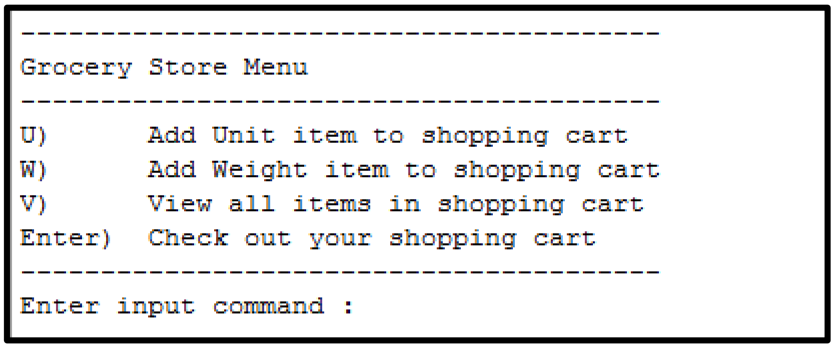
# Exercise2: Inheritance, Abstract, Polymorphism

## 1 Problem Statement: Grocery store

A grocery store sells many different items, where some of them are sold by units, while the others are sold by weights (kilograms). There are many vending machines installed in the store for customers to check out their shopping cart by themselves. Write a "Point-Of-Sale" (POS) program that is used in the vending machine. Figure 1 shows the main menu of the program composing of 4 options: U, W, V, and Enter.

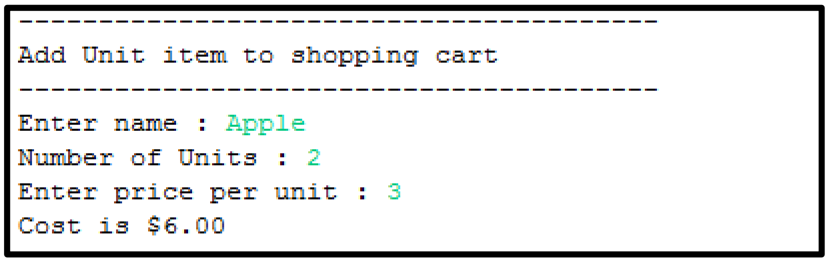
| | |
|---|---|
| • Option "U" | - add "item sold by units" to the shopping list |
| • Option "W" | - add "item sold by weights" to the shopping list |
| • Option "V" | - view all items in the shopping list |
| • Enter | - check out all items in the shopping list |



```
----------------------------------------
Grocery Store Menu
----------------------------------------
U)      Add Unit item to shopping cart
W)      Add Weight item to shopping cart
V)      View all items in shopping cart
Enter)  Check out your shopping cart
----------------------------------------
Enter input command :
```

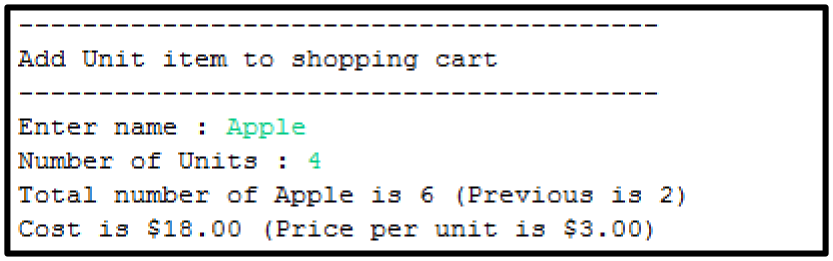
Figure 1. POS's main menu.

For option "U" (unit item), customers must be add to add **new unit item** (Figure 2) and add **existing unit item** (Figure 3) to the shopping list.



```
----------------------------------------
Add Unit item to shopping cart
----------------------------------------
Enter name : Apple
Number of Units : 2
Enter price per unit : 3
Cost is $6.00
```

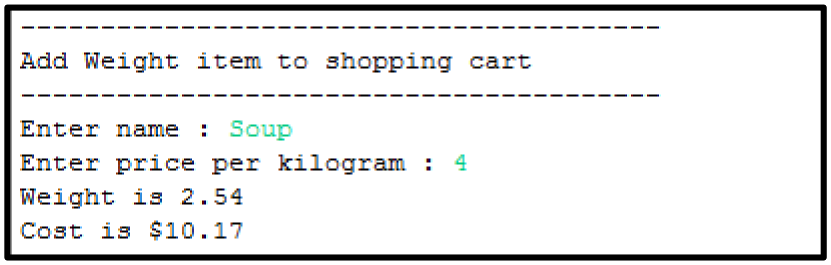
Figure 2. Add new "unit item"



```
----------------------------------------
Add Unit item to shopping cart
----------------------------------------
Enter name : Apple
Number of Units : 4
Total number of Apple is 6 (Previous is 2)
Cost is $18.00 (Price per unit is $3.00)
```

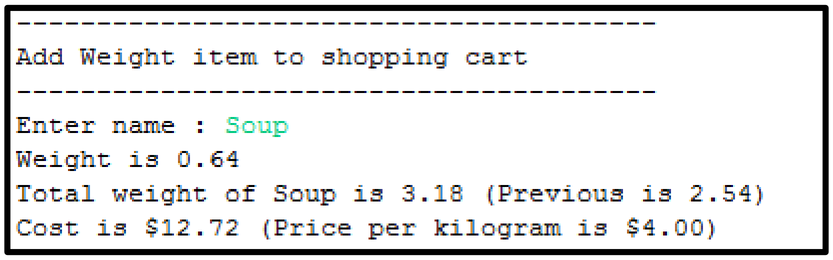
**Figure 3. Add existing "unit item"**

For option "W" (weight item), customers must be add to add **new weight item** (Figure 4) and add **existing weight item** (Figure 5) to the shopping list.



```
----------------------------------------
Add Weight item to shopping cart
----------------------------------------
Enter name : Soup
Enter price per kilogram : 4
Weight is 2.54
Cost is $10.17
```

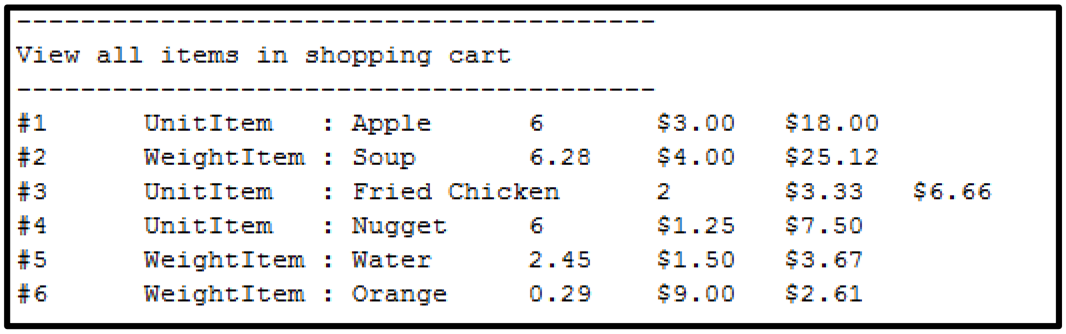
**Figure 4. Add new "weight item"**



```
----------------------------------------
Add Weight item to shopping cart
----------------------------------------
Enter name : Soup
Weight is 0.64
Total weight of Soup is 3.18 (Previous is 2.54)
Cost is $12.72 (Price per kilogram is $4.00)
```

**Figure 5. Add existing "weight item"**

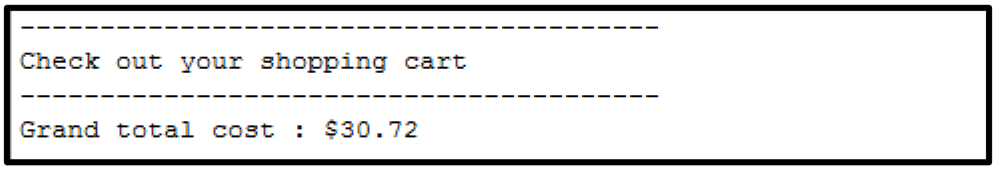
For option "V" (view), customers can show the list of all items in the shopping list (Figure 6)



```
----------------------------------------
View all items in shopping cart
----------------------------------------
#1      UnitItem   : Apple      6       $3.00   $18.00
#2      WeightItem : Soup       6.28    $4.00   $25.12
#3      UnitItem   : Fried Chicken      2       $3.33   $6.66
#4      UnitItem   : Nugget     6       $1.25   $7.50
#5      WeightItem : Water      2.45    $1.50   $3.67
#6      WeightItem : Orange     0.29    $9.00   $2.61
```

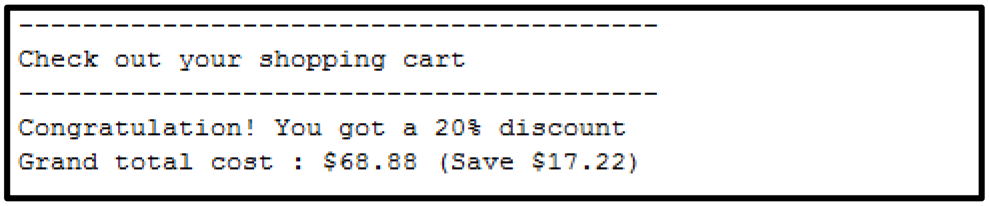
**Figure 6. View all items in the shopping cart**

After finish adding all items, the option “Enter” is used to check out. The program will show the total cost (Figure 7), the total cost with discount (Figure 8). Note that the promotion condition is shown in Figure 9.



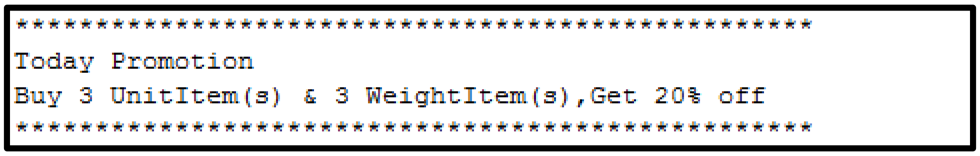
```
----------------------------------------
Check out your shopping cart
----------------------------------------
Grand total cost : $30.72
```

Figure 7. “Total cost” in the checkout menu



```
----------------------------------------
Check out your shopping cart
----------------------------------------
Congratulation! You got a 20% discount
Grand total cost : $68.88 (Save $17.22)
```

Figure 8. “Total cost with discount” in the checkout menu



```
**************************************************
Today Promotion
Buy 3 UnitItem(s) & 3 WeightItem(s),Get 20% off
**************************************************
```

Figure 9. Promotion condition

## 2 Implementation Detail

Figure 10 shows the class diagram of the POS program. There are 5 classes including Item, UnitItem, WeightItem, GroceryStore, and ShoppingCart.

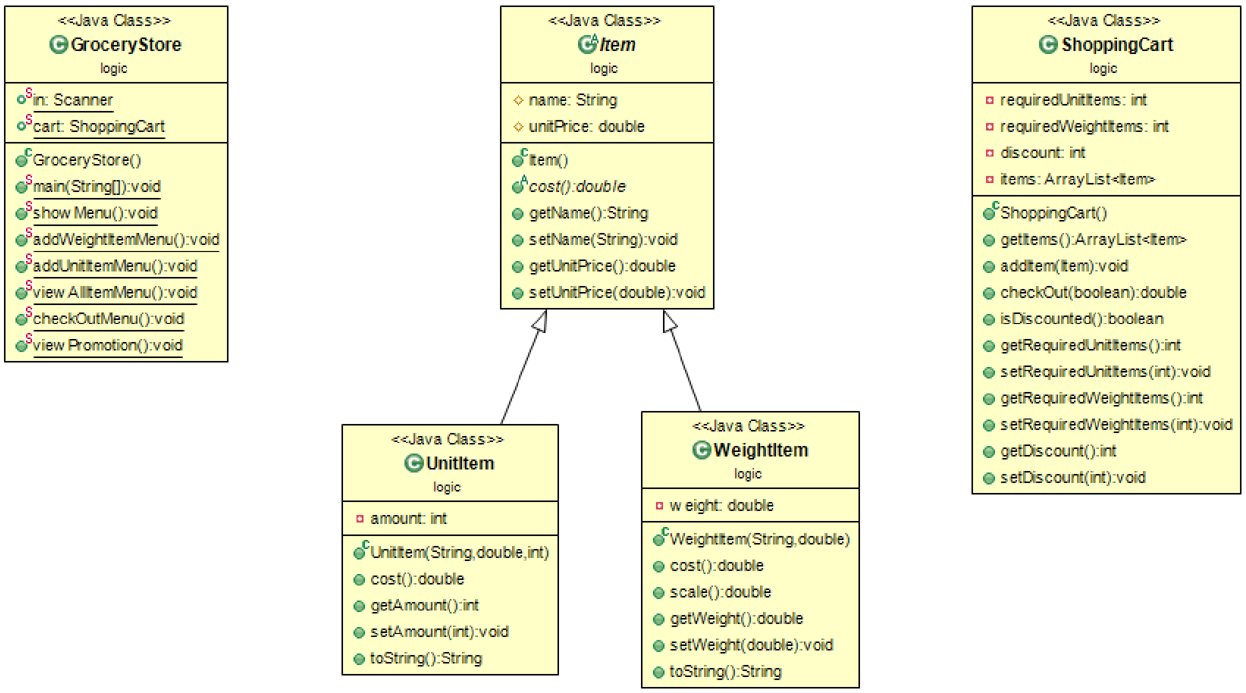


**Figure 10. Class diagram of the POS program**

### 2.1 Class GroceryStore

#### 2.1.1 Field

- public static ShopppingCart cart

#### 2.1.2 Method

- public static void main() - To show all available menu, get user input command and execute the command repeatedly until the shopping cart is checked out.
- public static void showMenu() - To show all menus (Figure 1)
- public static void addUnitItemMenu() - **/*Fill Code*/** To ask for name, unitPrice and amount of UnitItem and add it to the shopping cart. (Figure 2) **If the item's name is already existed, don't ask for unitPrice and increase the amount of existed item instead. (Figure 3)**
- public static void addWeightItemMenu() - **/*Fill Code*/** To ask for name and unitPrice of WeightItem and add it to the shopping cart (Figure 4) **If the item's name is already existed, don't ask for unitPrice increase the weight of existed item by the result of scale() instead. (Figure 5)**

- public static void viewAllItemsMenu() - **/*Fill Code*/** To list all items in the shopping cart including name, unitPrice and amount/weight as shown in Figure 6
- public static void checkOutMenu() - **/*Fill Code*/** To return the total cost of all items in the shopping cart with discount included (if the prerequisite is met). (Figure 7, Figure 8)
- public static void viewPromotion() - To display the prerequisite for a discount. (Figure 9)

## 2.2 Class Item (No code provided)

### 2.2.1 Field

- protected String name
- protected double unitPrice

### 2.2.2 Method

- public abstract double cost() - To calculate the total cost of the item.
- getters & setters

## 2.3 Class UnitItem (No code provided)

### 2.3.1 Field

- private int amount

### 2.3.2 Constructor

- public UnitItem(String name, double unitPrice, int amount)

### 2.3.3 Method

- public double cost() - To calculate the total cost of the item, multiply the unitPrice by the amount.
- public String toString() - To return the string as the following format
  “UnitItem : {name} {amount} {unitPrice} {cost}”
  EX. “UnitItem : Apple 2 3.11 6.22”
- getters & setters

## 2.4 Class WeightItem (No code provided)

### 2.4.1 Field

- private double weight

### 2.4.2 Constructor

- public WeightItem(String name, double unitPrice) - set the weight to the result of scale()

### 2.4.3 Method

- public double cost() - To calculate the total cost of the item, multiply the unitPrice by the weight.
- public double scale() - To return the value in range of 0.01 and 4.00 RANDOMLY.
- public String toString() - To return the string as the below format

  "WeightItem : "{name} {weight} {unitPrice} {cost}"

  EX. "WeightItem : Soup 1.63 5.25 8.56"
- getters & setters

## 2.5 Class ShoppingCart

### 2.5.1 Field

- private int requiredUnitItems - The number of different UnitItem required in shopping cart for a discount
- private int requiredWeightItems - The number of different WeightItem required in shopping cart for a discount
- private int discount - The discount rate
- private ArrayList<Item> items - The array contains all items in shopping cart.

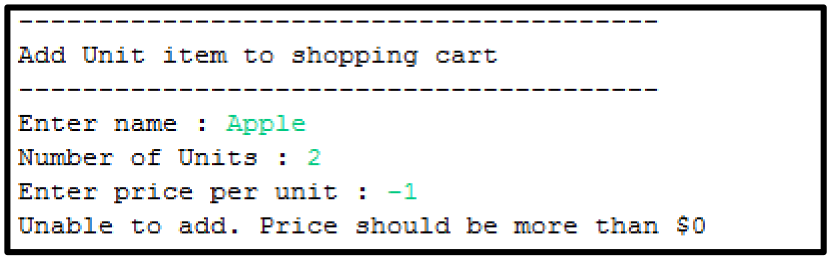
### 2.5.2 Constructor

- public ShoppingCart() - Initialize all fields. Set the discount to 20 and set the requiredUnitItems and the requiredWeightItem in range of 1 – 3.

### 2.5.3 Method

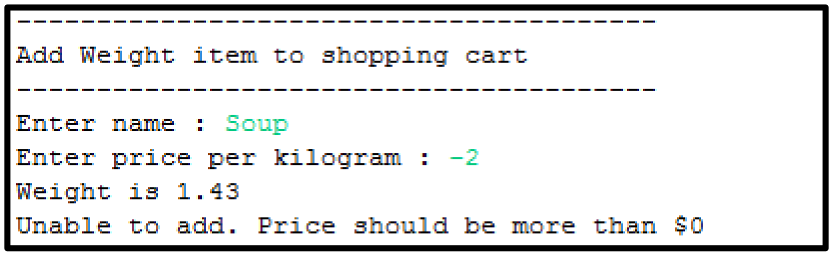
- public void addItem(Item item) - **/*Fill Code*/** To add the item to the shopping cart if it is enable.

  **UnitItem is unable to add if unitPrice or amount is equal or less than 0 (Figure 11). WeightItem is unable to add if unitPrice is equal or less than 0 (Figure 12).**
- public double checkOut() - **/*Fill Code*/** To return the total cost of all items in the shopping cart. Returns 0 if the shopping is empty.
- public boolean isDiscount() - **/*Fill Code*/** To return true, if the discount prerequisite is met.
- getters & setters



```
----------------------------------------
Add Unit item to shopping cart
----------------------------------------
Enter name : Apple
Number of Units : 2
Enter price per unit : -1
Unable to add. Price should be more than $0
```

**Figure 11. Add Unit Item (Unsuccessful)**



```
----------------------------------------
Add Weight item to shopping cart
----------------------------------------
Enter name : Soup
Enter price per kilogram : -2
Weight is 1.43
Unable to add. Price should be more than $0
```

**Figure 12. Add Weight Item (Unsuccessful)**

## 3 Instruction

- Create a project "Exercise2_2014-2_{ID}" to implement the POS program
- Use the provided JUnit test case in the folder "test" (ShoppingCartTest.java, UnitItemTest.java, WeightItemTest.java) to check your implementation
- After finishing the program and passing all test cases, export your project into a jar file with sources called "Exercise2_2014-2_{ID}.jar"