BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter
(Individual Work)

Student Information:	***Surname***	***Given Names***	***Student ID Number***
1.	*Roestam Moenaf*	*Rowin Faadhilah*	*2301944084*

Course Code	**:** COMP6510	**Course Name**	**:** Programming Languages
Class	**:** L2BC	**Name of Lecturer(s)**	**:** 1. Jude
Major	**:** Computer Science		
Title of Assignment	: Java Final Project		
Type of Assignment	**:** Final Project		
Submission Pattern			
Due Date	**:** 20th June 2020	**Submission Date**	**:** 16th June 2020

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.

2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

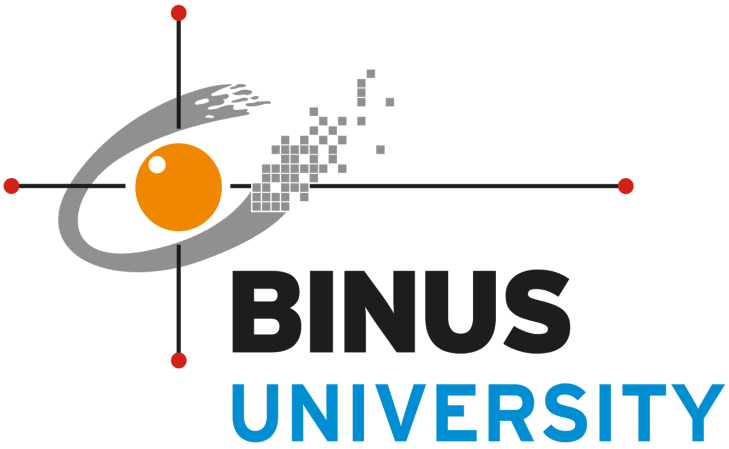
Declaration of Originality

By signing this assignment, I/we* understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I/we* declare that the work contained in this assignment is my/our* own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

(Rowin Faadhilah Roestam Moenaf)

Final Project Report
(Vending Machine)



By: Rowin Faadhilah Roestam Moenaf
NIM: 2031944084

Project Specification

This final project requires the student to create a program with Java. The program should demonstrate the knowledge that the student has acquired throughout the semester. The requirements of this project are set by the teacher and were used as guidance to finish the project.

The topic that was chosen for this project was vending machines. Vending machines have always been an iconic machine, It dispenses snacks and drinks wherever it is available. What is even more interesting is how it works, just by inputting through buttons it is able to identify what the user wants. The project will be based on this, a simple vending machine system using Java programming language. There will be a few more additions to the program instead of the basic function of a vending machine.

Solution Design

There are many vending machines that are being implemented today. Vending machines come in many different designs and for that, a design was chosen as the base of this project's vending machine. The type of vending machine that was going to be created in this project is a vending machine that sells two types of products which are snacks and drink. These products will be separated equally based on the number of items the vending machine could hold.

With this design, the requirements of the project can be reached. This design allows the use of inheritance and interfaces. List with a specified size will be used to contain the products depending on if they are a snack or a drink. There are a total of eight classes and one interface that will be used in this project. The first class is the vending machine class, this class contains the main functions of the vending machine. An account class was added to create a new method for the vending machine. The rest of the classes are mainly the products and the driver class of the vending machine, 2 being superclasses which are drinks and snacks, and the others are subclasses. IsProduct is the name of the interface that will be used by the superclasses as they are the products.

Discussion

Before the program starts the vending machine has to be initialized as well as its contents, and it will use addDrink() and addSnack() to add each product to its respective slots. The initialized products are randomized with different properties and names. There are also accounts already in the vending machine by using addAccount(). One of the accounts is an admin account which allows them to manage the vending machine.

When the program is run by the user, it will automatically display the contents of the vending machine, through displayContents(). After the display is run, the user would be able to choose an option between purchasing a product, creating an account, top up the user's account, or get product information. Once the user picks an option, a function of the chosen option would be run and the user would have to follow the steps. For the purchasing function, the user has the option to pay with either cash or their vending machine account, before paying they would be able to log in to a vending machine account if they have one. The login system only runs when the user chooses to pay with their account.

This vending machine system will run on a while loop, since most vending machines never closes or stops working, this way any user would be able to come and grab what they need from the vending machine and leave it on for the next customer to operate it. After every customer has finished using the vending machine, the system will save the account data of the users into a CSV file.

Code Images

I. Vending Machine Class

```
package VendingMachine;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class VendingMachine {
    private ArrayList<Account> UsersList = new ArrayList<~>();
    private Drink[][] DrinkList;
    private Snack[][] SnackList;
    private int register;
    private Drink selectedDrink;
    private Snack selectedSnack;

    public VendingMachine() {
        this.DrinkList = new Drink[3][6];
        this.SnackList = new Snack[3][6];
        this.register = 0;
    }

    public Drink getSelectedDrink() { return selectedDrink; }

    public Snack getSelectedSnack() { return selectedSnack; }
```

The image above shows all the libraries that are imported to complete the vending machine class. It also shows the properties of the vending machine class, the drinks and snacks are stored in two different multidimensional arrays, this allows the vending machine to be separated into two parts, drinks and snacks. The vending machine constructor initializes a vending machine with a 3 by 6 drink and snack slots the register is where the money that goes in the vending machine is stored.

```
public void setSelectedItem(String id) {
    switch (id.substring(0, 1).toUpperCase()) {
        case "A":
            this.selectedDrink = DrinkList[0][Integer.parseInt(id.substring(1)) - 1];
            System.out.println("Selected Drink: " + selectedDrink.toString());
            if (selectedDrink.getAmount() == 0) {
                System.out.println("Drink is unavailable, please select another item");
            }
            break;
        case "B":
            this.selectedDrink = DrinkList[1][Integer.parseInt(id.substring(1)) - 1];
            System.out.println("Selected Drink: " + selectedDrink.toString());
            if (selectedDrink.getAmount() == 0) {
                System.out.println("Drink is unavailable, please select another item");
            }
            break;
        case "C":
            this.selectedDrink = DrinkList[2][Integer.parseInt(id.substring(1)) - 1];
            System.out.println("Selected Drink: " + selectedDrink.toString());
            if (selectedDrink.getAmount() == 0) {
                System.out.println("Drink is unavailable, please select another item");
            }
            break;
        case "D":
            this.selectedSnack = SnackList[0][Integer.parseInt(id.substring(1)) - 1];
            System.out.println("Selected Snack: " + selectedSnack.toString());
            if (selectedSnack.getAmount() == 0) {
                System.out.println("Snack is unavailable, please select another item");
            }
            break;
```

The image above shows the setSelectedItem function. This function is the most impactful function of the vending machine. Its the function that finds an item based on its id and sets it as a chosen item.

```
public void addDrink(Drink drink) {
    int flag = 0;
    for (int i = 0; i < DrinkList.length; i++) {
        for (int j = 0; j < DrinkList[i].length; j++) {
            if (DrinkList[i][j] == null) {
                DrinkList[i][j] = drink;
                System.out.println("Drink has been added");
                flag = 1;
                break;
            } else if (i == DrinkList.length - 1 && j == DrinkList[i].length - 1) {
                System.out.println("Vending Machine drink slots are fully occupied");
                break;
            }
        }
        if (flag == 1) {
            break;
        }
    }
}
```

This shows the addDrink function, in this system, we have addDrink and addSnack functions which allows are the function that adds the product objects into the vending machine. addDrink and addSnack are the same functions but its only difference is that, addDrink would be used to add drink objects into the drink list, and addSnack is used to add snack objects into the snack list.

```java
public void addNewProduct(String id) { //adds a new product into a slot
    Scanner scan8 = new Scanner(System.in);
    Scanner scan9 = new Scanner(System.in);
    Scanner scan10 = new Scanner(System.in);
    if (id.substring(0, 1).toUpperCase().equals("A") || id.substring(0, 1).toUpperCase().equals("B") ||
            id.substring(0, 1).toUpperCase().equals("C")) {
        System.out.println("Product name: ");
        String name = scan8.nextLine();
        System.out.println("Product price: ");
        int price = scan8.nextInt();
        System.out.println("Product size: ");
        String size = scan9.nextLine();
        System.out.println("Product temperature: ");
        String temperature = scan10.nextLine();
        switch (id.substring(0, 1).toUpperCase()) {
            case "A":
                DrinkList[0][Integer.parseInt(id.substring(1)) - 1] = new Drink(name, price, size, temperature);
                System.out.println("Drink Added");
                break;
            case "B":
                DrinkList[1][Integer.parseInt(id.substring(1)) - 1] = new Drink(name, price, size, temperature);
                System.out.println("Drink Added");
                break;
            case "C":
                DrinkList[2][Integer.parseInt(id.substring(1)) - 1] = new Drink(name, price, size, temperature);
                System.out.println("Drink Added");
                break;
```

The image above shows our addNewProduct function. This function is used to list a new non-existing product into the vending machine system. It will ask for input to initialize the new product and it will sort if its a drink or snack based on the id of the slot. Only operated by the admin.

```java
public void displayContents() { // function to display contents of the vending machine
    System.out.println("Welcome to this Vending Machine");
    System.out.println("Vending Machine Contents");
    for (int i = 0; i < DrinkList.length; i++) {
        for (int j = 0; j < DrinkList[i].length; j++) { // loops the entire multidimensional array
            if (DrinkList[i][j] == null) {
                System.out.print("[This drink is out of stock], ");
            } else if (i == 0) {
                System.out.print("A" + (j + 1) + " " + DrinkList[i][j].getName() + ", ");
            } else if (i == 1) {
                System.out.print("B" + (j + 1) + " " + DrinkList[i][j].getName() + ", ");
            } else {
                System.out.print("C" + (j + 1) + " " + DrinkList[i][j].getName() + ", ");
            }
        }
        System.out.print("\n");
    }
    for (int i = 0; i < SnackList.length; i++) {
        for (int j = 0; j < SnackList[i].length; j++) {
            if (SnackList[i][j] == null) {
                System.out.print("[This Snack is out of stock], ");
            } else if (i == 0) {
                System.out.print("D" + (j + 1) + " " + SnackList[i][j].getName() + ", ");
            } else if (i == 1) {
                System.out.print("E" + (j + 1) + " " + SnackList[i][j].getName() + ", ");
            } else {
                System.out.print("F" + (j + 1) + " " + SnackList[i][j].getName() + ", ");
            }
        }
        System.out.print("\n");
    }
}
```

The displayContents function showed above is the function that would display the current contents of the vending machine. Print is used instead of println so that the display shows each coordinate of the row and column correctly.

```java
public void vendaccount() {
    Account logged = login();
    if (selectedDrink != null) {
        if (logged != null && logged.getBalance() > selectedDrink.getPrice()) {
            logged.setBalance(logged.getBalance() - selectedDrink.getPrice());
            System.out.println("Transaction Successful");
            System.out.println("Current Balance:");
            System.out.println(logged.getBalance());
        } else if (logged != null && logged.getBalance() < selectedDrink.getPrice()) {
            System.out.println("Insufficient Balance");
        } else {
            System.out.println("Payment Canceled");
        }
    } else {
        if (logged != null && logged.getBalance() > selectedSnack.getPrice()) {
            logged.setBalance(logged.getBalance() - selectedSnack.getPrice());
            System.out.println("Transaction Successful");
            System.out.println("Current Balance:");
            System.out.println(logged.getBalance());
        } else if (logged != null && logged.getBalance() < selectedSnack.getPrice()) {
            System.out.println("Insufficient Balance");
        } else {
            System.out.println("Payment canceled");
        }
    }
}
```

Shown above is the vendaccount function. This function is the highlight of this vending machine. It will be run when a user decides to pay using their vending machine accounts. The function will go through the users' list and find the right user, checking their balance, and if it's sufficient it will complete the transaction.

```java
public void cash() {
    boolean flag = true;
    int totalmoney = 0;
    Scanner scan = new Scanner(System.in);
    Scanner scan1 = new Scanner(System.in);
    while (flag) {
        System.out.println("Insert Cash (Machine only accepts, $10, $20, $50)");
        int moneyinput = scan.nextInt();
        if (moneyinput == 10 || moneyinput == 20 || moneyinput == 50) {
            totalmoney += moneyinput;
        } else {
            System.out.println("We only accept $10, $20, or $50");
        }
        System.out.println("Add more money? [yes/no]");
        String choice = scan1.nextLine();
        if (choice.equals("no")) {
            flag = false;
        } else if (choice.equals("yes")) {
            flag = true;
        } else {
            System.out.println("Input Invalid");
            System.exit( status: 0);
        }
    }
    setRegister(totalmoney);
    if (selectedDrink != null) {
        if (getRegister() < selectedDrink.getPrice()) {
            System.out.println("Insufficient Money");
            int change = getRegister();
            System.out.println("Money returned: " + change);
            setRegister(0);
```

Here’s another option for payment which is cash. Cash is a function that provides an alternative for users who does not own and do not want to create a vending machine account. This vending machine only accepts $10, $20, $50 anything other than these is rejected. The user is free to add any amount as well and the change will be returned. If the number of money inserted isn’t enough to pay, the transaction will be canceled

```java
public void topUpBalance(){
    Account logged = login();
    if(logged!=null){
        System.out.println("Current Balance:");
        System.out.println(logged.getBalance());
        boolean flag = true;
        int totalmoney = 0;
        Scanner scan = new Scanner(System.in);
        Scanner scan1 = new Scanner(System.in);
        while(flag) {
            System.out.println("Insert Cash (Machine only accepts, $10, $20, $50)");
            int moneyinput = scan.nextInt();
            if (moneyinput == 10 || moneyinput == 20 || moneyinput == 50) {
                totalmoney += moneyinput;
            } else {
                System.out.println("We only accept $10, $20, or $50");
            }
            System.out.println("Add more money? [yes/no]");
            String choice = scan1.nextLine();
            if (choice.equals("no")) {
                flag = false;
            } else if (choice.equals("yes")) {
                flag = true;
            } else {
                System.out.println("Input Invalid");
                System.exit( status: 0);
            }
        }
```

For the user who owns a vending machine account, the vending machine will provide a top-up feature. The topUpBalance function would be called whenever a user chooses that option. Similar to the cash function, the function will ask the user to input money. If they are satisfied with the amount they inserted the money will be added to their balance in their respective accounts.

```
public ArrayList<Account> getUsersList() {
    return UsersList;
}

public void fileWrite() throws IOException {
    FileWriter csvWriter = new FileWriter( fileName: "database.csv");
    for (int i = 0; i < getUsersList().size(); i++) {
        csvWriter.append(getUsersList().get(i).getUsername()).append(",");
        csvWriter.append(getUsersList().get(i).getPassword()).append(",");
        csvWriter.append(getUsersList().get(i).getName()).append(",");
        csvWriter.append(Integer.toString(getUsersList().get(i).getBalance()));
        csvWriter.append("\n");
    }

    csvWriter.flush();
    csvWriter.close();
}
```

```
public void loadFile() throws IOException {
    List<List<String>> data = new ArrayList<>();
    try (BufferedReader br = new BufferedReader(new FileReader( fileName: "database.csv"))) {
        String line;
        while ((line = br.readLine()) != null) {
            String[] values = line.split( regex: ",");
            data.add(Arrays.asList(values));
        }
        for (int i = 0; i < data.size(); i++) {
            String username = data.get(i).get(0);
            String password = data.get(i).get(1);
            String name = data.get(i).get(2);
            int balance = Integer.parseInt(data.get(i).get(3));
            addAccount(new Account(username, password, name, balance));
        }
    }

}
```

The following images shows the loadFile and fileWrite functions. These functions are the main functions that handle each user's account data. The loadFile function will run whenever the vending machine system is booted, while the fileWrite function will run whenever a user has finished with a certain transaction. This way small changes to the accounts would always be updated and whenever the system shuts down, the loadFile would always back up the data of the UsersList.

II. Drink Class

```java
package VendingMachine;

public class Drink implements IsProduct {
    private String name;
    private int price;
    private String size;
    private String temperature;
    private int amount;

    public Drink(String name, int price, String size, String temperature){
        this.name = name;
        this.price = price;
        this.size = size;
        this.temperature = temperature;
        this.amount = 5;
    }

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

```java
@Override
public void setName(String name) { this.name = name; }

@Override
public int getPrice() { return price; }

@Override
public void setPrice(int price) { this.price = price; }

@Override
public String getSize() { return size; }

@Override
public void setSize(String size) { this.size = size; }

@Override
public int getAmount() { return amount; }

@Override
public void setAmount(int amount) { this.amount = amount; }
```

```java
public String getTemperature() { return temperature; }

public void setTemperature(String temperature) { this.temperature = temperature; }

@Override
public String toString(){
    return "DRINK [Name= " + name + ", Price= " + price + ", Size= " + size + ", Temperature= " + temperature
}
```

The three images above shows the Drink class. This class implements an interface which is the isProduct interface. For an object to be classified as a drink it needs to have the following parameters: name, size, price, temperature, and amount. Each of the parameters has its own setter and getter methods. The toString function was overridden in this class to create a toString of their own which will then be a display for the drink's information.

III. Snack Class

```
package VendingMachine;

public class Snack implements IsProduct {
    private String name;
    private int price;
    private String size;
    private String flavor;
    private int amount;

    public Snack(String name, int price, String size, String flavor){
        this.name = name;
        this.price = price;
        this.size = size;
        this.flavor = flavor;
        this.amount = 5;
    }

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

```
@Override
public void setName(String name) { this.name = name; }

@Override
public int getPrice() { return price; }

@Override
public void setPrice(int price) { this.price = price; }

@Override
public String getSize() { return size; }

@Override
public void setSize(String size) { this.size = size; }

@Override
public int getAmount() { return amount; }

@Override
public void setAmount(int amount) { this.amount = amount; }
```

```
public String getFlavor() { return flavor; }

public void setFlavor(String flavor) { this.flavor = flavor; }

@Override
public String toString() {
    return "SNACK [Name= " + name + ", Price= " + price + ", Size= " + size + ", Flavor= " + flavor
}
```

The Snack class is not so different from the Drink class. The only difference is that the snack doesn't have a temperature parameter instead it has a flavor parameter.

IV. Subclasses

```
package VendingMachine;

public class Candy extends Snack {
    private String candytype;
    public Candy(String name, int price, String size, String flavor, String candytype) {
        super(name, price, size, flavor);
        this.candytype = candytype;
    }

    public String getCandytype() { return candytype; }

    public void setCandytype(String candytype) { this.candytype = candytype; }

    @Override
    public String toString() {
        return "CANDY [" + super.toString() + " candy type = " + candytype + " ]";
    }
}
```

```
package VendingMachine;

public class Chocolate extends Snack{
    private int cacaopercentage;
    public Chocolate(String name, int price, String size, String flavor, int cacaopercentage) {
        super(name, price, size, flavor);
        this.cacaopercentage = cacaopercentage;
    }

    public int getCacaopercentage() { return cacaopercentage; }

    public void setCacaopercentage(int cacaopercentage) { this.cacaopercentage = cacaopercentage; }

    @Override
    public String toString() {
        return "CHOCOLATE [" + super.toString() + " cacao percentage = " + cacaopercentage + " ]";
    }
}
```

```java
package VendingMachine;

public class SoftDrink extends Drink {
    private String flavor;
    public SoftDrink(String name, int price, String size, String temperature, String flavor) {
        super(name, price, size, temperature);
        this.flavor = flavor;
    }

    public String getFlavor() { return flavor; }

    public void setFlavor(String flavor) { this.flavor = flavor; }

    @Override
    public String toString(){ return "SOFT DRINK [" + super.toString() + " Flavor= " + flavor + " ]";
    }
}
```

The images above shows the three subclasses that exist in the system. There are Candy and Chocolate classes which are a subclass to the Snack class and SoftDrink class which is a subclass to the Drink class. Since they are subclasses they inherit all the methods from their superclass hence they only have a few new variables. For the initializer and toString of this class, we called the superclass initializer and toString but also added their own variables into it.

V. Account

```java
package VendingMachine;

public class Account {
    private String username;
    private String password;
    private String name;
    private int balance;

    public Account(String username, String password, String name){
        this.username = username;
        this.password = password;
        this.name = name;
        this.balance = 0;
    }
    public Account(String username, String password, String name, int balance){
        this.username = username;
        this.password = password;
        this.name = name;
        this.balance = balance;
    }
    public String getUsername() { return username; }

    public void setUsername(String username) { this.username = username; }

    public String getPassword() { return password; }

    public void setPassword(String password) { this.password = password; }
```

```
    public void setPassword(String password) { this.password = password; }

    public void setBalance(int balance) { this.balance = balance; }

    public int getBalance() { return balance; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }
}
```

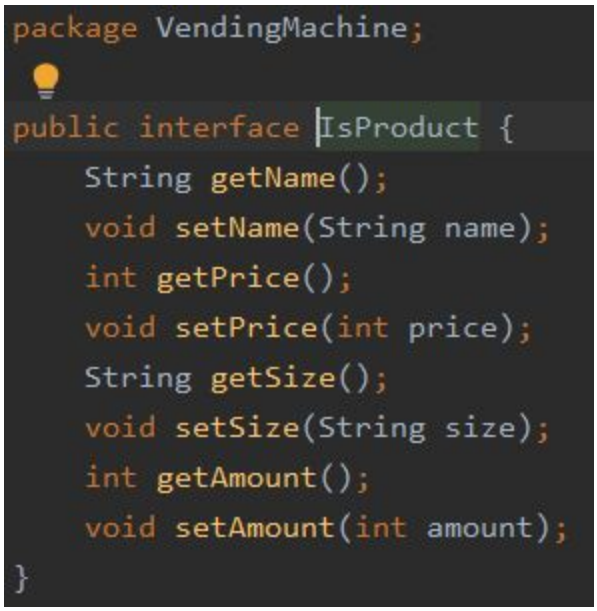
The account class is used create accounts for our users. It has its username, password, name, and balance. To make the account the users only need to input their username, password, and name while the balance will be automatically set to 0. Setter and getter methods are created to help to acquire and update the user's data

VI. Driver Class

```
boolean flag = true; //flag for while loop
v.loadFile(); //loads the user accounts
while (flag) {
    v.setSelectedSnack(null);
    v.setSelectedDrink(null);
    v.displayContents(); //display the vending machine contents
    System.out.println("Select what you want to do");
    System.out.println("1. Purchase");
    System.out.println("2. Create a Vend Account");
    System.out.println("3. Top up your Vend Account");
    System.out.println("4. Get product information");
    System.out.println("5. Admin Login");
    int choice = scan.nextInt();
    switch (choice) { //switch case for user choice input
        case 1:
            System.out.println("Select a product you want to purchase");
            while(v.getSelectedDrink() == null && v.getSelectedSnack() == null) {
                String id = scan1.nextLine();
                v.setSelectedItem(id);
            }
            Scanner scan2 = new Scanner(System.in);
            System.out.println("1. Confirm Selection");
            System.out.println("2. Cancel");
            choice = scan2.nextInt();
            if (choice == 1) {
                v.payment();
            } else {
                break;
            }
            break;
```

The driver class is where all the objects of the drinks and snacks are created and added to the vending machine before the users are able to use it. It contains all of the options that the users' can choose.

VII. Interface

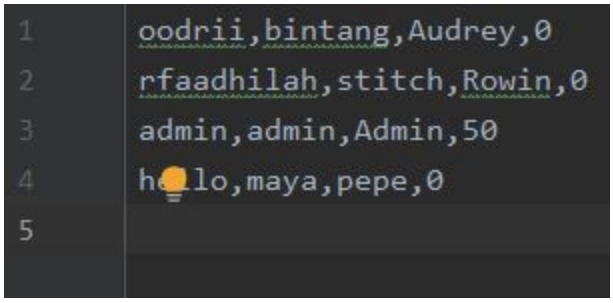


```
package VendingMachine;

public interface IsProduct {
    String getName();
    void setName(String name);
    int getPrice();
    void setPrice(int price);
    String getSize();
    void setSize(String size);
    int getAmount();
    void setAmount(int amount);
}
```

This is an image of the interface that is used in this program. Its the skeleton of the methods that are in Snack and Drink classes.

VIII. CSV File



```
1  oodrii,bintang,Audrey,0
2  rfaadhilah,stitch,Rowin,0
3  admin,admin,Admin,50
4  hello,maya,pepe,0
5
```

The initial CSV file looks like the image above. This where all the user data would be saved.

Conclusion

The program is working successfully, although there are many improvements that can be made. Judging by the requirements of the project, this vending machine has successfully achieved its goal. It is able to do what any other vending machine can do but also has new features being implemented. Codes that were learned from this semester and new codes from other sources are used to complete this vending machine. The program is working as what it is intended to do and errors were not encountered during the runtime of the program.

Evaluation

The program is not fully complete. One of the improvements that I could have done was to create a GUI for the program, therefore it looks more interactive. I feel that the codes are also a bit messy but hopefully, the code commenting would help to understand the code a bit better. In the future, there might be more subclasses to add, and more products than just snacks and drinks for example gacha vending machines. Also, I would like to implement a CSV for the drink and snacks in the future, it was hard to implement due to the subclasses.