

Persistence

An Introduction to the CRUD Process

Produced Dr. Siobhán Drohan
by: Mr. Colm Dunphy
 Mr. Diarmuid O'Connor
 Dr. Frank Walsh



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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

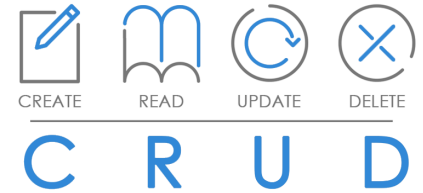
Topic List

1. What is CRUD?

2. Shop V4.0 (Driver.java):

- Recap of Shop V3.0
- 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

- **C**reate or add new objects



READ

- **R**ead, retrieve or search for existing objects



UPDATE

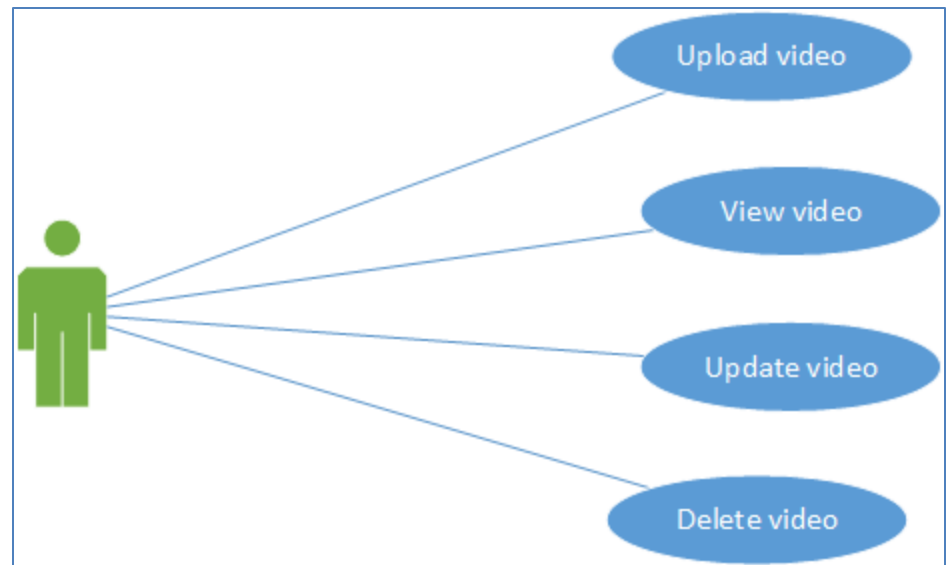
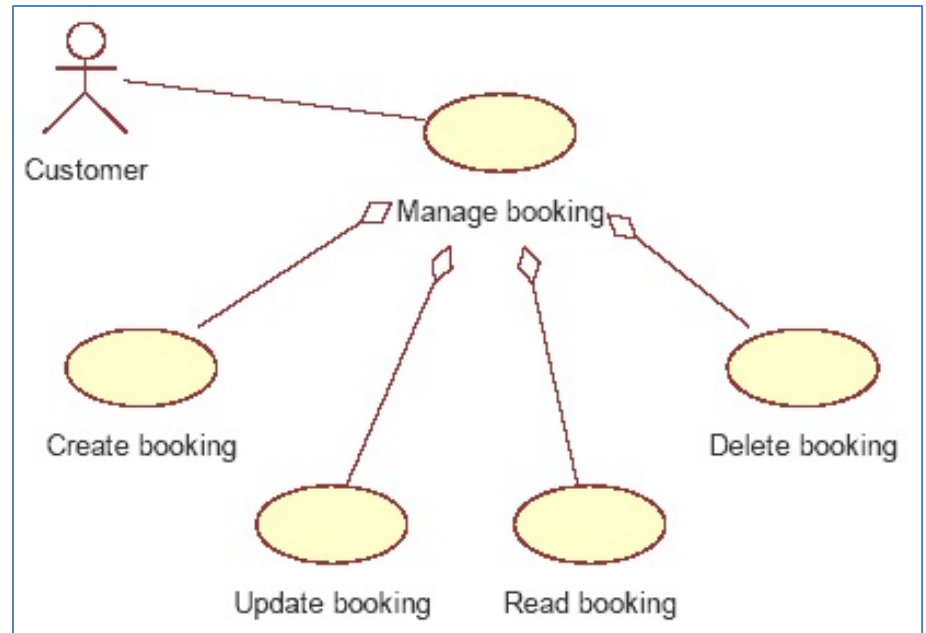
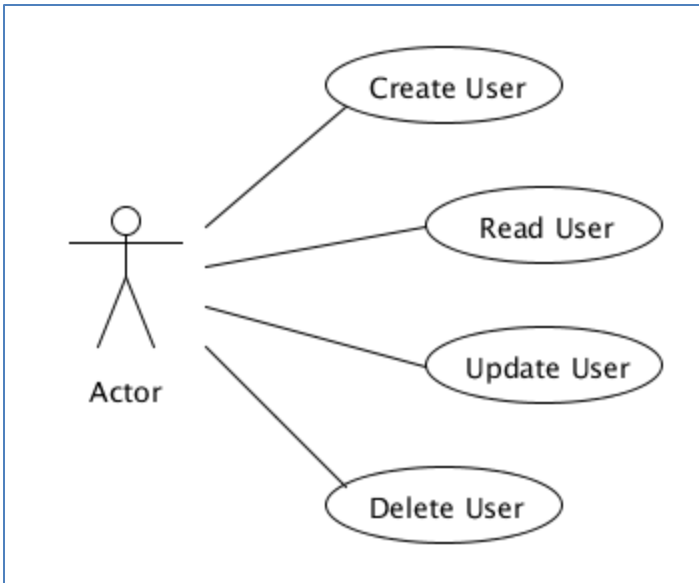
- **U**ppdate or edit existing objects



DELETE

- **D**elate existing objects


CRUD Examples



Topic List

1. What is CRUD?

2. Shop V4.0 (Driver.java):

- 
- Recap of Shop V3.0
 - 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)

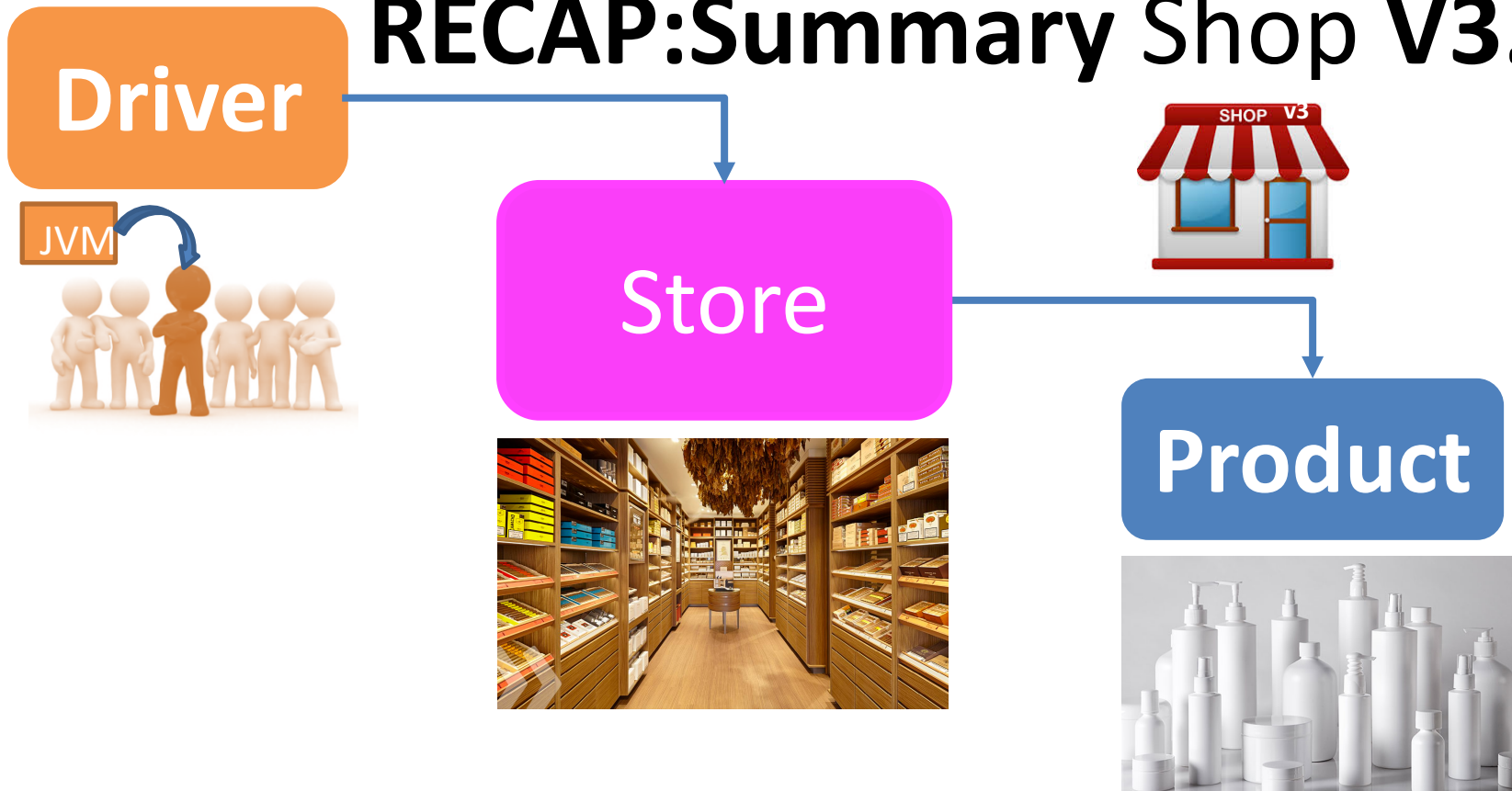
RECAP: Summary Shop V3.0



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: Summary Shop V3.0



Store class

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

RECAP: Summary Shop V3.0



Driver

- Runs the **menu**,
- contains the **main()** method
- negotiates with the user (i.e. handles **I/O**)

Shop V3.0 – a recap

- **C**reate a Product: Menu Option 1.
- **R**ead a Product(s): Menu Options 2 - 6.



- **The menu has NO Update or Delete!**



Shop Menu

- 1) Add a Product
- 2) List the Products

- 3) List the cheapest product
- 4) List the products in our current product line
- 5) Display average product unit cost
- 6) List products that are more expensive than a given price
- 0) Exit

==>>

Topic List

1. What is CRUD?

2. Shop V4.0 (Driver.java):

- Recap of Shop V3.0

- 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 cheapest product
- 6) List the products in our current product line
- 7) Display average product unit cost
- 8) List products that are more expensive than a given price
- 0) Exit

==>>



CREATE



READ



UPDATE



DELETE

Option 1 – Create a Product

Option 2 – Read products

Option 3 – Update a product

Option 4 – Delete a product

```
private int mainMenu()
{
    System.out.println("Shop 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) Delete a Product");
    System.out.println("-----");
    System.out.println("  5) List the cheapest product");
    System.out.println("  6) List the products in our current product line");
    System.out.println("  7) Display average product unit cost");
    System.out.println("  8) List products that are more expensive than a given price");
    System.out.println("  0) Exit");
    System.out.print("==>> ");
    int option = input.nextInt();
    return option;
}
```

NEXT:

We need to

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

Topic List

1. What is CRUD?

2. Shop V4.0 (Driver.java):

- Recap of Shop V3.0
- 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)

Code for case 1: Add a Product

```
switch (option)
{
    case 1: addProduct();
            break;
    case 2: System.out.println(store.listProducts());
            break;
}
```

```
//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));
}
```

Topic List

1. What is CRUD?

2. Shop V4.0 (Driver.java):

- Recap of Shop V3.0
- 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:

```
switch (option)
{
    case 1:    addProduct();
              break;
    case 2:    System.out.println(store.listProducts());
              break;
```

Code for case 2:
List the Products

Output from case 2 call:

```
Shop Menu
-----
1) Add a Product
2) List the Products
3) Update a Product
4) Delete a Product
-----
5) List the cheapest product
6) List the products in our current product line
7) Display average product unit cost
8) List products that are more expensive than a given price
0) Exit
==>> 2
0: Product description: 32 Inch TV, product code: 45443, unit cost: â,-3999.0, currently in product line: true
1: Product description: DVD Player, product code: 32445, unit cost: â,-1999.0, currently in product line: false
```

Code for case 2: List the Products

Store.java code:

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

Topic List

1. What is CRUD?

2. Shop V4.0 (Driver.java):

- Recap of Shop V3.0
- 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)

Code for case 4: Delete a Product

Driver.java code:

```
switch (option)
{
    case 1:    addProduct();
              break;
    case 2:    System.out.println(store.listProducts());
              break;
    case 4:    deleteProduct();
              break;
```

```
private void deleteProduct() {
    //list the products and ask the user to choose the product to delete
    System.out.println(store.listProducts());
    System.out.print("Enter the index of the product to delete ==> ");
    int index = input.nextInt();

    //delete the product at the given index
    store.getProducts().remove(index);
    System.out.println("Product deleted.");
}
```

The deleteProduct() method does not have any **validation**:

- What happens if there are **no products** in the ArrayList?
- What happens if the **index number does not exist** in the ArrayList?

```
private void deleteProduct() {  
    //list the products and ask the user to choose the product to delete  
    System.out.println(store.listProducts());  
    System.out.print("Enter the index of the product to delete ==> ");  
    int index = input.nextInt();  
  
    //delete the product at the given index  
    store.getProducts().remove(index);  
    System.out.println("Product deleted.");  
}
```


Validation:

- Only process the delete if **there are products** in the ArrayList and the **number entered is less than the size** of the ArrayList.

```
private void deleteProduct() {
    //list the products
    System.out.println(store.listProducts());

    if (store.getProducts().size() > 0) {
        //only ask the user to choose the product to delete if products exist
        System.out.print("Enter the index of the product to delete ==> ");
        int index = input.nextInt();

        if ((index >= 0) && (index < store.getProducts().size())) {
            //if the index is valid, delete the product at the given index
            store.getProducts().remove(index);
            System.out.println("Product deleted.");
        }
        else{
            System.out.println("There is no product for this index number");
        }
    }
}
```

Topic List

1. What is CRUD?

2. Shop V4.0 (Driver.java):

- Recap of Shop V3.0
- 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)

Coding case 3: Updating a Product

Driver.java code:

```
switch (option)
{
    case 1:    addProduct();
              break;
    case 2:    System.out.println(store.listProducts());
              break;
    case 3:    editProduct();
              break;
    case 4:    deleteProduct();
              break;
    case 5:    System.out.println(store.cheapestProduct());
              break;
}
```

Driver.java code:

Coding case 3: Updating a Product

```
private void editProduct() {
    //list the products and ask the user for an index to update
    System.out.println(store.listProducts());
    System.out.print("Enter the index of the product to update ==> ");
    int index = input.nextInt();

    //gather new details for each field from the user
    input.nextLine(); //dummy read of String to clear buffer - bug in Scanner.
    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;

    //retrieve the selected product from the ArrayList and update the details
    Product product = store.getProducts().get(index);
    product.setProductCode(productCode);
    product.setProductName(productName);
    product.setUnitCost(unitCost);
    product.setInCurrentProductLine(inCurrentProductLine);
}
```

The editProduct() method does not have any **validation** in it:

- What happens if there are **no products** in the ArrayList?
- What happens if the **index number does not exist** in the ArrayList?

Coding case 3: Updating a Product

Coding case 3: Updating a Product

```
private void editProduct() {
    //list the products
    System.out.println(store.listProducts());

    if (store.getProducts().size() > 0) {
        //only ask the user to choose a product if products exist
        System.out.print("Enter the index of the product to update ==> ");
        int index = input.nextInt();

        if ((index >= 0) && (index < store.getProducts().size())) {
            //if the index is valid, gather new details for each field from the user
            input.nextLine(); //dummy read of String to clear buffer - bug in Scanner.
            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;

            //retrieve the selected product from the ArrayList and update the details
            Product product = store.getProducts().get(index);
            product.setProductCode(productCode);
            product.setProductName(productName);
            product.setUnitCost(unitCost);
            product.setInCurrentProductLine(inCurrentProductLine);
        }
        else {
            System.out.println("There are no products for this index number");
        }
    }
}
```

**Any
Questions?**

