble();  
        System.out.print("\tIs this product in your current line (y/n): ");  
        char currentProduct = input.next().charAt(0);  
        boolean inCurrentProductLine = false;  
        if ((currentProduct == 'y') || (currentProduct == 'Y'))  
            inCurrentProductLine = true;  
  
        //retrieve the product and update it  
        Product product = store.get(index);  
        product.setInCurrentProductLine(inCurrentProductLine);  
        product.setProductCode(productCode);  
        product.setProductName(productName);  
        product.setUnitCost(unitCost);  
    }  
}
```

Is this  
better?

What  
happens  
now if we  
have **no**  
**products**?

MenuController.java

# ShopV3.0 - adding Update, #2

```
public String listProducts()  
{  
    String listOfProducts = "";  
    for (int index = 0; index < products.size(); index++)  
    {  
        listOfProducts = listOfProducts + index + ": " + products.get(index).toString() + "\n";  
    }  
    if (listOfProducts.equals(""))  
    {  
        return "No products";  
    }  
    else  
    {  
        return listOfProducts;  
    }  
}
```

Store.java

Shop Menu

```
-----  
1) Add a Product  
2) List the Products  
3) Update a Product  
4) Remove Product (by index)  
-----  
5) List the cheapest product  
-----  
6) View store details  
0) Exit  
==>> 3  
No products  
  
Press any key to continue...
```

**No  
products**  
results in a  
message  
being  
printed to  
the  
console.

# ShopV3.0 - adding Update, #2

```
private void updateProduct() {  
  
    //retrieve the index for the product for update  
    System.out.println(store.listProducts());  
    if (store.size() > 0){  
        System.out.print("Please enter the index for the product you wish to update: ");  
        int index = input.nextInt();  
  
        input.nextLine(); //dummy read - bug in Scanner class.  
  
        //gather details to update the selected product with  
        System.out.println("Enter the Product details...");  
        System.out.print("\tName: ");  
        String productName = input.nextLine();  
        System.out.print("\tCode (between 1000 and 9999): ");  
        int productCode = input.nextInt();  
        System.out.print("\tUnit Cost: ");  
        double unitCost = input.nextDouble();  
        System.out.print("\tIs this product in your current line (y/n): ");  
        char currentProduct = input.next().charAt(0);  
        boolean inCurrentProductLine = false;  
        if ((currentProduct == 'y') || (currentProduct == 'Y'))  
            inCurrentProductLine = true;  
  
        //retrieve the product and update it  
        Product product = store.get(index);  
        product.setInCurrentProductLine(inCurrentProductLine);  
        product.setProductCode(productCode);  
        product.setProductName(productName);  
        product.setUnitCost(unitCost);  
    }  
}
```

What happens if we have products but we enter an **invalid index**?

MenuController.java

# ShopV3.0 - adding Update, #2

```
Shop Menu
-----
1) Add a Product
2) List the Products
3) Update a Product
4) Remove Product (by index)
-----
5) List the cheapest product
-----
6) View store details
0) Exit
==>> 3
0: Product: Product Name(code: 1002). Unit cost: 19.99. In current product line: true

Please enter the index for the product you wish to update: 5
Enter the Product details...
  Name: New Product Name
  Code (between 1000 and 9999): 1004
  Unit Cost: 29.22
  Is this product in your current line (y/n): y
Exception in thread "main" java.lang.NullPointerException
    at MenuController.updateProduct(MenuController.java:137)
    at MenuController.runMenu(MenuController.java:63)
    at MenuController.<init>(MenuController.java:25)
    at MenuController.main(MenuController.java:14)
```

This happens if we have products but we enter an **invalid index!**

# ShopV3.0 - adding Update, #3

```
private void updateProduct() {  
  
    //retrieve the index for the product for update  
    System.out.println(store.listProducts());  
    if (store.size() > 0){  
        System.out.print("Please enter the index for the product you wish to update: ");  
        int index = input.nextInt();  
  
        if ((index >=0) && (index < store.size())){  
            input.nextLine(); //dummy read - bug in Scanner class.  
  
            //gather details to update the selected product with  
            System.out.println("Enter the Product details...");  
            System.out.print("\tName: ");  
            String productName = input.nextLine();  
            System.out.print("\tCode (between 1000 and 9999): ");  
            int productCode = input.nextInt();  
            System.out.print("\tUnit Cost: ");  
            double unitCost = input.nextDouble();  
            System.out.print("\tIs this product in your current line (y/n): ");  
            char currentProduct = input.next().charAt(0);  
            boolean inCurrentProductLine = false;  
            if ((currentProduct == 'y') || (currentProduct == 'Y'))  
                inCurrentProductLine = true;  
  
            //retrieve the product and update it  
            Product product = store.get(index);  
            product.setInCurrentProductLine(inCurrentProductLine);  
            product.setProductCode(productCode);  
            product.setProductName(productName);  
            product.setUnitCost(unitCost);  
        }  
        else {  
            System.out.println("Invalid index entered: " + index);  
        }  
    }  
}
```

NOW, what happens if we have products but we enter an **invalid index?**

# ShopV3.0 - adding Update, #3

```
Shop Menu
-----
1) Add a Product
2) List the Products
3) Update a Product
4) Remove Product (by index)
-----
5) List the cheapest product
-----
6) View store details
0) Exit
==>> 3
0: Product: Product Name(code: 1004). Unit cost: 3.43. In current product line: true

Please enter the index for the product you wish to update: -1
Invalid index entered: -1

Press any key to continue...
```

```
Shop Menu
-----
1) Add a Product
2) List the Products
3) Update a Product
4) Remove Product (by index)
-----
5) List the cheapest product
-----
6) View store details
0) Exit
==>> 3
0: Product: Product Name(code: 1004). Unit cost: 3.43. In current product line: true

Please enter the index for the product you wish to update: 1
Invalid index entered: 1

Press any key to continue...
```

This happens if we have products but we enter an **invalid index!**

MenuController.java

# Topic List

---

- What is CRUD?
- Shop V2.0 – a recap on **CRUD**
- Shop V3.0 – adding Update (CR**U**D)
- **DRY (Don't Repeat Yourself)**

# DRY Principle

---

- If you are copying and pasting blocks of code from one method to another, you are in violation of the DRY principle...Don't Repeat Yourself.
- Consider instead moving the code into a "reusable unit" e.g. a private helper method.
- DRY == Don't write the same code repeatedly!




# ShopV3.0 – DRYing it out!

---

```
private void addProduct(){
    //dummy read of String to clear the buffer - bug in Scanner class.
    input.nextLine();
    System.out.println("Enter the Product details...");
    System.out.print("\tName:  ");
    String productName = input.nextLine();
    System.out.print("\tCode (between 1000 and 9999):  ");
    int productCode = input.nextInt();
    System.out.print("\tUnit Cost:  ");
    double unitCost = input.nextDouble();
    System.out.print("\tIs 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));
}
```



```
private void updateProduct() {
    //retrieve the index for the product for update
    System.out.println(store.listProducts());
    if (store.size() > 0){
        System.out.print("Please enter the index for the product you wish to update: ");
        int index = input.nextInt();

        if ((index >=0) && (index < store.size())){
            input.nextLine(); //dummy read - bug in Scanner class.
            //gather details to update the selected product with
            System.out.println("Enter the Product details...");
            System.out.print("\tName: ");
            String productName = input.nextLine();
            System.out.print("\tCode (between 1000 and 9999): ");
            int productCode = input.nextInt();
            System.out.print("\tUnit Cost: ");
            double unitCost = input.nextDouble();
            System.out.print("\tIs this product in your current line (y/n): ");
            char currentProduct = input.next().charAt(0);
            boolean inCurrentProductLine = false;
            if ((currentProduct == 'y') || (currentProduct == 'Y'))
                inCurrentProductLine = true;

            //retrieve the product and update it
            Product product = store.get(index);
            product.setInCurrentProductLine(inCurrentProductLine);
            product.setProductCode(productCode);
            product.setProductName(productName);
            product.setUnitCost(unitCost);
        }
        else {
            System.out.println("Invalid index entered: " + index);
        }
    }
}
```

# ShopV3.0 – DRYing it out!

---

- Repeated Code in both `addProduct()` and `updateProduct()`:

```
System.out.println("Enter the Product details...");
System.out.print("\tName:  ");
String productName = input.nextLine();
System.out.print("\tCode (between 1000 and 9999):  ");
int productCode = input.nextInt();
System.out.print("\tUnit Cost:  ");
double unitCost = input.nextDouble();
System.out.print("\tIs this product in your current line (y/n): ");
char currentProduct = input.next().charAt(0);
boolean inCurrentProductLine = false;
if ((currentProduct == 'y') || (currentProduct == 'Y'))
    inCurrentProductLine = true;
```

- Can we farm it out into a private helper method?

# ShopV3.0 – DRYing it out!

---

- Repeated Code in new private helper method:

```
private Product readProductDetails() {
    //read the product details from the user and return them as a product object
    System.out.println("Enter the Product details...");
    System.out.print("\tName:  ");
    String productName = input.nextLine();
    System.out.print("\tCode (between 1000 and 9999):  ");
    int productCode = input.nextInt();
    System.out.print("\tUnit Cost:  ");
    double unitCost = input.nextDouble();
    System.out.print("\tIs this product in your current line (y/n):  ");
    char currentProduct = input.next().charAt(0);
    boolean inCurrentProductLine = false;
    if ((currentProduct == 'y') || (currentProduct == 'Y'))
        inCurrentProductLine = true;


    return (new Product(productName, productCode, unitCost, inCurrentProductLine));
}
```

# ShopV3.0 – DRYing it out!

---

- Calling the new private helper method from `addProduct()` in place of the repeated code:

```
private void addProduct() {  
    //dummy read of String to clear the buffer - bug in Scanner class.  
    input.nextLine();  
  
    //gather details of the new product  
    Product productDetails = readProductDetails();  
  
    //store the new product in the array list  
    store.add(productDetails);  
}
```




# ShopV3.0 – DRYing it out!

---

- Calling the new private helper method from `addProduct()` in place of the repeated code **(in one line)**:

```
private void addProduct() {  
    //dummy read of String to clear the buffer - bug in Scanner class.  
    input.nextLine();  
  
    //gather details of the new product AND  
    //store the new product in the array list  
    store.add(readProductDetails());  
}
```



# ShopV3.0 – DRYing it out!

---

```
private void updateProduct() {
    //retrieve the index for the product for update
    System.out.println(store.listProducts());
    if (store.size() > 0){
        System.out.print("Please enter the index for the product you wish to update: ");
        int index = input.nextInt();

        if ((index >=0) && (index < store.size())){
            input.nextLine(); //dummy read - bug in Scanner class.

            //gather details to update the selected product with
            Product productDetails = readProductDetails();

            //retrieve the product and update it using the details entered in by the user
            Product productToUpdate = store.get(index);
            productToUpdate.setInCurrentProductLine(productDetails.isInCurrentProductLine());
            productToUpdate.setProductCode(productDetails.getProductCode());
            productToUpdate.setProductName(productDetails.getProductName());
            productToUpdate.setUnitCost(productDetails.getUnitCost());
        }
        else {
            System.out.println("Invalid index entered: " + index);
        }
    }
}
```

# ShopV3.0 – DRYing it out!

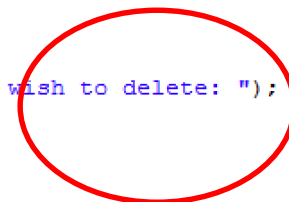
---

- Is there anything else we could push out to a helper method?



```
while (option != 0){
    switch (option){
        case 1:    addProduct();
                  break;
        case 2:    System.out.println(store.listProducts());
                  break;
        case 3:    updateProduct();
                  break;
        case 4:    System.out.println(store.listProducts());
                  if (store.size() > 0){
                      System.out.print("Please enter the index for the product you wish to delete: ");
                      int index = input.nextInt();
                      store.remove(index);
                  }
                  break;
        case 5:    System.out.println(store.cheapestProduct());
                  break;
        case 6:    System.out.println(store.toString());
                  break;
        default:   System.out.println("Invalid option entered: " + option);
    }
}
```

MenuController.  
java

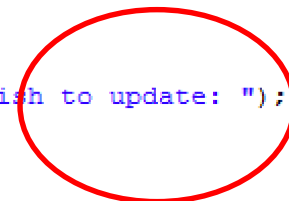


```
private void updateProduct() {
    //retrieve the index for the product for update
    System.out.println(store.listProducts());
    if (store.size() > 0){
        System.out.print("Please enter the index for the product you wish to update: ");
        int index = input.nextInt();

        if ((index >=0) && (index < store.size())){
            input.nextLine(); //dummy read - bug in Scanner class.

            //gather details to update the selected product with
            Product productDetails = readProductDetails();

            //retrieve the product and update it using the details entered in by the user
            Product productToUpdate = store.get(index);
            productToUpdate.setCurrentProductLine(productDetails.isInCurrentProductLine());
            productToUpdate.setProductCode(productDetails.getProductCode());
            productToUpdate.setProductName(productDetails.getProductName());
            productToUpdate.setUnitCost(productDetails.getUnitCost());
        }
        else {
            System.out.println("Invalid index entered: " + index);
        }
    }
}
```



```
while (option != 0){
    switch (option){
        case 1:    addProduct();
                  break;
        case 2:    System.out.println(store.listProducts());
                  break;
        case 3:    updateProduct();
                  break;
        case 4:    System.out.println(store.listProducts());
                  if (store.size() > 0){
                      System.out.print("Please enter the index for the product you wish to delete: ");
                      int index = input.nextInt();
                      store.remove(index);
                  }
                  break;
        case 5:    System.out.println(store.cheapestProduct());
                  break;
        case 6:    System.out.println(store.toString());
                  break;
        default:   System.out.println("Invalid option entered: " + option);
                  break;
    }
}
```

MenuController.  
java

```
public void remove(int index)
{
    if ((index >= 0) && (index < products.size()))
    {
        products.remove(index);
    }
}
```


Store.java

The remove  
method in  
Store  
validates  
the index  
passed to it

# ShopV3.0 – DRYing it out!

MenuController.java

```
switch (option){
    case 1:    addProduct ();
               break;
    case 2:    System.out.println(store.listProducts ());
               break;
    case 3:    updateProduct ();
               break;
    case 4:    store.remove (getIndex ());
               break;
    case 5:    System.out.println(store.cheapestProduct ());
               break;
    case 6:    System.out.println(store.toString ());
               break;
    default:   System.out.println("Invalid option entered: " + option);
               break;
}
```



Would this  
work?

```
private int getIndex(){
    System.out.println(store.listProducts ());
    if (store.size() > 0){
        System.out.print("Please enter the index to select the product: ");
        return input.nextInt ();
    }
    else {
        return -1;
    }
}
```

```
□Shop Menu
```

```
-----
```

- 1) Add a Product
- 2) List the Products
- 3) Update a Product
- 4) Remove Product (by index)

```
-----
```

5) List the cheapest product

```
-----
```

- 6) View store details
- 0) Exit

```
==>> 4
```

```
|No products
```

```
Press any key to continue...
```

```
□Shop Menu
```

```
-----
```

- 1) Add a Product
- 2) List the Products
- 3) Update a Product
- 4) Remove Product (by index)

```
-----
```

5) List the cheapest product

```
-----
```

- 6) View store details
- 0) Exit

```
==>> 4
```

```
0: Product: Coca Cola(code: 1003). Unit cost: 1.34. In cur
```

```
Please enter the index to select the product: 5
```

```
|  
Press any key to continue...
```

```
□Shop Menu
```

```
-----
```

- 1) Add a Product
- 2) List the Products
- 3) Update a Product
- 4) Remove Product (by index)

```
-----
```

5) List the cheapest product

```
-----
```

- 6) View store details
- 0) Exit

```
==>> 4
```

```
0: Product: Coca Cola(code: 1003). Unit cost: 1.34. In current product line: true
```

```
Please enter the index to select the product: 0
```

```
|  
Press any key to continue...
```

YES!

# ShopV3.0 – DRYing it out!

MenuController.java

```
switch (option){
    case 1:    addProduct ();
              break;
    case 2:    System.out.println(store.listProducts ());
              break;
    case 3:    updateProduct ();
              break;
    case 4:    store.remove (getIndex ());
              break;
    case 5:    System.out.println (store.cheapestProduct ());
              break;
    case 6:    System.out.println (store.toString ());
              break;
    default:   System.out.println ("Invalid option entered: " + option);
              break;
}
```

```
private int getIndex(){
    System.out.println(store.listProducts ());
    if (store.size() > 0){
        System.out.print ("Please enter the index to select the product: ");
        return input.nextInt ();
    }
    else {
        return -1;
    }
}
```

Let's use  
the new  
**getIndex()**  
method in  
the  
update...

```
private void updateProduct() {
    //retrieve the index for the product for update
    int index = getIndex();
    if ((index >=0) && (index < store.size())){
        input.nextLine(); //dummy read - bug in Scanner class.

        //gather details to update the selected product with
        Product productDetails = readProductDetails();

        //retrieve the product and update it using the details entered in by the user
        Product productToUpdate = store.get(index);
        productToUpdate.setInCurrentProductLine(productDetails.isInCurrentProductLine());
        productToUpdate.setProductCode(productDetails.getProductCode());
        productToUpdate.setProductName(productDetails.getProductName());
        productToUpdate.setUnitCost(productDetails.getUnitCost());
    }
    else {
        System.out.println("Invalid index entered: " + index);
    }
}
```

```
private int getIndex(){
    System.out.println(store.listProducts());
    if (store.size() > 0){
        System.out.print("Please enter the index to select the product: ");
        return input.nextInt();
    }
    else {
        return -1;
    }
}
```

Will this work?  
What happens  
when the  
ArrayList is  
empty?

```
Please enter the store location: Athlone
Shop Menu
-----
1) Add a Product
2) List the Products
3) Update a Product
4) Remove Product (by index)
-----
5) List the cheapest product
-----
6) View store details
0) Exit
==>> 3
No products
Invalid index entered: -1
Press any key to continue...
```

We have a bug!

Our new  
getIndex()  
method is not  
behaving itself!

We need to  
refactor it...

```
private int getIndex() {
    System.out.println(store.listProducts());
    if (store.size() > 0) {
        System.out.print("Please enter the index to select the product: ");
        return input.nextInt();
    }
    else {
        return -1;
    }
}
```

MenuController.java

```
private void updateProduct() {
    //retrieve the index for the product for update
    int index = getIndex();
    if ((index >=0) && (index < store.size())){
        input.nextLine(); //dummy read - bug in Scanner class.

        //gather details to update the selected product with
        Product productDetails = readProductDetails();

        //retrieve the product and update it using the details entered in by the user
        Product productToUpdate = store.get(index);
        productToUpdate.setCurrentProductLine(productDetails.isInCurrentProductLine());
        productToUpdate.setProductCode(productDetails.getProductCode());
        productToUpdate.setProductName(productDetails.getProductName());
        productToUpdate.setUnitCost(productDetails.getUnitCost());
    }
}
```

Remove the  
check for invalid  
index from  
here...

```
private int getIndex(){
    System.out.println(store.listProducts());
    if (store.size() > 0){
        System.out.print("Please enter the index to select the product: ");
        int index = input.nextInt();
        if ((index >= 0) && (index < store.size())){
            return index;
        }
        else{
            System.out.println("Invalid index entered: " + index);
            return -1; //error code - invalid index entered
        }
    }
    else {
        return -2; //error code - empty array list
    }
}
```

...to here



# ShopV3.0 – DRYing it out!

---

- Is there anything else we could push out to a helper method?

# ShopV3.0 – DRYing it out!

```
private void updateProduct() {  
    //retrieve the index for the product for update  
    int index = getIndex();  
    if ((index >=0) && (index < store.size())){  
        input.nextLine(); //dummy read - bug in Scanner class.  
    }  
}
```

MenuController.java

```
private int getIndex(){  
    System.out.println(store.listProducts());  
    if (store.size() > 0){  
        System.out.print("Please enter the index to select the product: ");  
        int index = input.nextInt();  
        if ((index >= 0) && (index < store.size())){  
            return index;  
        }  
    }  
}
```

MenuController.java

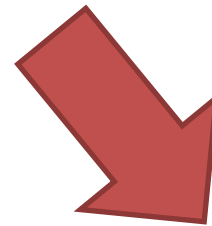
What could  
we do with  
these  
checks?

```
public void remove(int index)  
{  
    if ((index >= 0) && (index < products.size()))  
    {  
        products.remove(index);  
    }  
}
```

Store.java

Store.java

```
public void remove(int index)
{
    if ((index >= 0) && (index < products.size()))
    {
        products.remove(index);
    }
}
```

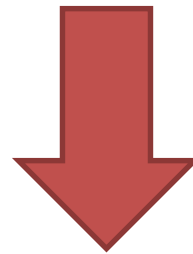


Store.java

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

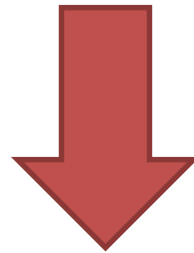
```
public void remove(int index)
{
    if (isValidIndex(index))
    {
        products.remove(index);
    }
}
```

```
private void updateProduct() {
    //retrieve the index for the product for update
    int index = getIndex();
    if ((index >=0) && (index < store.size())){
        input.nextLine(); //dummy read - bug in Scanner class.
```



```
private void updateProduct() {
    //retrieve the index for the product for update
    int index = getIndex();
    if (store.isValidIndex(index)){
        input.nextLine(); //dummy read - bug in Scanner class.
```

```
private int getIndex(){
    System.out.println(store.listProducts());
    if (store.size() > 0){
        System.out.print("Please enter the index to select the product: ");
        int index = input.nextInt();
        if ((index >= 0) && (index < store.size())){
            return index;
        }
    }
}
```



```
private int getIndex(){
    System.out.println(store.listProducts());
    if (store.size() > 0){
        System.out.print("Please enter the index to select the product: ");
        int index = input.nextInt();
        if (store.isValidIndex(index)){
            return index;
        }
    }
}
```

**Any  
Questions?**





Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>



Waterford Institute of Technology  
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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