

SAN JOSE STATE UNIVERSITY



SPRING 2017



CMPE 202 - Team Hackathon

Submitted by:

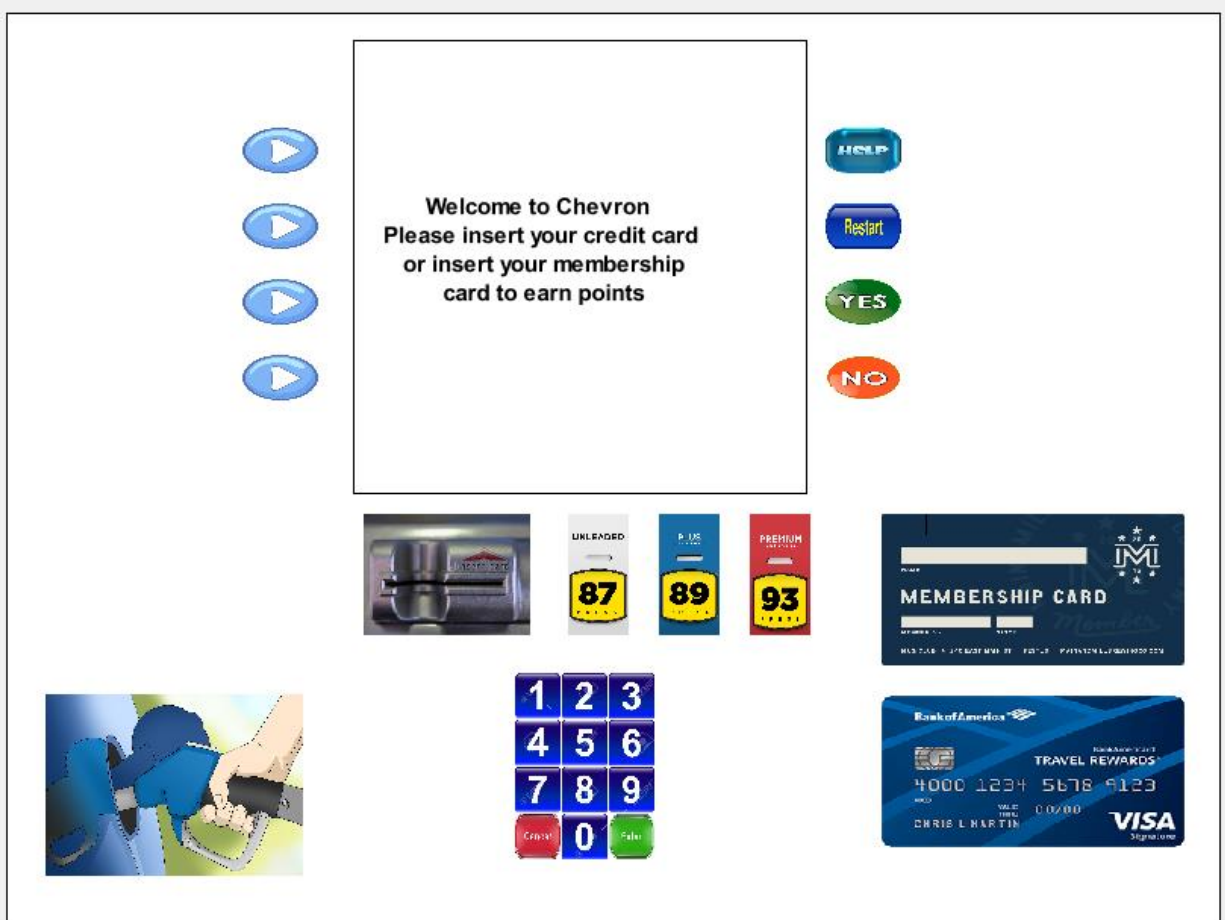
Team-14 Teen Titans

The Gas Pump Machine Processes Orders as follows:

A gas pump machine contains the following features

- A numeric keypad
- A credit card reader (with receipt printer) and
- A display screen with two rows of four buttons (as shown below).
- Options displayed on the screen depends on which step the machine is currently on while waiting for customer actions.

1. Customer must first insert a credit card and is prompted with Welcome message. If the customer scans the membership card, the customer is benefit with rewards point and a display message is prompted on the screen.



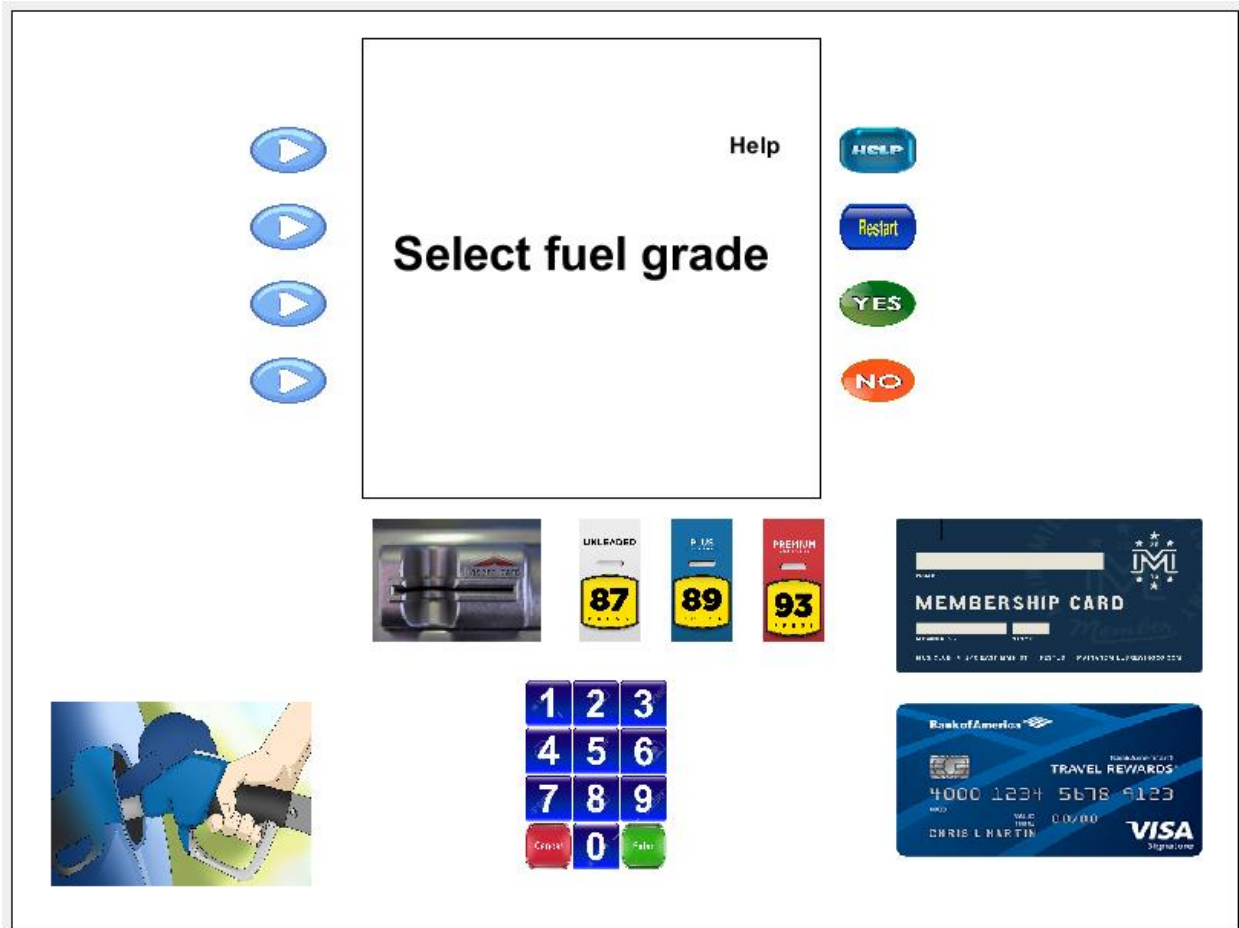
2. If a credit card is scanned, the screen displays a prompt for the customer to validate with a Zip Code. If not validated, selecting the fuel type should not be allowed.



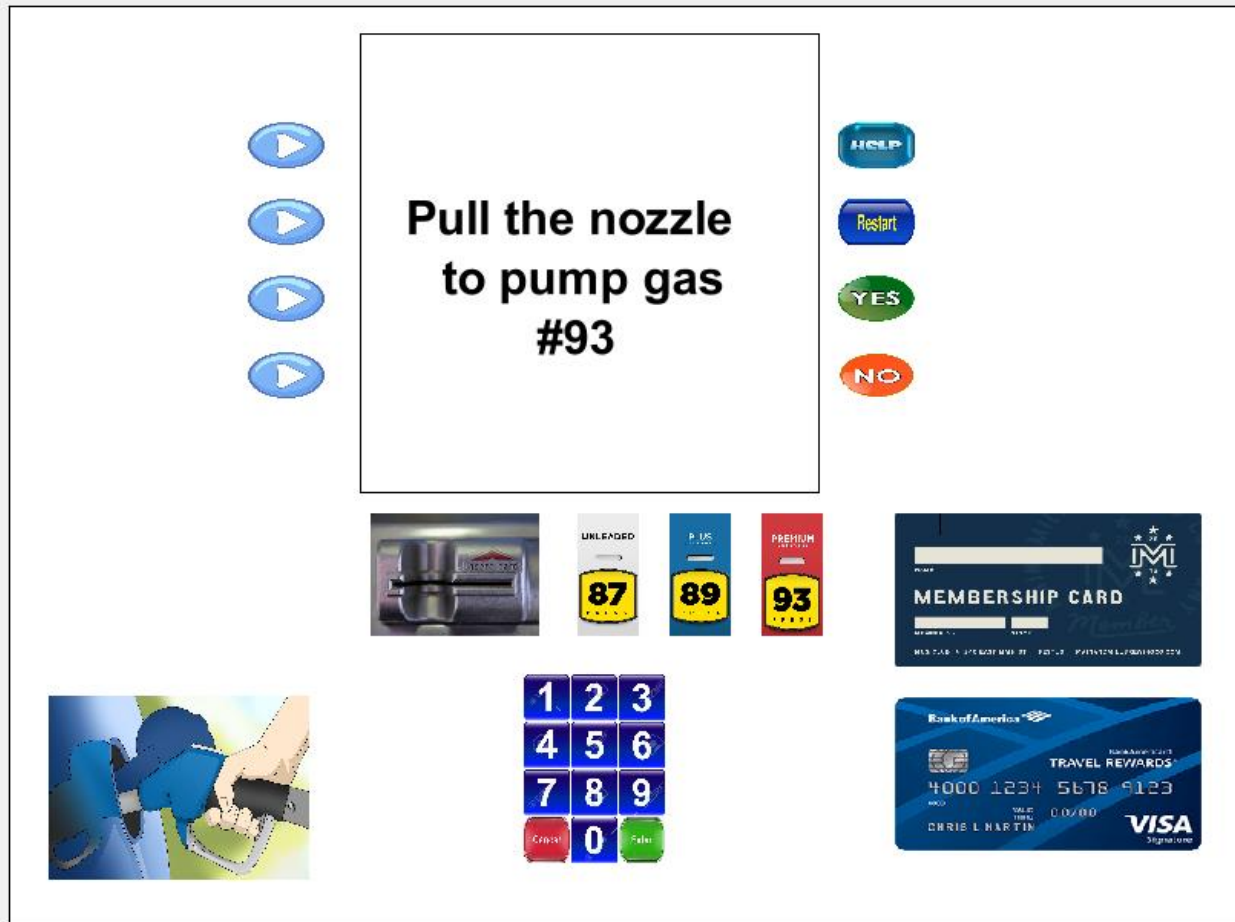
3. Customer has entered a valid zip code to get the gas type. A customer is prompted with a display message to wash the car or not.



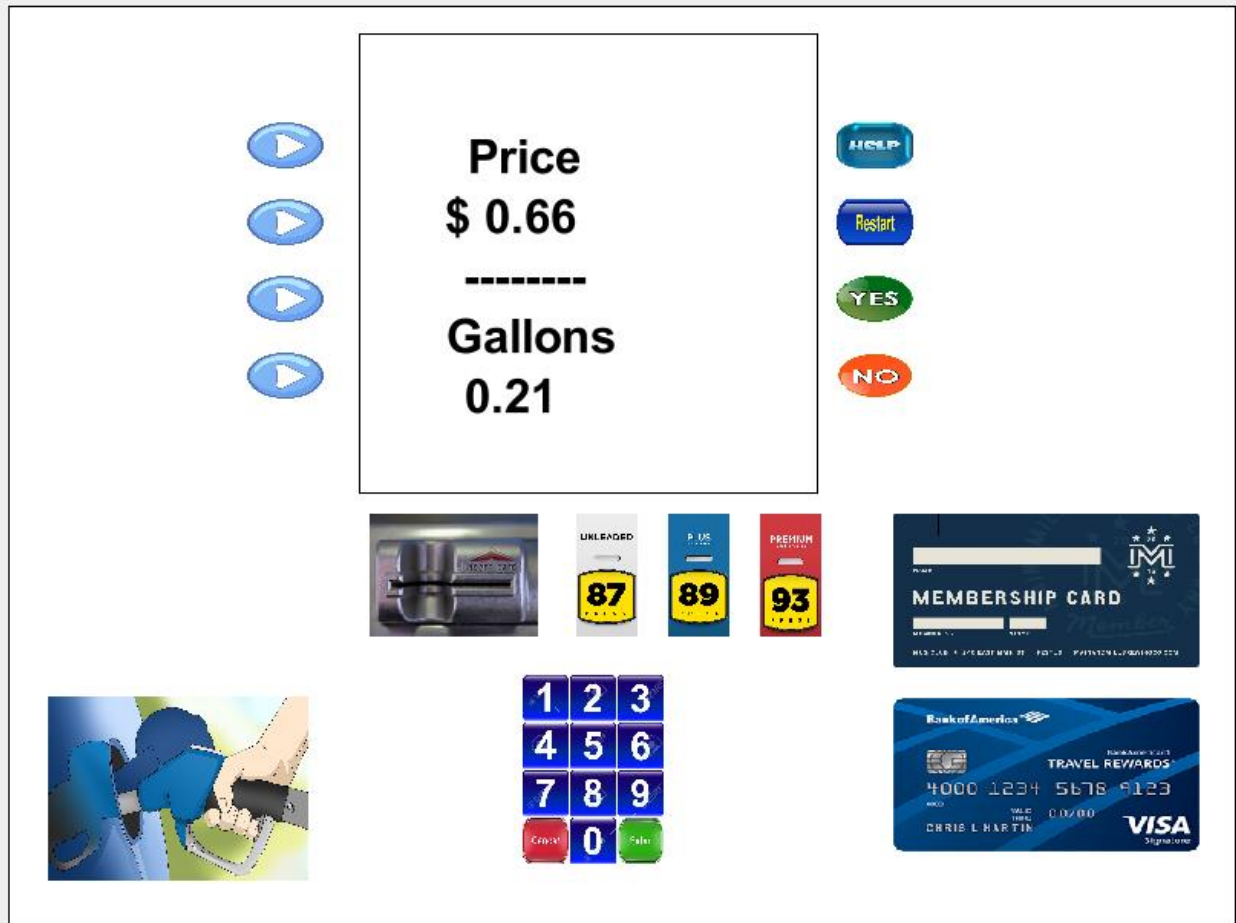
4. If the credit card has been validated, the customer is presented with options to proceed, the customer would select the Gas type and remove the pump nozzle to dispense gas.



5. Customer has selected gas type 93 which is premium.




6. When pumping is completed, the customer inserts the nozzle back into the holder. The display is prompted with the total price and gallons of the gas. The display now shows a message asking if the customer wants to print a receipt. Two options are shown, "Yes" or "No".



8. If the customer chooses “Yes”, a receipt is printed and machine prints out a message “Thank You” after printing the receipt.

Gas Receipt

×



Chevron

SALE RECEIPT

ADDRESS 1: San Jose 95112

DATE:4/9/2017

INVOICE# 10101010

AUTH#

MASTER CARD

ACCOUNT NUMBER

*****19

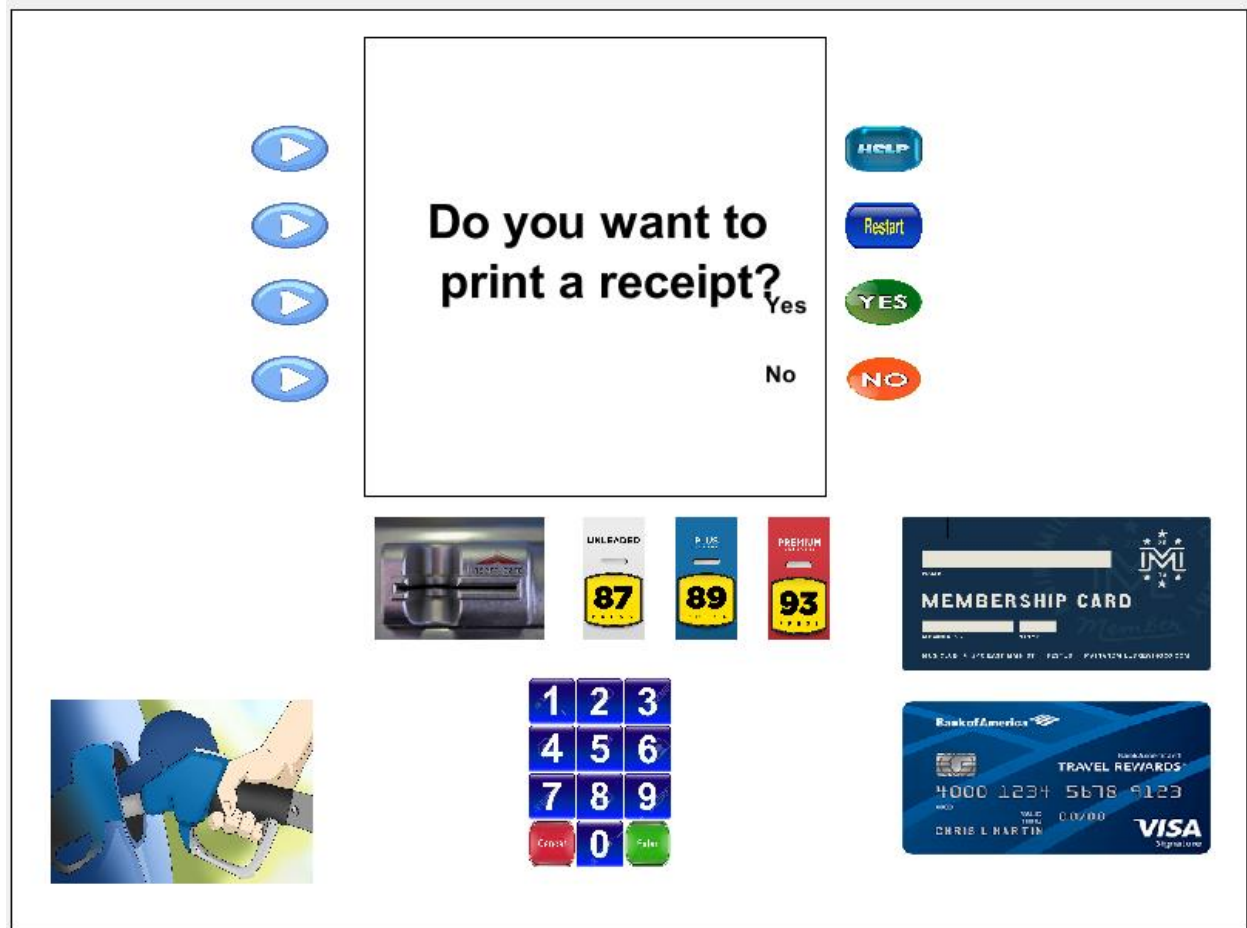
PUMP	GALLONS	CAR_WASH	DISCOUNTED	TOTAL
#93	0.21	\$5.00		\$5.66

Thank You!

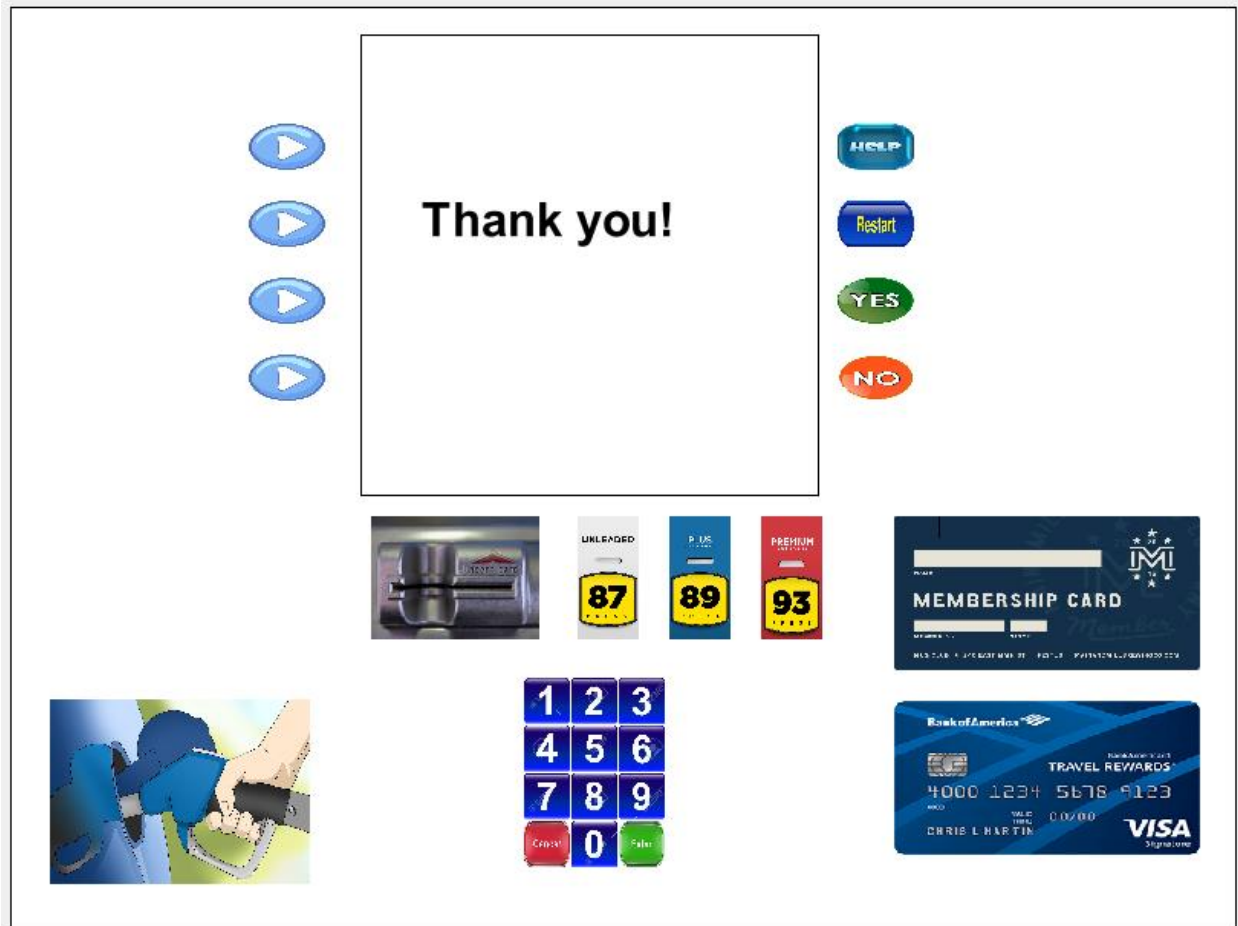
For Visiting Chevron!!

OK

9. If the customer does not want car wash option, click on NO and later display message for the receipt to be printed or not.

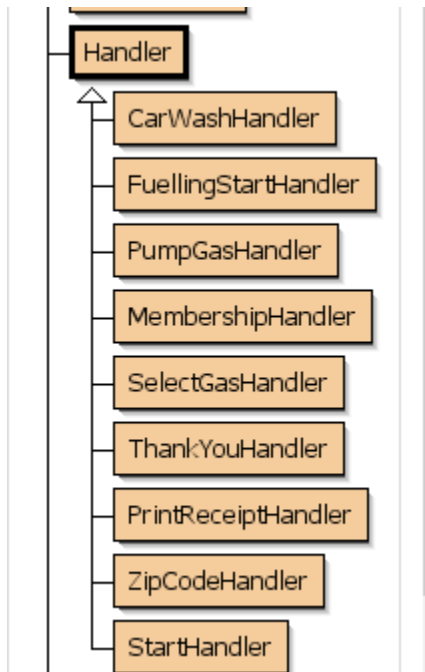


10. If customer don't want the receipt to be printed they can click on NO and a display message uis prompted with a THANK YOU.



Implementation of the design pattern:

We have used the chain of responsibility in our Handler class.



Handler - CMPE202-Team-Hackathon-master

Class Edit Tools Options

Handler X CarWashHandler X

Compile Undo Cut Copy Paste Find... Close

```
import greenfoot.*;
/**
 * Write a description of class Handler here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public abstract class Handler extends Actor
{
    Handler next;
    // instance variables - replace the example below with your own
    public abstract void handle(GasPumpMachine gasPumpMachine, String state,String buttonValue);
    public void setNext(Handler handler) {
        next = handler;
    }
}
```

1. Strategy Design pattern:
 - a. Strategy pattern in Receipt Class

Receipt - CMPE202-Team-Hackathon-master

Class Edit Tools Options

Receipt X

Compile Undo Cut Copy Paste Find... Close

```
import java.util.concurrent.TimeUnit;

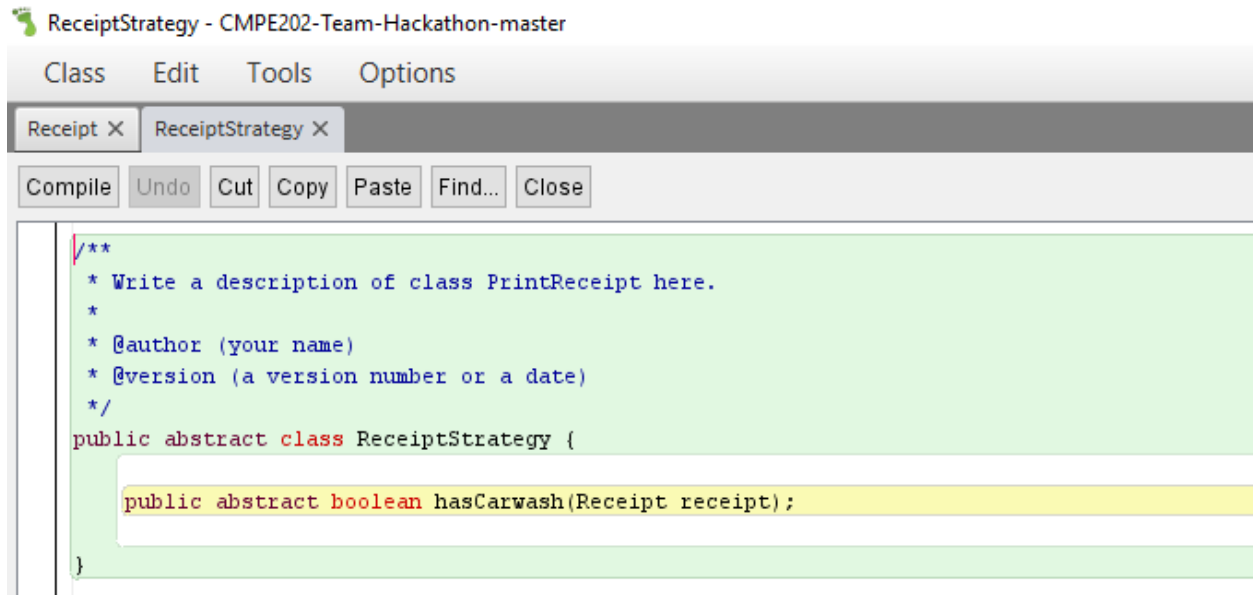
/**
 * Write a description of class Display here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Receipt {
    private double bill;
    private double price;
    private String message;
    private ReceiptStrategy receiptStrategy;
    private boolean hasCarWash;
    private StringBuilder st;

    //public Receipt(double bill, double price, String message) {
    public Receipt(double bill, double price, String message, ReceiptStrategy receiptStrategy) {
        this.bill = bill;
        this.price = price;
        this.message = message;
        this.receiptStrategy = receiptStrategy;
    }

    /*public boolean hasCarWash() {
        return printReceipt.hasCarWash(this);
    }*/

    public void displayReceipt() {
        st = new StringBuilder();
        Calendar calendar = Calendar.getInstance();
        int date2 = calendar.get(Calendar.DATE);
        int month2 = calendar.get(Calendar.MONTH);
        int year2 = calendar.get(Calendar.YEAR);
        String gallon2 = String.format("%.2f", bill/price);
        st.append("GAS STATION\nBRAND \nSALE RECEIPT \n\nADDRESS 1 \nADDRESS 2");
    }
}
```

b. Receipt strategy :



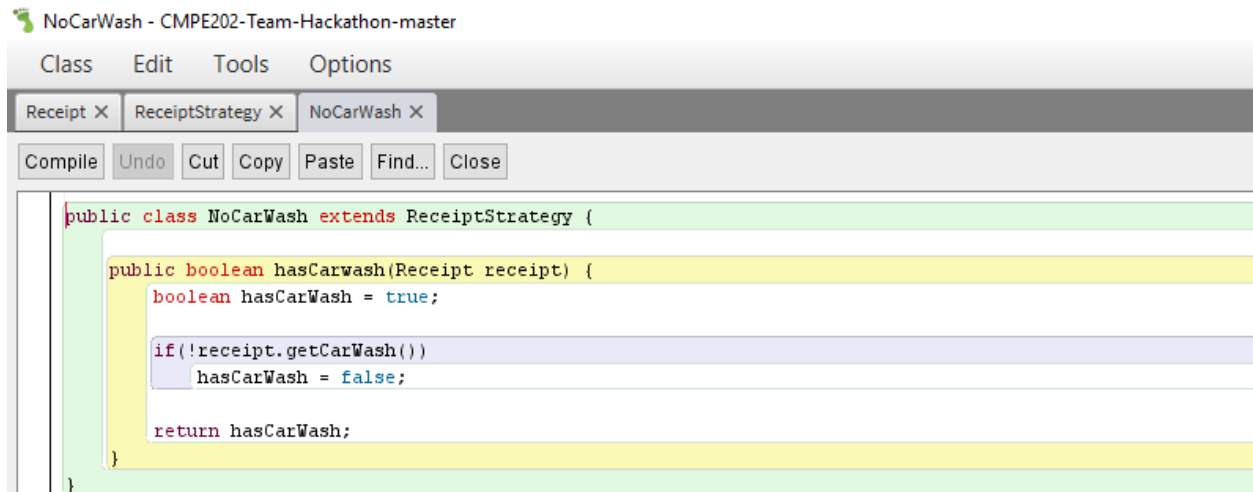
The screenshot shows an IDE window titled "ReceiptStrategy - CMPE202-Team-Hackathon-master". The menu bar includes "Class", "Edit", "Tools", and "Options". The tab bar shows "Receipt X" and "ReceiptStrategy X". The toolbar contains "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The code editor displays the following Java code:

```
/**
 * Write a description of class PrintReceipt here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public abstract class ReceiptStrategy {

    public abstract boolean hasCarwash(Receipt receipt);

}
```

c. We have also implemented Strategy design pattern for No car wash as follows :



The screenshot shows an IDE window titled "NoCarWash - CMPE202-Team-Hackathon-master". The menu bar includes "Class", "Edit", "Tools", and "Options". The tab bar shows "Receipt X", "ReceiptStrategy X", and "NoCarWash X". The toolbar contains "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The code editor displays the following Java code:

```
public class NoCarWash extends ReceiptStrategy {

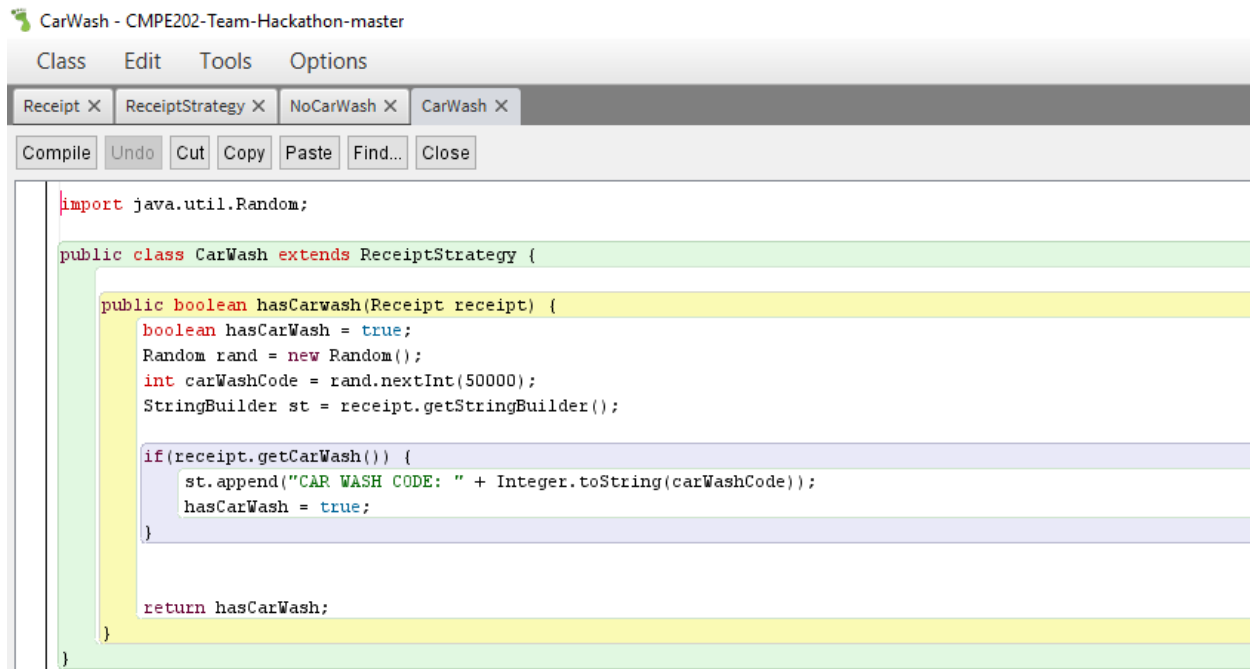
    public boolean hasCarwash(Receipt receipt) {
        boolean hasCarWash = true;

        if(!receipt.getCarWash())
            hasCarWash = false;

        return hasCarWash;
    }

}
```

d. We have also implemented Strategy design pattern for car wash as follows :



The screenshot shows an IDE window titled "CarWash - CMPE202-Team-Hackathon-master". The interface includes a menu bar with "Class", "Edit", "Tools", and "Options". Below the menu bar is a tab bar with four tabs: "Receipt X", "ReceiptStrategy X", "NoCarWash X", and "CarWash X". The "CarWash X" tab is selected. Below the tab bar is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The main editor area displays the following Java code:

```
import java.util.Random;

public class CarWash extends ReceiptStrategy {

    public boolean hasCarwash(Receipt receipt) {
        boolean hasCarWash = true;
        Random rand = new Random();
        int carWashCode = rand.nextInt(50000);
        StringBuilder st = receipt.getStringBuilder();

        if(receipt.getCarWash()) {
            st.append("CAR WASH CODE: " + Integer.toString(carWashCode));
            hasCarWash = true;
        }

        return hasCarWash;
    }
}
```

UML Class Diagram for gas pump machine:

