Laporan Final Project

PBO – B

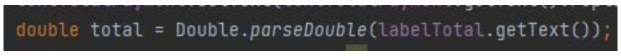
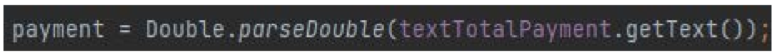
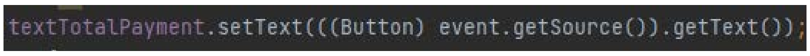
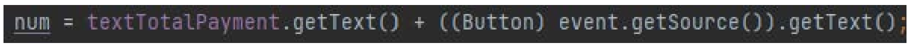
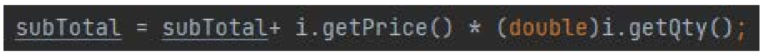
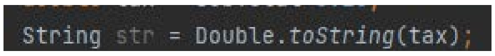
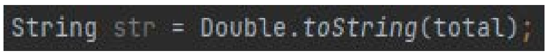
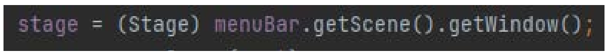
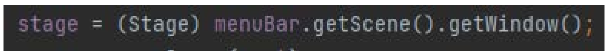
Nama : Richard Ryan

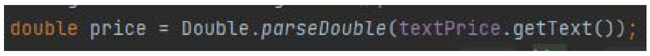
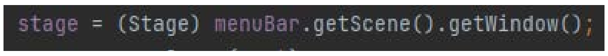
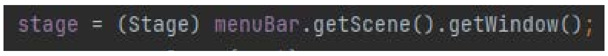
NRP : 5025211141

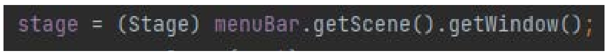
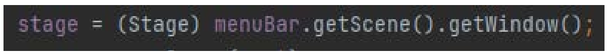
Dosen : Dr. Agus Budi Raharjo

Link Github : https://github.com/RichardRyan141/FP-PBO-2022

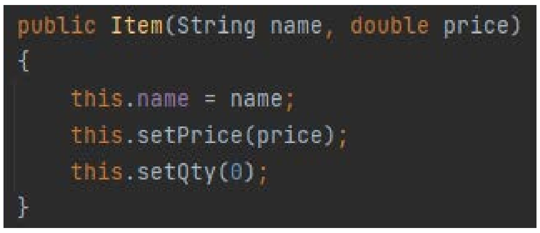
1. Casting / Conversion
   a) Cashier Controller

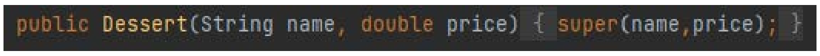
Line 138 : 
```java
double total = Double.parseDouble(labelTotal.getText());
```
Line 144 : 
```java
payment = Double.parseDouble(textTotalPayment.getText());
```
Line 313 : 
```java
textTotalPayment.setText(((Button) event.getSource()).getText());
```
Line 315 : 
```java
num = textTotalPayment.getText() + ((Button) event.getSource()).getText();
```
Line 372 : 
```java
subTotal = subTotal+ i.getPrice() * (double)i.getQty();
```
Line 381 : 
```java
String str = Double.toString(tax);
```
Line 390 : 
```java
String str = Double.toString(total);
```
Line 428 : 
```java
stage = (Stage) menuBar.getScene().getWindow();
```
Line 443 : 
```java
stage = (Stage) menuBar.getScene().getWindow();
```
   b) Product Add Controller

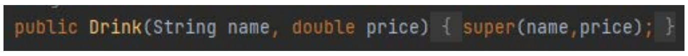
Line 67 : 
```java
double price = Double.parseDouble(textPrice.getText());
```
Line 185 : 
```java
stage = (Stage) menuBar.getScene().getWindow();
```
Line 205 : 
```java
stage = (Stage) menuBar.getScene().getWindow();
```
   c) Product Delete Controller

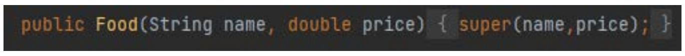
Line 175 : 
```java
stage = (Stage) menuBar.getScene().getWindow();
```
Line 190 : 
```java
stage = (Stage) menuBar.getScene().getWindow();
```

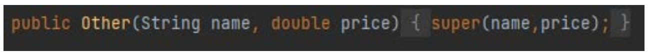
2. Constructor
   a) Item

Line 8 – 13 :

```java
public Item(String name, double price)
{
    this.name = name;
    this.setPrice(price);
    this.setQty(0);
}
```
   b) Dessert

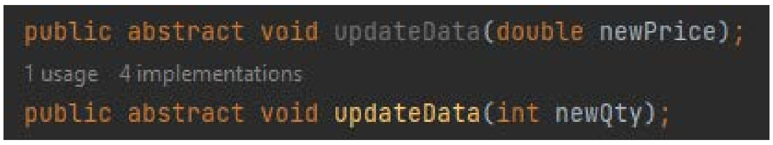
Line 5 : 
```java
public Dessert(String name, double price) { super(name,price); }
```
   c) Drink

Line 5 : 
```java
public Drink(String name, double price) { super(name,price); }
```
   d) Food

Line 5 : 
```java
public Food(String name, double price) { super(name,price); }
```
   e) Other

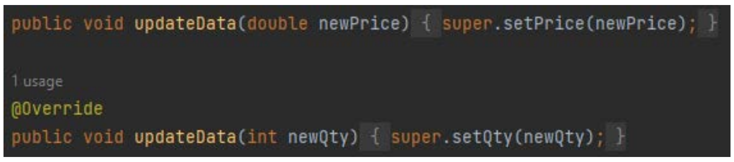
Line 5 : 
```java
public Other(String name, double price) { super(name,price); }
```

3. Overloading
   a) Item
      Line 15 – 16 :



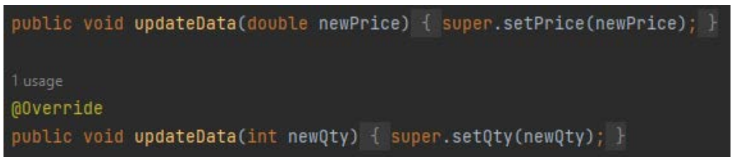
```
public abstract void updateData(double newPrice);
1 usage   4 implementations
public abstract void updateData(int newQty);
```

   b) Dessert
      Line 11 – 16 :



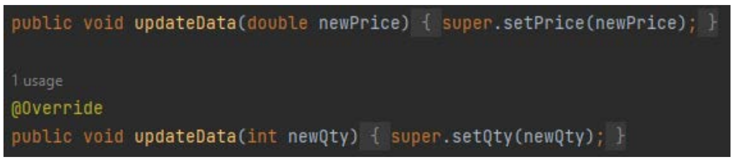
```
public void updateData(double newPrice) { super.setPrice(newPrice); }

1 usage
@Override
public void updateData(int newQty) { super.setQty(newQty); }
```

   c) Drink
      Line 11 – 16 :



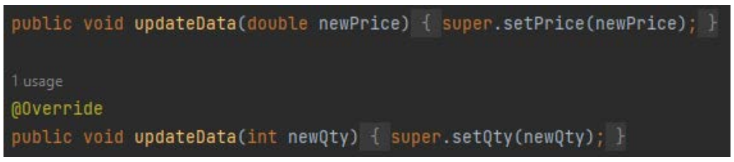
```
public void updateData(double newPrice) { super.setPrice(newPrice); }

1 usage
@Override
public void updateData(int newQty) { super.setQty(newQty); }
```

   d) Food
      Line 11 – 16 :



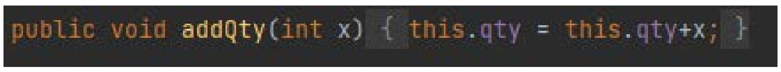
```
public void updateData(double newPrice) { super.setPrice(newPrice); }

1 usage
@Override
public void updateData(int newQty) { super.setQty(newQty); }
```

   e) Other
      Line 11 – 16 :



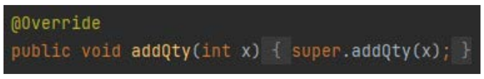
```
public void updateData(double newPrice) { super.setPrice(newPrice); }

1 usage
@Override
public void updateData(int newQty) { super.setQty(newQty); }
```

4. Overriding
   a) Item
      Line 39 :



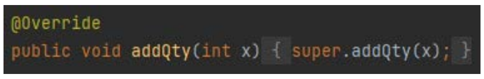
```
public void addQty(int x) { this.qty = this.qty+x; }
```

   b) Dessert
      Line 20 – 21 :



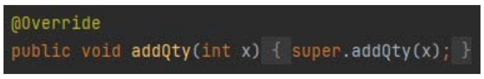
```
@Override
public void addQty(int x) { super.addQty(x); }
```

   c) Drink
      Line 20 – 21 :



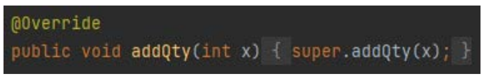
```
@Override
public void addQty(int x) { super.addQty(x); }
```

   d) Food
      Line 20 – 21 :



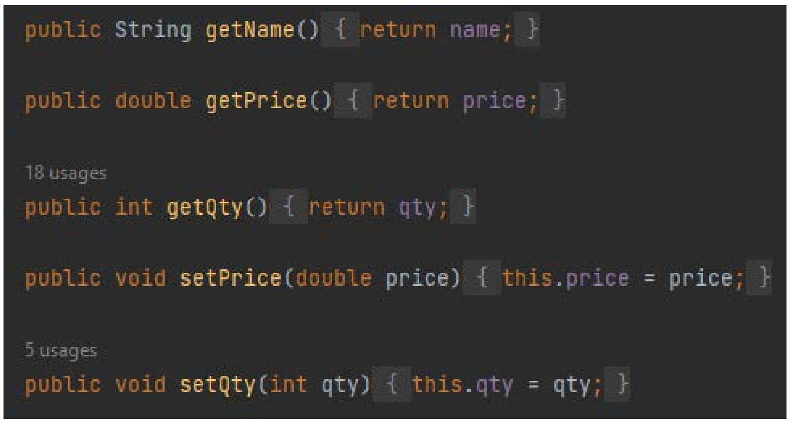
```
@Override
public void addQty(int x) { super.addQty(x); }
```

   e) Other
      Line 20 – 21 :



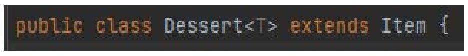
```
@Override
public void addQty(int x) { super.addQty(x); }
```

5. Encapsulation
   a) Item
      Line 19 – 35 :



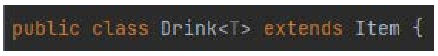
```
public String getName() { return name; }

public double getPrice() { return price; }

18 usages
public int getQty() { return qty; }

public void setPrice(double price) { this.price = price; }

5 usages
public void setQty(int qty) { this.qty = qty; }
```

6. Inheritance
   a) Dessert
      Line 3 :



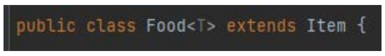
```
public class Dessert<T> extends Item {
```

   b) Drink
      Line 3 :



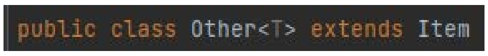
```
public class Drink<T> extends Item {
```

   c) Food
      Line 3 :



```
public class Food<T> extends Item {
```

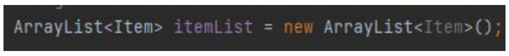
   d) Other
      Line 3 :



```
public class Other<T> extends Item
```

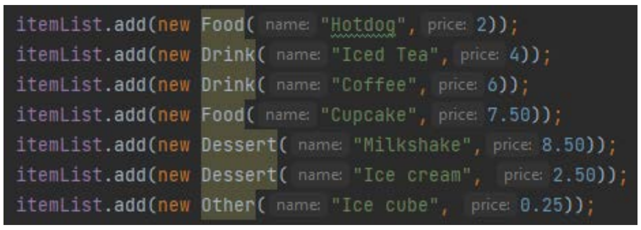
7. Polymorphism

a) Cashier Controller

Line 129 : 

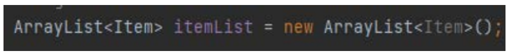
```
ArrayList<Item> itemList = new ArrayList<Item>();
```

Line 342 – 348 :



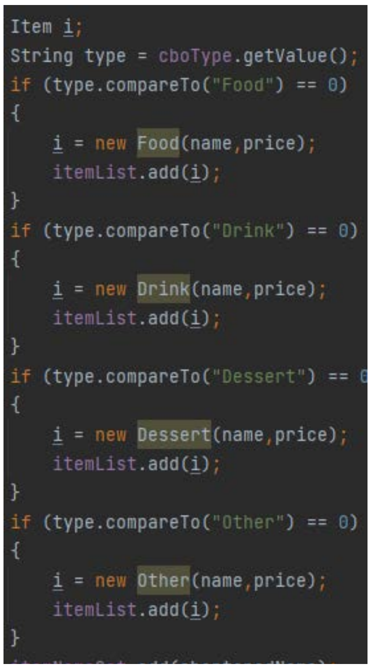
```
itemList.add(new Food( name: "Hotdog", price: 2));
itemList.add(new Drink( name: "Iced Tea", price: 4));
itemList.add(new Drink( name: "Coffee", price: 6));
itemList.add(new Food( name: "Cupcake", price: 7.50));
itemList.add(new Dessert( name: "Milkshake", price: 8.50));
itemList.add(new Dessert( name: "Ice cream", price: 2.50));
itemList.add(new Other( name: "Ice cube", price: 0.25));
```

b) Product Add Controller

Line 60 : 

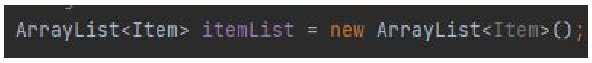
```
ArrayList<Item> itemList = new ArrayList<Item>();
```

Line 72 – 93 :



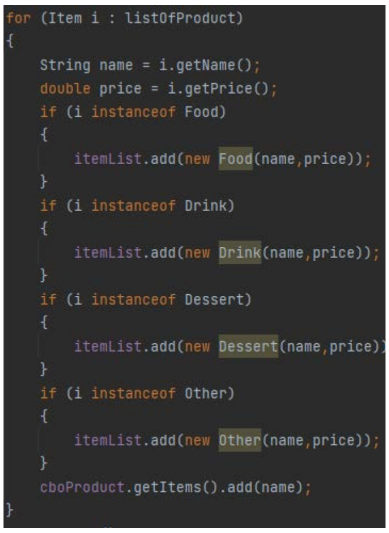
```
Item i;
String type = cboType.getValue();
if (type.compareTo("Food") == 0)
{
    i = new Food(name,price);
    itemList.add(i);
}
if (type.compareTo("Drink") == 0)
{
    i = new Drink(name,price);
    itemList.add(i);
}
if (type.compareTo("Dessert") == 
{
    i = new Dessert(name,price);
    itemList.add(i);
}
if (type.compareTo("Other") == 0)
{
    i = new Other(name,price);
    itemList.add(i);
}
```

c) Product Delete Controller

Line 65 : 

```
ArrayList<Item> itemList = new ArrayList<Item>();
```

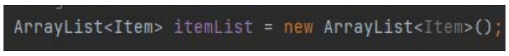
Line 216 – 235 :



```
for (Item i : listOfProduct)
{
    String name = i.getName();
    double price = i.getPrice();
    if (i instanceof Food)
    {
        itemList.add(new Food(name,price));
    }
    if (i instanceof Drink)
    {
        itemList.add(new Drink(name,price));
    }
    if (i instanceof Dessert)
    {
        itemList.add(new Dessert(name,price))
    }
    if (i instanceof Other)
    {
        itemList.add(new Other(name,price));
    }
    cboProduct.getItems().add(name);
}
```

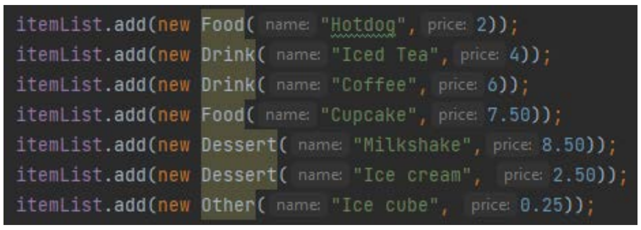
8. ArrayList

a) Cashier Controller

Line 129 : 

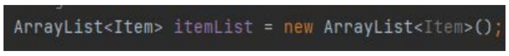
```
ArrayList<Item> itemList = new ArrayList<Item>();
```

Line 342 – 348 :



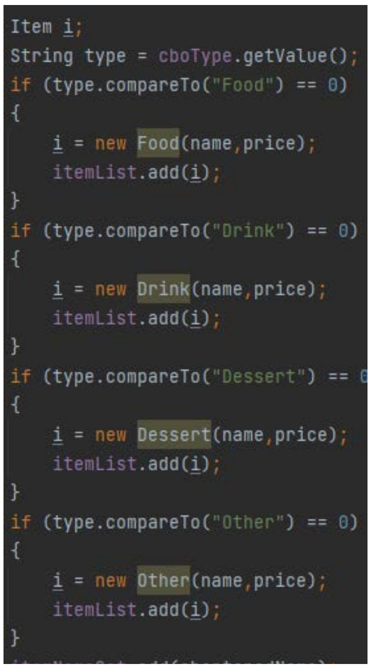
```
itemList.add(new Food( name: "Hotdog", price: 2));
itemList.add(new Drink( name: "Iced Tea", price: 4));
itemList.add(new Drink( name: "Coffee", price: 6));
itemList.add(new Food( name: "Cupcake", price: 7.50));
itemList.add(new Dessert( name: "Milkshake", price: 8.50));
itemList.add(new Dessert( name: "Ice cream", price: 2.50));
itemList.add(new Other( name: "Ice cube", price: 0.25));
```

b) Product Add Controller

Line 60 : 

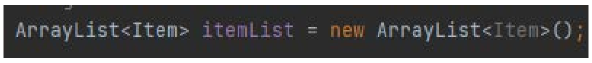
```
ArrayList<Item> itemList = new ArrayList<Item>();
```

Line 72 – 93 :



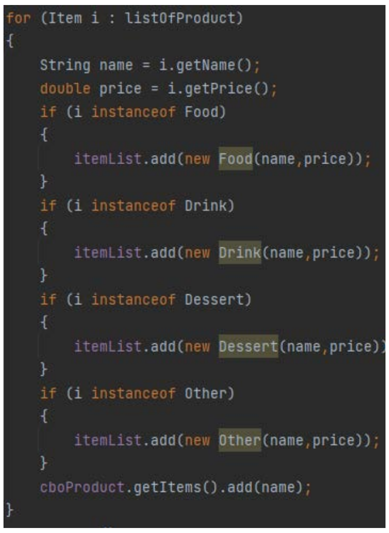
```
Item i;
String type = cboType.getValue();
if (type.compareTo("Food") == 0)
{
    i = new Food(name,price);
    itemList.add(i);
}
if (type.compareTo("Drink") == 0)
{
    i = new Drink(name,price);
    itemList.add(i);
}
if (type.compareTo("Dessert") == 
{
    i = new Dessert(name,price);
    itemList.add(i);
}
if (type.compareTo("Other") == 0)
{
    i = new Other(name,price);
    itemList.add(i);
}
```

c) Product Delete Controller

Line 65 : 

```
ArrayList<Item> itemList = new ArrayList<Item>();
```

Line 216 – 235 :

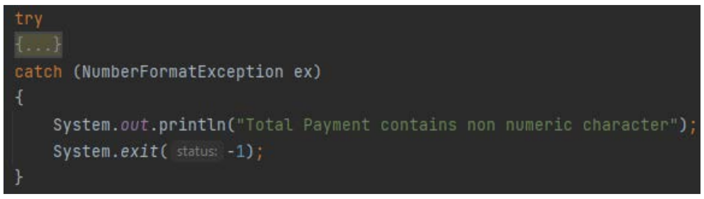


```
for (Item i : listOfProduct)
{
    String name = i.getName();
    double price = i.getPrice();
    if (i instanceof Food)
    {
        itemList.add(new Food(name,price));
    }
    if (i instanceof Drink)
    {
        itemList.add(new Drink(name,price));
    }
    if (i instanceof Dessert)
    {
        itemList.add(new Dessert(name,price))
    }
    if (i instanceof Other)
    {
        itemList.add(new Other(name,price));
    }
    cboProduct.getItems().add(name);
}
```

9. Exception Handling

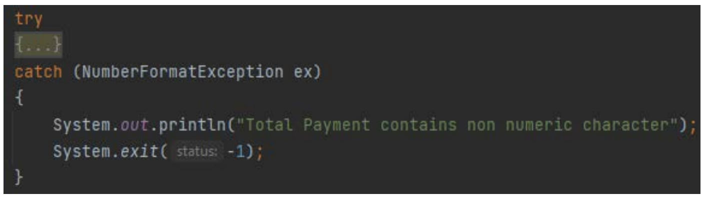
a) Cashier Controller

Line 133 – 161 :



```
try
{...}
catch (NumberFormatException ex)
{
    System.out.println("Total Payment contains non numeric character");
    System.exit( status: -1);
}
```

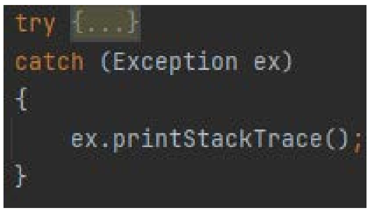
Line 297 – 365 :



```
try
{...}
catch (NumberFormatException ex)
{
    System.out.println("Total Payment contains non numeric character");
    System.exit( status: -1);
}
```

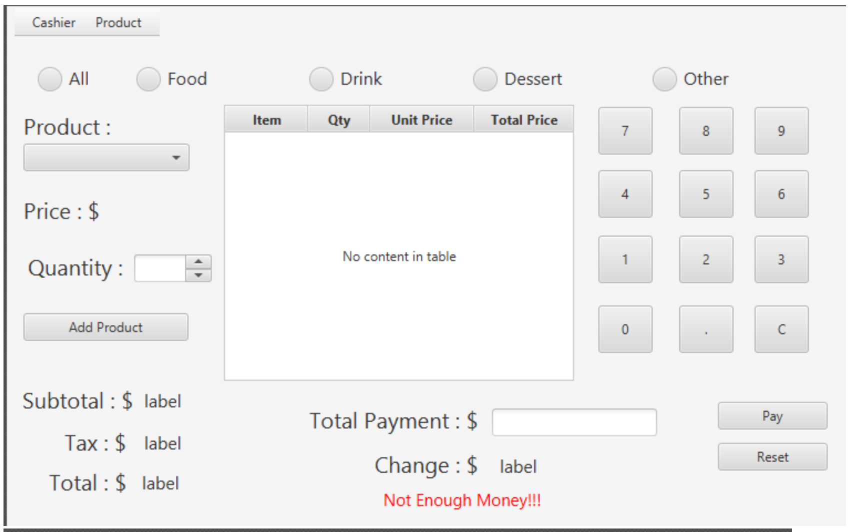
b) Product Add Controller

Line 65 – 109 :



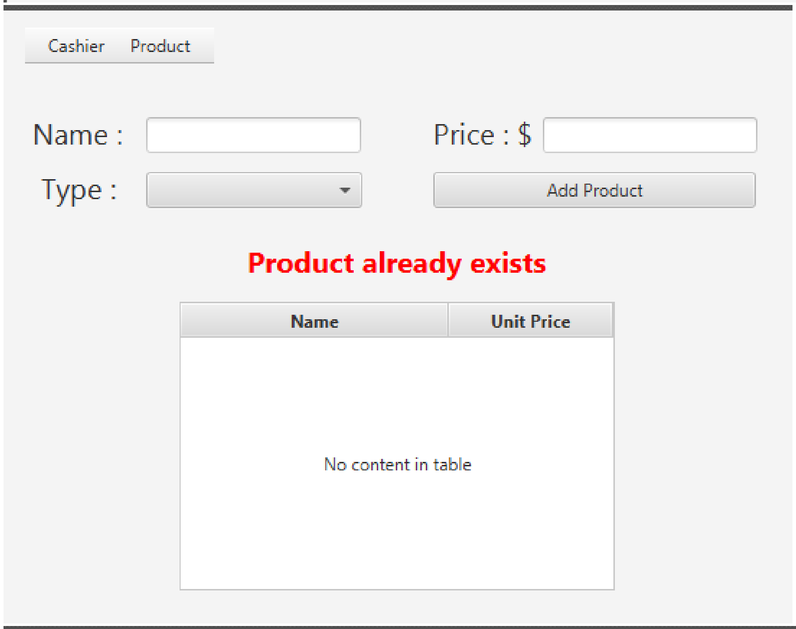
```
try {...}
catch (Exception ex)
{
    ex.printStackTrace();
}
```

10. GUI



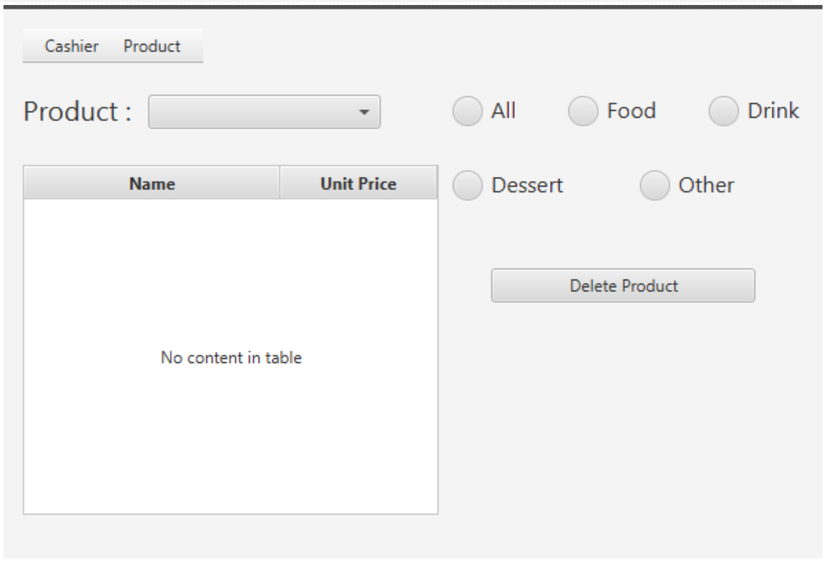
No content in table

Not Enough Money!!!



Product already exists

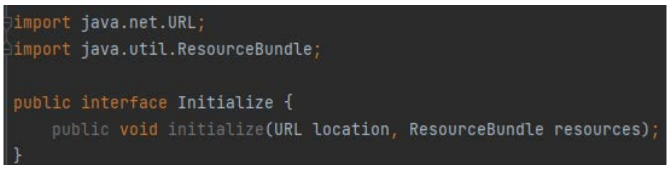
No content in table



No content in table

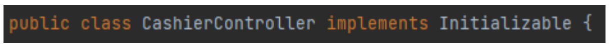
11. Interface

a) Initialize :



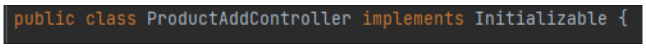
```
import java.net.URL;
import java.util.ResourceBundle;

public interface Initialize {
    public void initialize(URL location, ResourceBundle resources);
}
```

b) Cashier Controller

Line 23 : 

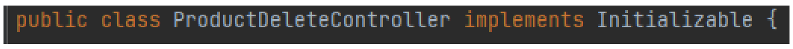
```
public class CashierController implements Initializable {
```

c) Product Add Controller

Line 25 : 

```
public class ProductAddController implements Initializable {
```

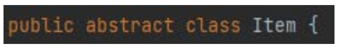
d) Product Delete Controller

Line 22 : 

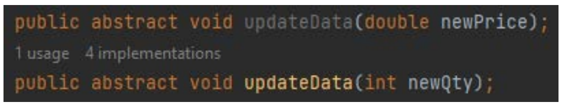
```
public class ProductDeleteController implements Initializable {
```

12. Abstract Class

a) Item

Line 3 : 

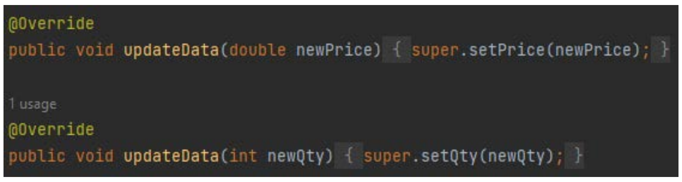
```
public abstract class Item {
```

Line 15 – 16 : 

```
public abstract void updateData(double newPrice);
1 usage   4 implementations
public abstract void updateData(int newQty);
```

b) Dessert

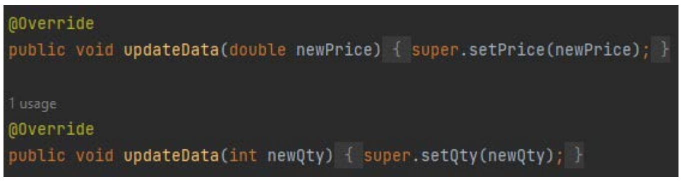
Line 10 – 16 :



```
@Override
public void updateData(double newPrice) { super.setPrice(newPrice); }

1 usage
@Override
public void updateData(int newQty) { super.setQty(newQty); }
```

c) Drink

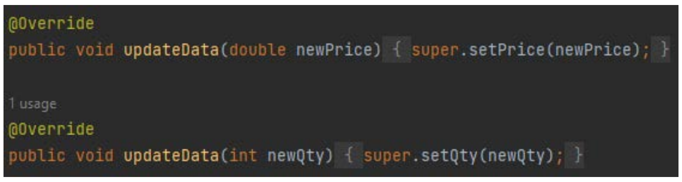
Line 10 – 16 :



```
@Override
public void updateData(double newPrice) { super.setPrice(newPrice); }

1 usage
@Override
public void updateData(int newQty) { super.setQty(newQty); }
```

d) Food

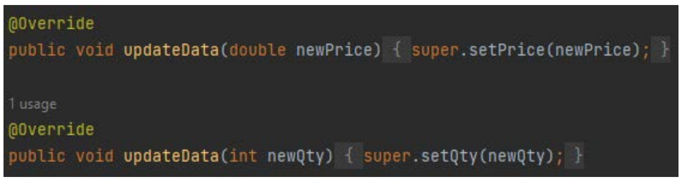
Line 10 – 16 :



```
@Override
public void updateData(double newPrice) { super.setPrice(newPrice); }

1 usage
@Override
public void updateData(int newQty) { super.setQty(newQty); }
```

e) Other

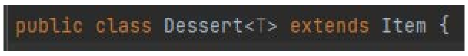
Line 10 – 16 :



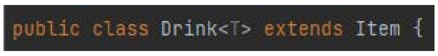
```
@Override
public void updateData(double newPrice) { super.setPrice(newPrice); }

1 usage
@Override
public void updateData(int newQty) { super.setQty(newQty); }
```

13. Generics

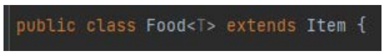
a) Dessert

Line 3 : 
```java
public class Dessert<T> extends Item {
```

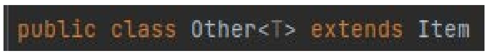
b) Drink

Line 3 : 
```java
public class Drink<T> extends Item {
```

c) Food

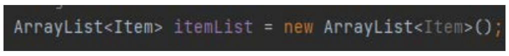
Line 3 : 
```java
public class Food<T> extends Item {
```

d) Other

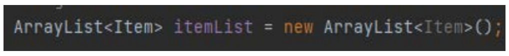
Line 3 : 
```java
public class Other<T> extends Item
```

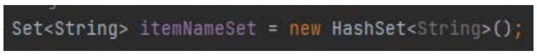
14. Collection

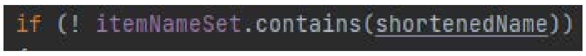
a) Cashier Controller

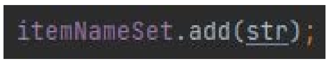
Line 129 : 
```java
ArrayList<Item> itemList = new ArrayList<Item>();
```

b) Product Add Controller

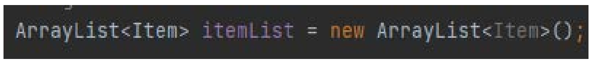
Line 60 : 
```java
ArrayList<Item> itemList = new ArrayList<Item>();
```

Line 61 : 
```java
Set<String> itemNameSet = new HashSet<String>();
```

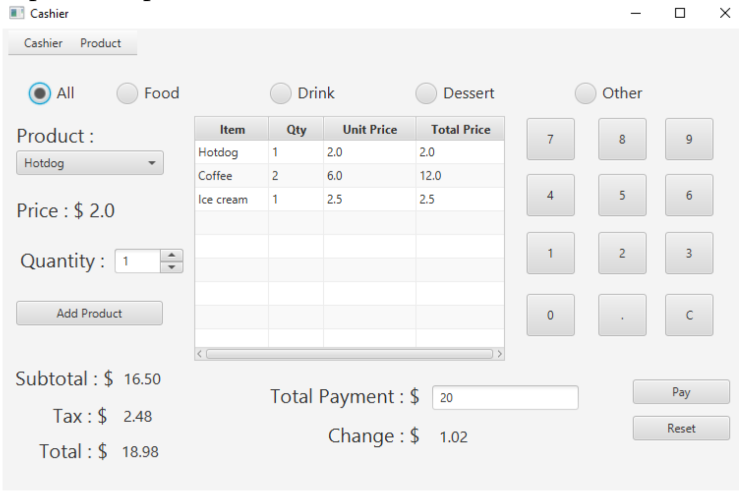
Line 70 : 
```java
if (! itemNameSet.contains(shortenedName))
```

Line 251 : 
```java
itemNameSet.add(str);
```

c) Product Delete Controller

Line 65 : 
```java
ArrayList<Item> itemList = new ArrayList<Item>();
```

15. Input / Output



Cashier

Cashier Product

[✓ All] Food Drink Dessert Other

Product : Hotdog

Price : $ 2.0

Quantity : 1

Add Product

Item	Qty	Unit Price	Total Price
Hotdog	1	2.0	2.0
Coffee	2	6.0	12.0
Ice cream	1	2.5	2.5

7	8	9
4	5	6
1	2	3
0	.	C

Subtotal : $ 16.50

Tax : $ 2.48

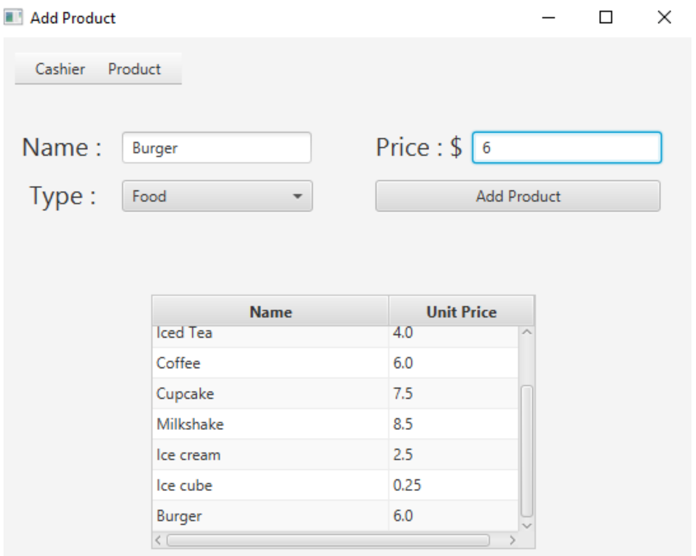
Total : $ 18.98

Total Payment : $ 20

Change : $ 1.02

Pay

Reset



Add Product

Cashier Product

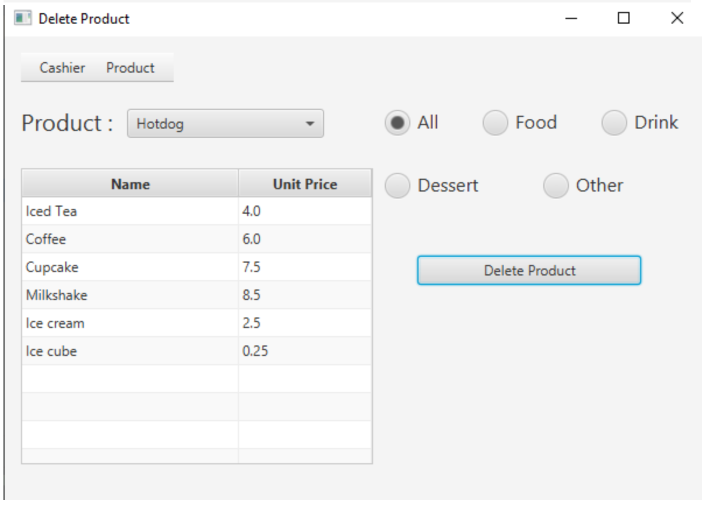
Name : Burger

Price : $ 6

Type : Food

Add Product

Name	Unit Price
Iced Tea	4.0
Coffee	6.0
Cupcake	7.5
Milkshake	8.5
Ice cream	2.5
Ice cube	0.25
Burger	6.0



Delete Product

Cashier Product

Product : Hotdog

[✓ All] Food Drink Dessert Other

Name	Unit Price
Iced Tea	4.0
Coffee	6.0
Cupcake	7.5
Milkshake	8.5
Ice cream	2.5
Ice cube	0.25

Delete Product