



INDIVIDUAL ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

AAPP004-4-2-JP

JAVA PROGRAMMING

UCDF1805ICT (SE)

HAND OUT DATE: 20th FEBRUARY 2020

HAND IN DATE: 20th APRIL 2020

WEIGHTAGE: 60%

INSTRUCTIONS TO CANDIDATES:

- 1 Students are advised to underpin their answers with the use of references (cited using the Harvard Name System of Referencing).**
- 2 Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.**
- 3 Cases of plagiarism will be penalised.**
- 4 The assignment should be bound in an appropriate style (comb bound or stapled).**
- 5 Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.**

NAME: LIM JINQ KUEN TP051241

LECTURER: DR. MIKE ONG TEONG JOO

Table of Contents

Introduction.....	4
Sample outputs and Explanation.....	5
1. Start up.....	5
2. Menu.....	5
3. Insert coins & coins validation	6
4. Dispense and Return changes	6
5. Enter admin site	7
6. Restock.....	7
7. Deduct stock.....	8
8. Edit drinks.....	8
9. Delete drinks	9
10. Change password.....	9
11. Logout (admin).....	10
12. Running out of coins (for changes).....	10
13. Refill coins	11
14. Shutdown machine	11
Sample code	12
Encapsulation.....	12
1. Class.....	13
2. Variables	14
3. Constructor	14
4. Method.....	14
5. Setter.....	14
6. Getter	14
Input (scanner)	15
Array	15
Loop.....	16
While loop & Do while loop.....	16
For loop.....	17
Exception	17
Throws	17
Try catch	18

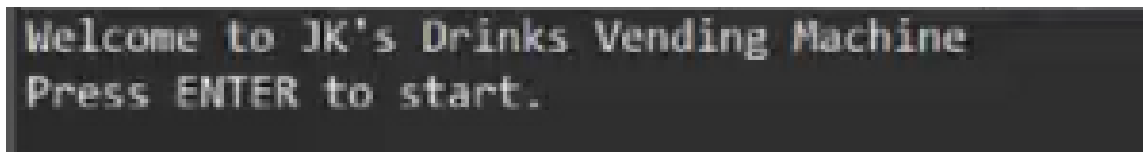
Polymorphism	18
Method overloading	18
Method overriding	19
Abstraction	19
Inheritance.....	19
Additional features.....	20
Allocate text file.....	20
Read data from text file.....	20
Write data to text file	21
Return changes based on stock of coins	21
Purchase history	22
Data validation	22
Assumption	24
References	25

Introduction

This documentation is explaining about the development of a vending machine that created by using the OOP concepts in Java including encapsulation, abstraction, inheritance and polymorphism. It explains the sample output, user manual, sample code explaining the OOP concepts and some additional features incorporated with the system. Some screenshots are included in the documentation.

Sample outputs and Explanation

1. Start up

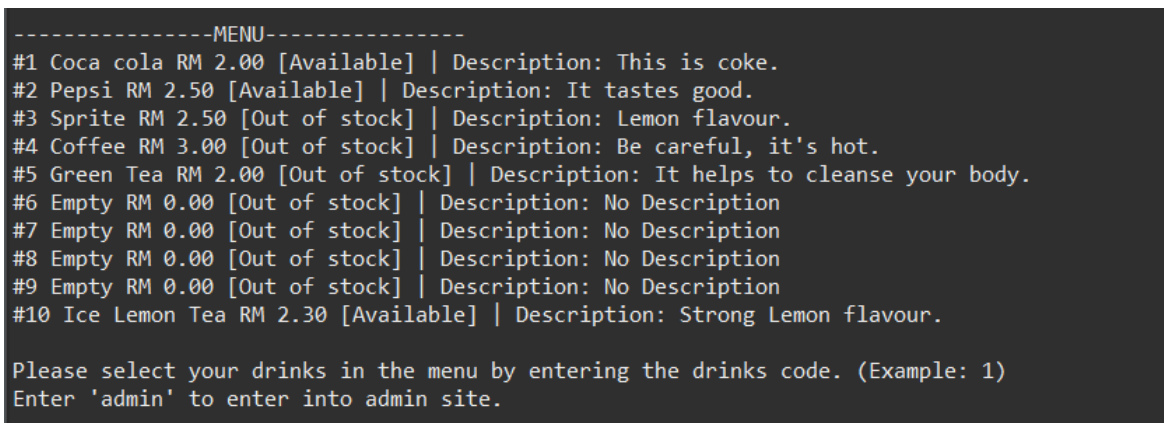


```
Welcome to JK's Drinks Vending Machine
Press ENTER to start.
```

Diagram 1: Start up

At first, welcome message will be shown once the machine started and user need to press enter to continue.

2. Menu



```
-----MENU-----
#1 Coca cola RM 2.00 [Available] | Description: This is coke.
#2 Pepsi RM 2.50 [Available] | Description: It tastes good.
#3 Sprite RM 2.50 [Out of stock] | Description: Lemon flavour.
#4 Coffee RM 3.00 [Out of stock] | Description: Be careful, it's hot.
#5 Green Tea RM 2.00 [Out of stock] | Description: It helps to cleanse your body.
#6 Empty RM 0.00 [Out of stock] | Description: No Description
#7 Empty RM 0.00 [Out of stock] | Description: No Description
#8 Empty RM 0.00 [Out of stock] | Description: No Description
#9 Empty RM 0.00 [Out of stock] | Description: No Description
#10 Ice Lemon Tea RM 2.30 [Available] | Description: Strong Lemon flavour.

Please select your drinks in the menu by entering the drinks code. (Example: 1)
Enter 'admin' to enter into admin site.
```

Diagram 2: Drinks menu

After pressing the 'enter', a drinks menu will be shown. User can choose any drinks from the menu unless the drink is out of stock or empty. For admin, admin site can be entered from here, admin just have to type 'admin' to gain access into the admin site with a correct password entered.

3. Insert coins & coins validation

```
Please select your drinks in the menu by entering the drinks code. (Example: 1)
Enter 'admin' to enter into admin site.
1
Please insert coins. Selected drinks: coca cola. Amount To Pay: RM 2.50.
Enter 'cancel' to go back to menu.
1
Please insert coins. Selected drinks: coca cola. Amount To Pay: RM 1.50.
Enter 'cancel' to go back to menu.
abc
Only 1, 5, 10, 0.1, 0.2, 0.5 are accepted!
Enter 'cancel' to go back to menu.
2
Only 1, 5, 10, 0.1, 0.2, 0.5 are accepted!
Enter 'cancel' to go back to menu.
1
Please insert coins. Selected drinks: coca cola. Amount To Pay: RM 0.50.
Enter 'cancel' to go back to menu.
1
```

Diagram 3: Prompt for coins and data validation

Once the user selected a drink, then the system will prompt for money. Some data validation has been done here, user can only insert 1, 5, 10, 0.1, 0.2 and 0.5 into the system. And if the user suddenly changes his/her mind they can type 'cancel' to go back to drinks selection.

4. Dispense and Return changes

```
1
|Thank you for using me :), Here's your drinks:
|/#####\
|-----|
|#####|
|# coca cola #
|#####|
|-----|
|\#####/
|
|Here's your changes: 0 x 10 sen, 0 x 20 sen, 1 x 50 sen.
|Have a nice day.
|##### NEXT TRANSACTION #####
```

Diagram 4: Dispense drink and refund changes

Once the amount for the drink has collected sufficiently, the drink will be dispensed from the machine and return the changes if any. Then proceed to next transaction.

5. Enter admin site

```
Please select your drinks in the menu by entering the drinks code. (Example: 1)
Enter 'admin' to enter into admin site.
admin
Password:
123
Correct Password!

##### ADMIN #####

#1 Coca cola RM 2.00 [Available] Stock: 9 | Description: This is coke.
#2 Pepsi RM 2.50 [Available] Stock: 10 | Description: It tastes good.
#3 Sprite RM 2.50 [Out of stock] Stock: 0 | Description: Lemon flavour.
#4 Coffee RM 3.00 [Out of stock] Stock: 0 | Description: Be careful, it's hot.
#5 Green Tea RM 2.00 [Out of stock] Stock: 0 | Description: It helps to cleanse your body.
#6 Empty RM 0.00 [Out of stock] Stock: 0 | Description: No Description
#7 Empty RM 0.00 [Out of stock] Stock: 0 | Description: No Description
#8 Empty RM 0.00 [Out of stock] Stock: 0 | Description: No Description
#9 Empty RM 0.00 [Out of stock] Stock: 0 | Description: No Description
#10 Ice Lemon Tea RM 2.30 [Available] Stock: 9 | Description: Strong Lemon flavour.

Please select a drink in the menu to edit by entering the drinks code. (Example: 1)
Enter 'coin' to refill coins.
Enter 'password' to change admin password.
Enter 'history' to view transaction history.
Enter 'logout' to logout.
Enter 'shutdown' to turn off the machine.
```

Diagram 5: Entering admin site

As it mentioned above, admin site can be accessed during the drink selection state, so if the user enters 'admin' the machine prompt for admin password. If the password is valid then it will go to the admin site, but if is not then it goes back to drink selection state. In admin site, there are few functions such as restock, edit, delete, refill coins, change password, view history, logout and shutdown machine.

6. Restock

```
#10 Ice Lemon Tea RM 2.30 [Available] Stock: 9 | Description: Strong Lemon flavour.

Please select a drink in the menu to edit by entering the drinks code. (Example: 1)
Enter 'coin' to refill coins.
Enter 'password' to change admin password.
Enter 'history' to view transaction history.
Enter 'logout' to logout.
Enter 'shutdown' to turn off the machine.
3
Please enter 'restock' or 'edit' or 'delete' to continue.
restock
Enter the restock amount. Use negative number to deduct stock.
10
Updated successfully!

##### ADMIN #####

#1 Coca cola RM 2.00 [Available] Stock: 9 | Description: This is coke.
#2 Pepsi RM 2.50 [Available] Stock: 10 | Description: It tastes good.
#3 Sprite RM 2.50 [Available] Stock: 10 | Description: Lemon flavour.
```

Diagram 6: Restocking

At first, admin has to select a drink code, after that, then enter what they want to do with the drink like restock. Once the admin entered 'restock', it will prompt for number of stock and once the number of stocks is entered, the stock of the selected drink will be updated. As you can see, by default in diagram 5 the stock for Sprite is 0 and in diagram 6 it has restocked to 10.

7. Deduct stock

For deduct stock, it is using the same way as restock. Select drink > 'restock' > enter a negative value to deduct stock.

8. Edit drinks

```
Please select a drink in the menu to edit by entering the drinks code. (Example: 1)
Enter 'coin' to refill coins.
Enter 'password' to change admin password.
Enter 'history' to view transaction history.
Enter 'logout' to logout.
Enter 'shutdown' to turn off the machine.
5
Please enter 'restock' or 'edit' or 'delete' to continue.
edit
Enter a new drink name: Tea
Enter a new drink description: It is just a normal tea.
Enter a new price: 1.5
Edited successfully!

##### ADMIN #####

#1 Coca cola RM 2.00 [Available] Stock: 9 | Description: This is coke.
#2 Pepsi RM 2.50 [Available] Stock: 10 | Description: It tastes good.
#3 Sprite RM 2.50 [Available] Stock: 10 | Description: Lemon flavour.
#4 Coffee RM 3.00 [Out of stock] Stock: 0 | Description: Be careful, it's hot.
#5 Tea RM 1.50 [Out of stock] Stock: 0 | Description: It is just a normal tea.
```

Diagram 7: Editing drink details

Same goes to edit, so the admin has to select a drink first then only start editing the drink. Select a drink > enter 'edit' > enter a new name > enter a new description > enter a new price. In the example, it selected the fifth drink for editing and in diagram 5 the name for fifth drink is 'Green Tea', but after the editing it has changed to 'Tea' in diagram 7.

9. Delete drinks

```
Enter 'shutdown' to turn off the machine.
10
Please enter 'restock' or 'edit' or 'delete' to continue.
delete
|
##### ADMIN #####

#1 Coca cola RM 2.00 [Available] Stock: 9 | Description: This is coke.
#2 Pepsi RM 2.50 [Available] Stock: 10 | Description: It tastes good.
#3 Sprite RM 2.50 [Available] Stock: 10 | Description: Lemon flavour.
#4 Coffee RM 3.00 [Out of stock] Stock: 0 | Description: Be careful, it's hot.
#5 Tea RM 1.50 [Out of stock] Stock: 0 | Description: It is just a normal tea.
#6 Empty RM 0.00 [Out of stock] Stock: 0 | Description: No Description
#7 Empty RM 0.00 [Out of stock] Stock: 0 | Description: No Description
#8 Empty RM 0.00 [Out of stock] Stock: 0 | Description: No Description
#9 Empty RM 0.00 [Out of stock] Stock: 0 | Description: No Description
#10 Empty RM 0.00 [Out of stock] Stock: 0 | Description: No Description
```

Diagram 8: Deleting drink

For deletion of drink, the process is Select a drink > enter 'delete'. That's it, the drink slot has now change to empty. In diagram 8, since the admin has selected the tenth drink and now everything is changed to empty for tenth drink details which means it is successfully deleted the drink.

10. Change password

```
Please select a drink in the menu to edit by entering the drinks code. (Example: 1)
Enter 'coin' to refill coins.
Enter 'password' to change admin password.
Enter 'history' to view transaction history.
Enter 'logout' to logout.
Enter 'shutdown' to turn off the machine.
password
New password:
12345678
```

Diagram 9: Changing password

In this machine, there is also a change password function provided in admin site. So, the admin just has to 'password' to change a new password for admin site. Once the admin entered 'password' the system will asked for the new password and after the new password is entered the password for admin site will be updated.

11. Logout (admin)

```
Please select a drink in the menu to edit by entering the drinks code. (Example: 1)
Enter 'coin' to refill coins.
Enter 'password' to change admin password.
Enter 'history' to view transaction history.
Enter 'logout' to logout.
Enter 'shutdown' to turn off the machine.
logout
Welcome to JK's Drinks Vending Machine
Press ENTER to start.

-----MENU-----
#1 coke RM 2.00 [Available]
#2 Empty RM 0.00 [Out of stock]
#3 sprite RM 4.50 [Out of stock]
#4 coffee RM 5.00 [Out of stock]
#5 tea RM 2.00 [Out of stock]
#6 empty RM 0.00 [Out of stock]
#7 empty RM 0.00 [Out of stock]
#8 empty RM 0.00 [Out of stock]
#9 empty RM 0.00 [Out of stock]
#10 tehtarik RM 2.00 [Available]

Please select your drinks in the menu by entering the drinks code. (Example: 1)
Enter 'admin' to enter into admin site.
```

Diagram 10: Logout from the admin site

Once the admin has finished with the editing, he/she can logout from the admin site by entering 'logout'. After that, they will be redirected back to the normal user site.

12. Running out of coins (for changes)

```
Please select your drinks in the menu by entering the drinks code. (Example: 1)
Enter 'admin' to enter into admin site.
1
|
##### OUT OF SERVICE #####
## INSUFFICIENT COINS FOR CHANGES ##

Welcome to JK's Drinks Vending Machine
Press ENTER to start.
```

Diagram 11: Machine is running out of coins for changes

If the sum of coins in the machine is not exceed RM 8.00, the machine will become out of service. And it only starts working once the admin has refilled the coin with a total more than RM 8.00.

13. Refill coins

```
Please select a drink in the menu to edit by entering the drinks code. (Example: 1)
Enter 'coin' to refill coins.
Enter 'password' to change admin password.
Enter 'history' to view transaction history.
Enter 'logout' to logout.
Enter 'shutdown' to turn off the machine.
coin
Please enter the amount of 10 sen you want to refill:
10
Please enter the amount of 20 sen you want to refill:
10
Please enter the amount of 50 sen you want to refill:
10
```

Diagram 12: Refilling coins into the machine

Before refilling the coins, it has entered into the admin site. The steps are Enter into admin site > enter 'coin' > enter the amount of 10 sen > enter the amount of 20 sen > enter the amount of 50 sen. After that, logout from the admin site and the machine will now working fine again.

14. Shutdown machine

```
Please select your drinks in the menu by entering the drinks code. (Example: 1)
Enter 'admin' to enter into admin site.
admin
Password:
123
Correct Password!

##### ADMIN #####

#1 Coca cola RM 2.00 [Available] Stock: 9 | Description: This is coke.
#2 Pepsi RM 2.50 [Available] Stock: 10 | Description: It tastes good.
#3 Sprite RM 2.50 [Available] Stock: 10 | Description: Lemon flavour.
#4 Coffee RM 3.00 [Out of stock] Stock: 0 | Description: Be careful, it's hot.
#5 Tea RM 1.50 [Out of stock] Stock: 0 | Description: It is just a normal tea.
#6 Empty RM 0.00 [Out of stock] Stock: 0 | Description: No Description
#7 Empty RM 0.00 [Out of stock] Stock: 0 | Description: No Description
#8 Empty RM 0.00 [Out of stock] Stock: 0 | Description: No Description
#9 Empty RM 0.00 [Out of stock] Stock: 0 | Description: No Description
#10 Empty RM 0.00 [Out of stock] Stock: 0 | Description: No Description

Please select a drink in the menu to edit by entering the drinks code. (Example: 1)
Enter 'coin' to refill coins.
Enter 'password' to change admin password.
Enter 'history' to view transaction history.
Enter 'logout' to logout.
Enter 'shutdown' to turn off the machine.
shutdown
The system is shutting down.
```

Diagram 13: Shutdown the machine

The steps to turn off the machine: Get into the admin site > enter 'shutdown'. Once the machine is showing 'The system is shutting down.', it means the machine is turned off.

Sample code

Before I start explaining on my code, I would like to brief on how classes are allocated. First, I have created 5 classes VendingMachine, Coinchecker, Drinks, Inventory and Description.

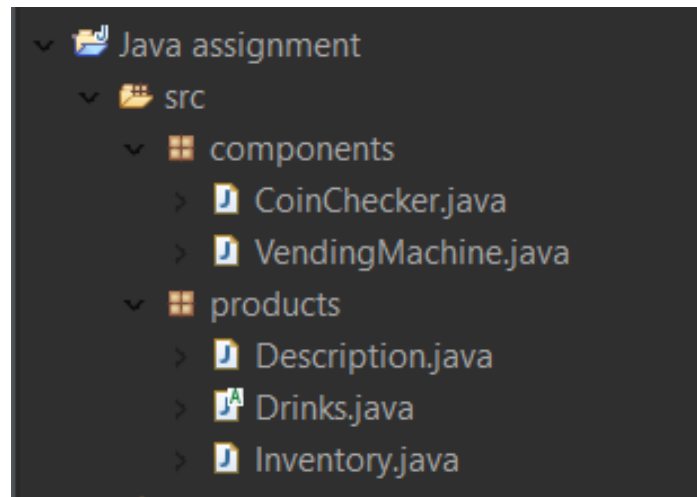


Diagram 14: Project file

As you can see above, there are two packages created which are components and products. Components package is storing all the classes that related to the machine while products package is storing every classes related to drinks. And there are also some text files being stored in the components package which are coins.txt, drinks.txt, history.txt and password.txt, the evidence is shown below.

JK (D:) > eclipse-workspace > Java assignment > src > components > data				
Name	^	Date modified	Type	Size
coins		9/4/2020 10:27 PM	Text Document	1 KB
drinks		9/4/2020 10:27 PM	Text Document	1 KB
history		9/4/2020 10:27 PM	Text Document	2 KB
password		9/4/2020 10:03 PM	Text Document	1 KB

Diagram 15: Text files location

That's all for the briefing on the project structures.

Encapsulation

According to (tutorialspoint, n.d.), encapsulation is one of the four fundamental OOP concepts in Java. Encapsulation means wrapping the variables and methods in a class and all the data (variables) can only be accessed by calling the method in the class. I will explain

how I implement it in this assignment down below. For this section, I am taking inventory class as the example to explain.

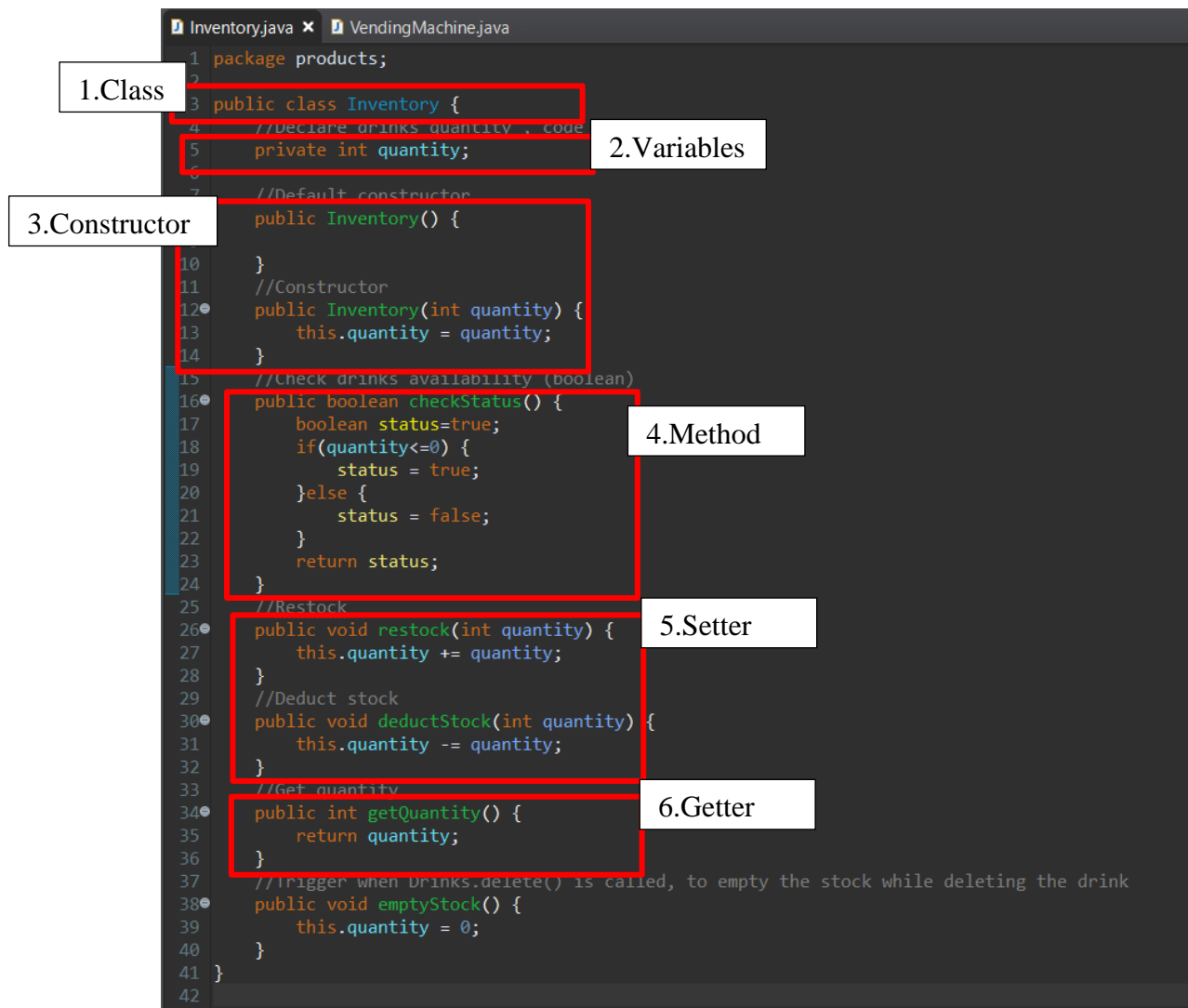


Diagram 16: Inventory class

1. Class

As you can see in diagram 16, I have created public class for inventory. Class is actually a ‘blueprint’, taking words from my lecturer Dr. Mike and (ww3schools, n.d.) and I agree with this. For example, I have created an inventory class which means whenever I instantiate a new object using inventory class it means an inventory object is being created. The properties of created object from a class is exactly the same as what it is declared in the class, so it is a good evidence saying that a class is a ‘blueprint’

2. Variables

In this class, I also declared a variable in global which is quantity. It means whenever I instantiate an inventory class there will be a variable for storing the value of quantity in the object. Variables are where the data stored in an object based on the declaration in class. And the variable I have declared it as protected which means it can only be accessed by its method.

3. Constructor

A constructor is a method that used in instantiation, so whenever you are trying to 'new' an object you need to call the constructor method. In diagram 16, I have created two constructors, one is without parameter while the another one is with a parameter. It means whenever I tried to instantiate an inventory object, I have two ways, first I can instantiate it without any quantity specified which is 0 by default and second, I can instantiate it with a value of quantity.

4. Method

For method, I have created a method called "checkStatus". This method is to check whether there is stock for the particular drinks. In every method you have specify the method return type, it can be void, int, String, float and so on, even it can be class type that created by your own. So, in my code I use boolean as my return type for this method, so at the end of this method I have to return a value with either true or false.

5. Setter

Setter is a method that used to set value for private or protected variable in encapsulation concept. In diagram 16, I have created two methods for restock and deduct stock. Restock method is used to add number of stocks into the system which is used in admin site while deduct stock method is used to deduct stock by 1 whenever user bought a drink.

6. Getter

Getter is a method that used to get value from private or protected variable in encapsulation concept. Based my code, I have created a method called "getQuantity", it is used to get the value of quantity from the inventory object since the quantity variable is declared as protected.

Input (scanner)

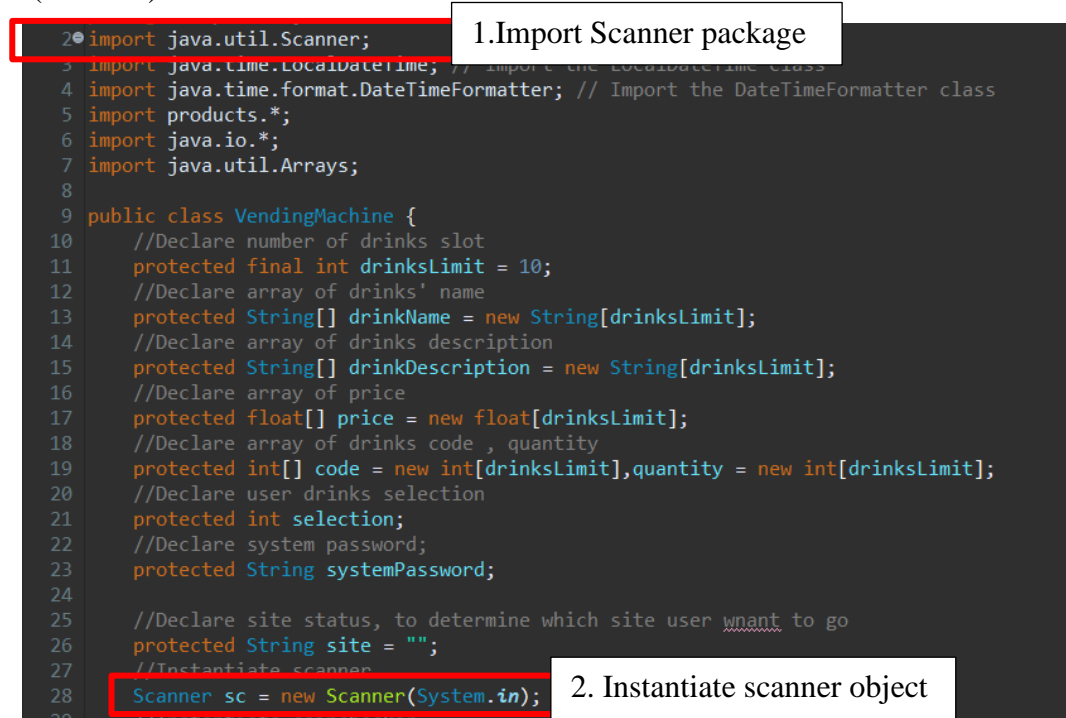


Diagram 17: Declaring scanner object

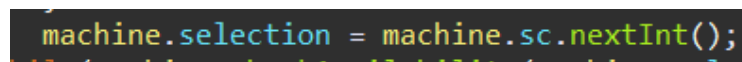


Diagram 18: Use scanner for input

Scanner is one of the famous class used for input. At first, I import the scanner into my main class first and then instantiate it afterward. After that, I use its methods for requesting input from user such next(), nextInt() and nextFloat(). Different methods are used for different data type of input. I have used this method for any input in the machine whenever it is needed.

Array

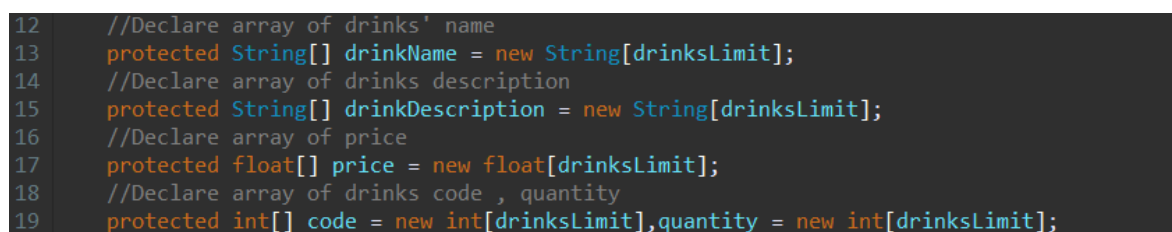


Diagram 19: Declaring variables for array

Array is implemented in my system. Array are used to store multiple value in a single variable (w3schools, n.d.). Since I have multiple drinks in the machine class, that is the reason I used array for storing the data of the drinks.

Loop

Loop is a way to execute a block of code or lines of code more than once depends on the boolean condition.

While loop & Do while loop

```
492 //System loop
493 while(true) {
494     //Print start up message
495     machine.startup();
496     //Display drinks menu
497     for(int i=0;i<drinkNum;i++) {
498         System.out.println(machine.displayMenu(machine.drinks[i].getName(), machine.drinks[i].
499     }
```

Diagram 20: While loop

I used while loop for the entire purchasing process so that it is loopable even when the user finishes the purchase it still can proceed to next purchase. It will only break of loop when 'exit' is called in admin site.

```
501 //If 'admin' entered break the loop
502 do{
503     machine.promptForSelection();
504     //If 'admin' entered break the loop
505     if(machine.site.equals("admin")) {
506         break;
507     }
508     machine.selection = machine.sc.nextInt();
509 }while(machine.checkAvailability(machine.selection,drinkNum));
```

Diagram 21: Do while loop

Do while loop is similar to while loop, the only different is it won't on the boolean condition at the first time, it will start checking on the condition after. The reason I used it is because I

don't have to check the amount before I start inserting coin, so it will only start checking on the balance after the user inserted some money.

For loop

```
//Insert data into Drinks type
for(int i=0;i<drinkNum;i++) {
    machine.drinks[i] = new Drinks(drinkName[i],price[i],code[i]);
    machine.inventory[i] = new Inventory(code[i],quantity[i]);
}
```

Diagram 22: For loop

For loop is unlike from while loop and do while loop it will only execute number of times specified in the loop. I use for loop here is because I know the number of drinks I have so for loop is more suitable in this situation.

Exception

An exception is an unwanted or unexpected event, which occurs during the execution of a program, for instance, at run time, that disrupts the normal flow of the program's instructions (GeeksForGeeks, n.d.). There are two ways to handle exception which are throw and try catch.

Throws

```
162 //Dispense drink
163 public String drinkDispense(int code) throws FileNotFoundException {
164     //deduct product's quantity
165     inventory[code-1].deductStock(1);
166     quantity[code-1]-=1;
167     //update data
168     updateData(drinkName,drinkDescription,price,this.code,quantity);
169     return "Thank you for using me :), Here's your drinks:\n\n"+drinks[selection-1].gene
170 }
```

Diagram 23: Throws exception

I use this throw exception to prevent an exception occurred when the file is not found in updateData() method.

Try catch

```
320 //Print history
321 public void printHistory() {
322     Scanner data = null;
323     try {
324         data = new Scanner(setFilePath("history.txt"));
325         System.out.println("\nHISTORY\n");
326         while(data.hasNextLine()) {
327             System.out.println(data.nextLine());
328         }
329     }
330     catch(FileNotFoundException e) {
331         System.out.println(e);
332     }
333     finally {
334         if(data != null) {
335             //Close file
336             data.close();
337         }
338     }
339 }
```

Diagram 24: Try catch

Try catch is good in exception handling. I use try catch in printHistory() it is because the code here is trying to overwrite to a text file. So, if there is any exception or error occurred the text file is still able to close, and it is easier for programmer to find out the error made.

Polymorphism

Polymorphism means "many forms" (w3schools, n.d.).

Method overloading

```
34 //Constructor
35 VendingMachine() {
36     this.drinkName = new String[drinksLimit] ;
37     this.price = new float[drinksLimit];
38     this.code = new int[drinksLimit];
39     this.quantity = new int[drinksLimit];
40     this.systemPassword = "123";
41 }
42 VendingMachine(String[] drinkName, float[] price,int[] code,int[] quantity,String password) {
43     this.drinkName = drinkName;
44     this.price = price;
45     this.code = code;
46     this.quantity = quantity;
47     this.systemPassword = password;
48 }
```

Diagram 25: Method overloading

Method overloading means there is few methods sharing same name in a class but with different of parameters. As you can see, there is no parameter for my first constructor method but there five parameters in the second one. Although they are having the same name but they

are actually not the same, they might work something similar but different parameter or return type.

Method overriding

```
11 }  
12 @Override  
13 public String toString() {  
14     return " | Description: "+description+"\n";  
15 }
```

Diagram 26: Method overriding

I have implemented method overriding in my system. It is because there is a ‘freebie’ created by default and due to this I need to override the method. Method overriding means overriding method of the super class.

Abstraction

```
2  
3 public abstract class Drinks {  
4     //Declare drink name  
5     protected String name;
```

Diagram 27: Abstraction of drinks class

Since I want to prevent the drinks class being directly called in the main class, therefore, I use abstract in class drinks. Abstraction is to prevent the class or method from being called directly which means it can only be called in subclass which extended the class.

Inheritance

```
2  
3 public class Description extends Drinks{  
4     String description;
```

Diagram 28: Implementing inheritance

Inheritance in OO concept is all about subclass and parent class (super class). Since I have grouped my description class as child class of drinks class therefore, I extend my description class to drinks class. Although the way I implement this concept is incorrect, it is because my machine is allowed to change drinks and it is not suitable to create subclasses for drinks class. The reason I create this description class is to show I am understanding its concept but just it is not suitable for my system.

Additional features

Allocate text file

```
48 }
49 public static File setFilePath(String filename) {
50     //Set relative file path
51     String filepath = new File("").getAbsolutePath()+"\\src\\components\\data\\"+filename;
52     //Instantiate File
53     File file = new File(filepath);
54     return file;
55 }
```

Diagram 29: Method for locating the text file

Before I instantiate a file object, 'java.io.*' has to be imported before its instantiation. After that, I use the file method called 'getAbsolutePath()' to get the file path of the project file followed by the sub file path for the text file. And return it as file object so that it can be used directly in Scanner and Print Writer.

Read data from text file

```
422 Scanner data = null;
423
424 //Declare name and price and code arrays
425 String[] drinkName = new String[10];
426 float[] price = new float[10];
427 int[] code = new int[10], quantity = new int[10];
428
429 try {
430     data = new Scanner(setFilePath("drinks.txt"));
431     //Read name from text file
432     drinkName = data.nextLine().split(",");
433
434     //Read price from text file
435     String[] p = data.nextLine().split(" ");
436     for(int i=0; i<drinkName.length; i++) {
437         price[i] = Float.parseFloat(p[i]);
438     }
439 }
```

Diagram 30: Reading data from drinks.txt

For scanning data from text file, it is quite similar to input scanning. Instead of putting System.in in the bracket use the file returned by the method in Diagram 29. And then use nextLine() method to get data line by line. Split() method is used to split the line of data into an array while 'Integer.parseInt()' is to convert the data type from string to integer, same goes to float type.

Write data to text file

```
199 //Update text file
200 public void updateData(String[] drinkName,String[] drinkDescription, float[] price,int[] code,int[] quantity) {
201     //Declare PrintWriter
202     PrintWriter pw = null;
203     try {
204         //Set file name
205         File file = setFilePath("drinks.txt");
206         //Allow PrintWriter overwrite the current text file
207         pw = new PrintWriter(new FileOutputStream(file, false));
208         //Store all the updated data into String
209         String newData = String.join(", ", drinkName)+"\n";
210         newData += String.join(", ", drinkDescription)+"\n";
211         String pricelist = Arrays.toString(price).replace(", ", "").replace("[", "").replace("]", "");
212         newData += pricelist+"\n";
213         String quantitylist = Arrays.toString(quantity).replace(", ", "").replace("[", "").replace("]", "");
214         newData += quantitylist+"\n";
215         String codelist = Arrays.toString(code).replace(", ", "").replace("[", "").replace("]", "");
216         newData += codelist;
217         //Overwrite the current file
218         pw.println(newData);
219     }
220 }
```

Diagram 31: Write data to drinks.txt

PrintWriter class has to be instantiated before using it. It is similar to the scanner class, put the file object returned by the setFilePath() method into the bracket of PrintWriter when it is being instantiated. And store all the data into a string and use println() method to overwrite the text file with the new data and close the PrintWriter afterward.

Return changes based on stock of coins

```
74 //Return changes
75 public int[] returnChanges() {
76     float changes = balance;
77     balance = 0;
78     int[] coins = new int[3];
79     while(changes > 0) {
80         if(numOf50sen > 0 && changes >= 0.5) {
81             changes -= 0.5f;
82             numOf50sen -= 1;
83             coins[2] += 1;
84         }
85         else if(numOf20sen > 0 && changes >= 0.2) {
86             changes -= 0.2f;
87             numOf20sen -= 1;
88             coins[1] += 1;
89         }
90         else if(numOf10sen > 0) {
91             changes -= 0.1f;
92             numOf10sen -= 1;
93             coins[0] += 1;
94         }
95         System.out.println(changes);
96     }
97     return coins;
98 }
99 }
```

Diagram 32: Return changes method

In Coinchecker class, I have declared three integer variables for storing the amount of coins for the machine. So, when 50 sen is still available, the changes will be mainly focus on

returning 50 sen unless the changes are smaller than 50 sen, same goes to the following 10 sen and 20 sen. There is also a text file storing the data of the coins' amount.

Purchase history

```
294 //Update history
295 public void updateHistory(int code) {
296     PrintWriter pw = null;
297     try {
298         File file = setFilePath("history.txt");
299         pw = new PrintWriter(new FileOutputStream(file, true));
300         LocalDateTime currentDateTime = LocalDateTime.now();
301         DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
302
303         String formattedDateTime = formatter.format(currentDateTime);
304         pw.println("Date time:" + formattedDateTime + " | "+"Selected drinks: "+drinkName[code-1]
305     }
306     catch(FileNotFoundException e) {
307         System.out.println(e);
308     }
309     finally {
310         if(pw != null) {
311             pw.close();
312         }
313     }
314 }
315 }
```

Diagram 33: Storing purchase history

For this method, since it is involving in date format, therefore, I have imported 'java.time.LocalDateTime' and 'java.time.DateTimeFormatter'. The 'java.time.LocalDateTime' is used to get the current date time while the 'java.time.DateTimeFormatter' is used to format the date time. In this method, PrintWriter class is also being used since it is writing to the text file. And by default, using PrintWriter it is overwriting the text file, so I use FileOutputStream class with a 'true' as its parameter to prevent it from overwriting the text file.

Data validation

```
97 //Ask user to make selection
98 public void promptForSelection() {
99     //Check if the input has integer, if not keep looping until it gets a integer
100     while(!sc.hasNextInt()) {
101         site = sc.next();
102         //If it gets string "admin" break the loop, and take user to admin site
103         if(site.equals("admin")) {
104             break;
105         }
106         System.out.println("Please enter a number.");
107     }
```

Diagram 34: Data validation of inputs

The way I did for all data validations in the system are the same. Before the user input, I use a while loop with hasNext() method to check whether the data type for input is correct. This

loop will only break once the data type of the input is correct or some keywords are matched. Once the loop is break, there will be a next() method to store the input into a variable.

Assumption

The assumption for this assignment is that inheritance might not be important in object-oriented concepts since sometimes we may not need to create any subclass in order to complete a project. For example, just like what I did in this assignment the subclass is not needed for drinks class due to the way of how my machine work but if it is a complex system then inheritance would be important. The most important concept in OO concepts would be encapsulation because that is where class, method and variable come from. Coding with the OO concepts is really helpful in building a software especially a huge program/ system. OO concepts to me is actually like we are creating our own data type like integer, string and any other data types, it is easier for me to learn OO when I realise this statement.

References

GeeksForGeeks, n.d.. *Exceptions in Java*. [Online]

Available at: <https://www.geeksforgeeks.org/exceptions-in-java/>

[Accessed 10 April 2020].

tutorialspoint, n.d.. *Java - Encapsulation*. [Online]

Available at: https://www.tutorialspoint.com/java/java_encapsulation.htm

[Accessed 10 April 2020].

w3schools, n.d.. *Java Arrays*. [Online]

Available at: https://www.w3schools.com/java/java_arrays.asp

[Accessed 10 April 2020].

w3schools, n.d.. *Java Polymorphism*. [Online]

Available at: https://www.w3schools.com/java/java_polymorphism.asp

[Accessed 10 April 2020].

ww3schools, n.d.. *Java Classes and Objects*. [Online]

Available at: https://www.w3schools.com/java/java_classes.asp

[Accessed 10 April 2020].