****

**Term\_Project 상세설계서**

|  |  |
| --- | --- |
| 과 목 | 자바프로그래밍 |
| 담당교수 | **김석훈** |
| 학 과 | **컴퓨터소프트웨어공학과** |
| 학 번 | **20204052** |
| 이 름 | **이현우** |
| 제 출 일 | **23.06.09** |

**목차**

**1. 클래스 설계**

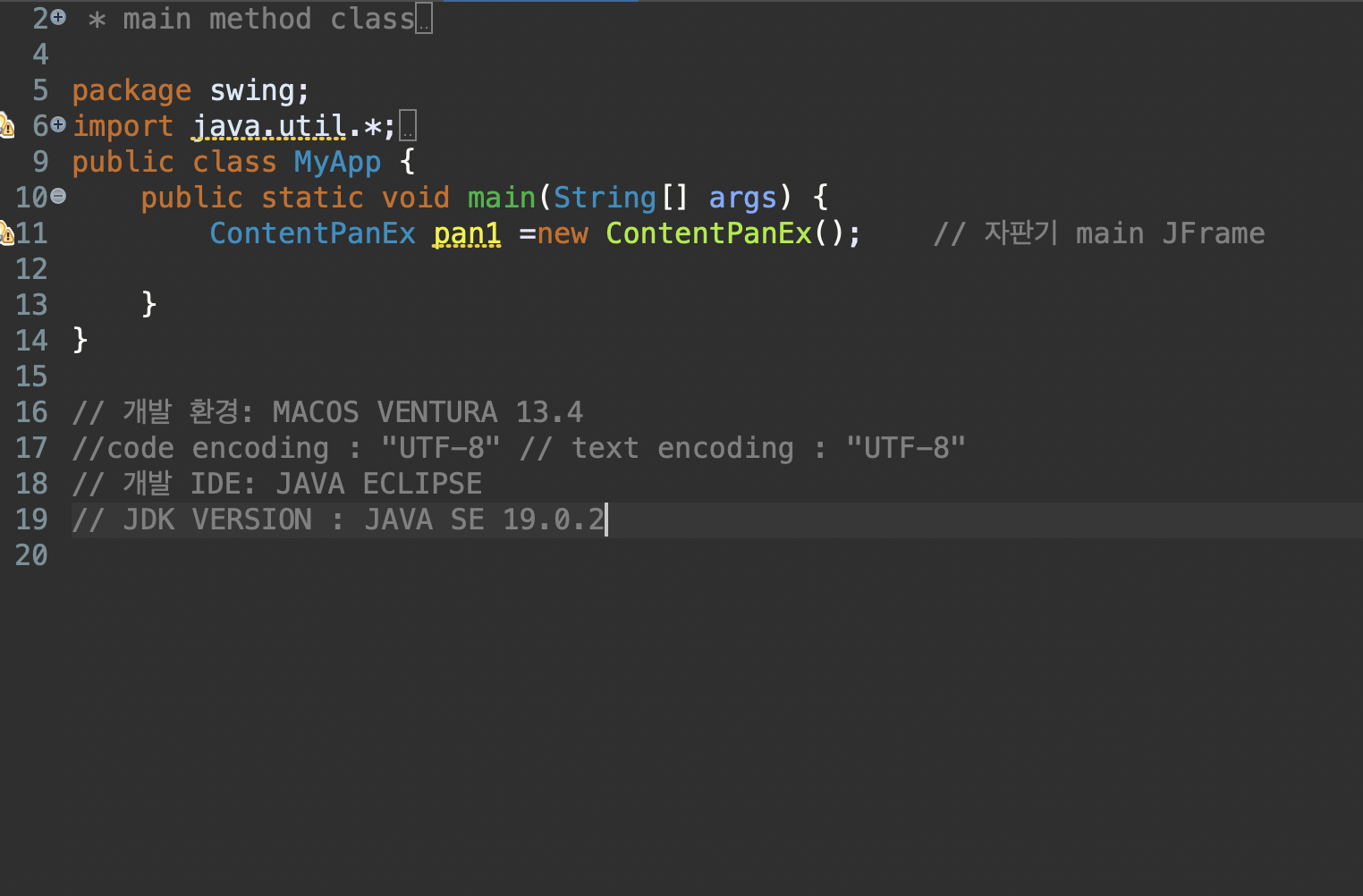
**2. 코드 설명**

1. **클래스 설계**

|  |  |
| --- | --- |
| 클래스명 | 클래스 개요 |
| myApp | Main method 가 있는 클래스 |
| ContentPanEx | MAIN GUI Frame 및 여러 instance를 정의한 class |
| CoinPanel | 자판기의 화폐 투입, 반환, 관리자 모드 Button 을 구현한 panel class |
| Productpanel | 상품의 이미지label, 상품 이름label, 상품 재고, 구매 button 을 구현한 panel class |
| MyFrame | 관리자 모드 Frame 및 여러 기능 frame class로 이동하는 Button을 구현한 class (change\_passwd 기능은 여기에 구현함) |
| CollectMoneyFrame | 화폐 수금 및 각 화폐의 추가 기능를 구현한 Frame class |
| SalesFrame | 일/월 매출을 Jtable로 구현한 Frame class |
| DrinkPanel | 음료 재고 추가, 반출, 음료 이름, 음료 가격을 변동할 수 있는 Frame class |
| FileIo | Txt file을 read, write method를 구현한 class |
| StackClass | 화폐투입 or 반출할때 사용하는 stack method 를 구현한 class |

1. **Class, Code 설명**

**2.1 Myapp**



main method가 있는 class이다. main method에서 자판기의 메인 GUI인 ContentPanEx class를 호출했다.

**2.2 ContentPanEx**

*/\**

*\* vendingmachine 의 main Frame*

*\*/*

**package** **swing**;

**import** **java.awt.Color**;

**import** **java.awt.Container**;

**import** **java.awt.Graphics**;

**import** **java.io.BufferedReader**;

**import** **java.io.FileNotFoundException**;

**import** **java.io.FileReader**;

**import** **java.io.IOException**;

**import** **java.util.concurrent.atomic.AtomicInteger**;

**import** **javax.swing.ImageIcon**;

**import** **javax.swing.JFrame**;

**import** **javax.swing.JLabel**;

**import** **javax.swing.JOptionPane**;

**import** **javax.swing.JPanel**;

**public** **class** **ContentPanEx** **extends** JFrame {

**static** **final** int MAX\_STACK = 50;

JLabel statusLa; *// 이벤트 상태를 출력해주는 label*

String[] drinkNames; *// 음료 이름*

AtomicInteger[] at\_pd\_count; *// 음료 재고 갯수*

AtomicInteger money; *// 넣어진 돈*

AtomicInteger Atprice[] ; *// 음료 가격*

AtomicInteger[][] stack\_money ;*// 기계안에 있는 돈 스택*

AtomicInteger[] stack\_top ;*// 돈 스택*

AtomicInteger charge\_1000\_count;*// 1000원을 넣은 갯수 count*

AtomicInteger machine\_totalmoney; *// 기계 안에 있는 총 돈*

ImageIcon imageIcon[];

JLabel money\_status;

**public** ContentPanEx () {

Container contentPane = getContentPane();

ImageIcon icon = **new** ImageIcon("src/image/vending\_machine.png"); *// 그림 출처: 20 김창환*

*//배경 Panel 생성후 컨텐츠페인으로 지정*

JPanel background = **new** JPanel() {

**public** void paintComponent(Graphics g) {

g.drawImage(icon.getImage(), 0, 0, **null**);

setOpaque(**false**); *//그림을 표시하게 설정,투명하게 조절*

**super**.paintComponent(g);

}

};

*// vendingmachine background 생성 method*

contentPane.setLayout(**null**);*//레이아웃을 내맘대로 설정가능하게 해줌.*

setTitle("Vending Machine");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); *// x버튼을 누르면 전원이 꺼짐*

statusLa = **new** JLabel("default",JLabel.CENTER); *// 이벤트 상태를 출력해주는 label*

setResizable(**false**);

setLocationRelativeTo(**null**);*//창이 가운데 나오게*

FileIo fileio = **new** FileIo();

**final** int Coin[] = {10,50,100,500,1000};

drinkNames = **new** String[5]; *// 음료 이름*

at\_pd\_count = **new** AtomicInteger[5]; *// 음료의 재고*

money = **new** AtomicInteger(0); *// 넣어진 돈*

Atprice = **new** AtomicInteger[5]; *// 음료 가격*

stack\_money = **new** AtomicInteger[5][MAX\_STACK+1];*// 기계안에 있는 돈 스택*

stack\_top = **new** AtomicInteger[5];*// 돈 스택*

charge\_1000\_count= **new** AtomicInteger(0);*// 1000원을 넣은 갯수 count*

machine\_totalmoney = **new** AtomicInteger(0); *//기계안에 있는 총 돈*

**for**(int i=0;i<5;i++) {

stack\_top[i] = **new** AtomicInteger(-1); *// 돈의 stack top을 -1 초기화*

}

**for** (int i = 0; i < 5; i++) {

**for** (int j = 0; j < MAX\_STACK+1; j++) {

stack\_money[i][j] = **new** AtomicInteger();

}

}

*// 돈의 스택 초기화*

fileio.DrinkRead(drinkNames, at\_pd\_count, Atprice);

fileio.CoinRead(stack\_top, stack\_money, Coin, machine\_totalmoney);

*// 음료 이름과, 음료 갯수, 음료 가격을 drink.txt에서 읽어와 저*

*// 돈의 현황을 coin.txt에서 읽어와 저장*

money\_status = **new** JLabel(String.valueOf(money.get()),JLabel.CENTER);

*// 돈의 투입 금액을 출력해주는 label*

imageIcon = **new** ImageIcon[5];

imageIcon[0] = **new** ImageIcon("src/image/water.jpg");

imageIcon[1] = **new** ImageIcon("src/image/coffee.jpg");

imageIcon[2] = **new** ImageIcon("src/image/energy\_drink.jpeg");

imageIcon[3] = **new** ImageIcon("src/image/high\_level\_coffee.jpg");

imageIcon[4]= **new** ImageIcon("src/image/cider.jpeg");

*// 음료 사진을 ImageIcon으로 생성*

ProductPanel propanel[] = **new** ProductPanel[5]; *// 상품 class 객체 배열로 생성 (상품이 5개 이므로)*

**for** (int i = 0; i < 5; i++) {

propanel[i] = **new** ProductPanel(drinkNames[i], imageIcon[i], Atprice[i], at\_pd\_count[i],statusLa,

money\_status, money,stack\_money,stack\_top,machine\_totalmoney, Coin

,charge\_1000\_count);

contentPane.add(propanel[i]);

propanel[i].setBounds(i\*100,20,80,300);

} *//productPanel을 초기화 하여 contentPane Frame 에 add함*

CoinPanel coinPanel = **new** CoinPanel(money, statusLa, money\_status, stack\_money,

stack\_top, machine\_totalmoney,charge\_1000\_count, drinkNames, at\_pd\_count, Atprice,

propanel); *// coinpanel 객체 생성*

contentPane.add(coinPanel); *// coinpanel을 현재 프레임에 add*

statusLa.setOpaque(**true**);

statusLa.setBounds(0,580,410,78);

statusLa.setBackground(Color.black);

statusLa.setForeground (Color.red);

statusLa.setFont(statusLa.getFont().deriveFont(13.0f));

contentPane.add(statusLa);

*// status label option 을 설정 및 contentpane에 add*

money\_status.setOpaque(**true**);

money\_status.setBackground(Color.black);

money\_status.setForeground (Color.red);

money\_status.setBounds(280,335,60,25);

contentPane.add(money\_status);

*// money status label option 설정 및 contentpane에 add*

setSize(480, 720);

setVisible(**true**);

*// Jframe 사이즈*

contentPane.add(background); *// background 이미지 add*

background.setBounds(0,0,480,720); *// background size 조절*

}

}

본인 코드에서 가장 main인 frame 코드이다. 여기서 여러 변수들과 자판기의 frame을 구현했다. 사진으로 보면 다음과 같은 구조이다. mainframe의 background image는 김창환 학우로 부터 양해를 구하고 이미지를 사용했다.

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

대부분의 변수를 AtomicInteger를 사용했는데 여러 프로세스의 동시성을 해결하기 위해 사용했다. contentpanEx에 선언된 변수들을 여러 class에서 사용할 것이기 때문에 AtomicInteger를 고안했다.

이 Frame class에 여러 panel class를 불러 Frame 위에 붙이는 작업을 했다.

contentPanEx에서는 크게 두가지 클래스를 호출하는데 Coinpanel과 ProductPanel이다.

CoinPanel은 위에 사진에 보이는 10,50,100,500,1000원, program종료, 반환 버튼을 구현했고, ProductPanel은 음료 사진, 이름, 가격, 구매버튼이 하나의 class이다. 따라서 contentpanEx class에서는 5개의 productPanel class를 호출했다.

**2.3 Coinpanel**

*/\**

*\* 지폐 투입 버튼과, 관리자 모드, 반환 버튼을 구현한 jpanel class*

*\*/*

**package** **swing**;

**import** **java.awt.Color**;

**import** **java.awt.event.ActionEvent**;

**import** **java.awt.event.ActionListener**;

**import** **java.io.BufferedReader**;

**import** **java.io.FileNotFoundException**;

**import** **java.io.FileReader**;

**import** **java.io.IOException**;

**import** **java.util.Stack**;

**import** **java.util.concurrent.atomic.AtomicInteger**;

**import** **javax.swing.\***;

**import** **javax.swing.JButton**;

**import** **javax.swing.JLabel**;

**import** **javax.swing.JOptionPane**;

**import** **javax.swing.JPanel**;

**class** **CoinPanel** **extends** JPanel {

**static** **final** int MAX\_STACK = 50 ;

CoinPanel(AtomicInteger money, JLabel status, JLabel money\_status, AtomicInteger stack\_money[][], AtomicInteger stack\_top[]

, AtomicInteger machine\_totalmoney,AtomicInteger charge\_1000\_count, String [] drinkNames, AtomicInteger[] at\_pd\_count,

AtomicInteger Atprice[], ProductPanel[] propanel){

StackClass stackClass = **new** StackClass();

String[] CoinNames = {"10원", "50원", "100원", "500원", "1000원"}; *//화폐 이름*

**final** int[] Coin= {10,50,100,500,1000};

JButton[] charge\_won = **new** JButton[5]; *// 10, 50, 100, 500, 1000원 버튼 레퍼런스*

int buttonWidth = 100; *// 버튼의 가로 크기*

int buttonHeight = 40; *// 버튼의 세로 크기*

int gap = 10; *// 버튼 사이의 간격*

**this**.setVisible(**true**);

**this**.setLayout(**null**);

**this**.setSize(480,720);

**this**.setBackground(**new** Color(255,0,0,0));

*// 버튼 초기 생성*

**for**(int i=0;i<5;i++) {

**final** int index = i; *// void actionPerforme에서 i를 사용하기위해 index 변수 선언*

charge\_won[i] = **new** JButton(CoinNames[i]); *// 지폐 투입 버튼*

charge\_won[i].addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

**if**(index==4) {

charge\_1000\_count.getAndAdd(1); *// 1000원을 입력 받으면 1000원 count*

}

**if** (stackClass.is\_full(stack\_top[index])==-1) {

status.setText(CoinNames[index]+"더이상 투입할 수 없습니다");

*//화폐 스택이 가득 차 있으면 staus label 출력*

}

**else** **if** (money.get() + Coin[index] <= 5000) { *// 투입 금액이 5000원 이하이고*

**if** (index==4 && charge\_1000\_count.get()>3) {

status.setText("1000원을 투입할 수 없습니다.");

charge\_1000\_count.getAndAdd(-1);

*//만약 1000원권 투입을 3개 초과 넣었을 시 1000원 투입불*

}

**else** {

money.addAndGet(Coin[index]);

*// 총투입금액 변수 money에 돈을 투입*

money\_status.setText(String.valueOf(money.get()));

*// money vaulue 값 money label에 출력*

stackClass.push(stack\_money[index], stack\_top[index], Coin[index], machine\_totalmoney);

*// money stack에 coin push*

**for**(int j=0;j<5;j++) {

propanel[j].button\_refresh(money,Atprice[j]);

*// 구매 버튼의 색깔을 바꾸는 refresh method 실행*

}

}

}

**else** {

status.setText("최대 금액을 초과합니다."); *// 금액이 5000원 초과이면 status label 출력*

}

}

});

charge\_won[i].setBounds(i\*75,525,70,50);

**this**.add(charge\_won[i]);

}

JButton return\_coin = **new** JButton("반환"); *// 반환 버*

*// 반환 조건은 1000원 부터 10원까지 최대한 큰 화폐로 반*

return\_coin.addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

int count=4;

int return\_count[]= {0,0,0,0,0}; *// 동전 갯수 반환 카운트 변*

charge\_1000\_count.set(0); *// 반환 할 시 1000원 count 0으로 초기화*

**while**(money.get()!=0) { *// 투입 금액이 0 일때까지*

*// 투입금액이 coin[count]보다 크고 스택이 비어있지 않을 때 까지*

**while**(money.get()>=Coin[count] && stack\_top[count].get()!=-1) {

money.addAndGet(-(stackClass.pop(stack\_money[count], stack\_top[count], machine\_totalmoney)));

return\_count[count]++; *// moneystack에서 pop 한만큼 money에서 sub*

}

count--;

**if**(count==-1) **break**;

}

**for**(int j=0;j<5;j++) {

propanel[j].button\_refresh(money,Atprice[j]); *// 구매 버튼 refresh method 실행*

}

money\_status.setText(Integer.toString(money.get())); *// money status set*

String tmp = **null**;

**for**(int i=0;i<5;i++) { *// 반환 화폐 갯수를을 status label에 set*

**if**(return\_count[i]!=0) {

**if** (tmp == **null**) {

tmp = CoinNames[i] + ":" + Integer.toString(return\_count[i]) + "개 ";

} **else** {

String tmp2 = CoinNames[i] + ":" + Integer.toString(return\_count[i]) + "개 ";

System.out.println(tmp2);

tmp += tmp2;

}

}

}

**if**(tmp!= **null**) status.setText(tmp + "반환 되었습니다. ");

**else** {

status.setText("반환할 돈이 없습니다.");

}

}

});

JButton manager\_bu = **new** JButton("관리자 모드"); *// 관리자 모드 버*

JButton exit\_bu = **new** JButton("Program Exit"); *// 자판기프로그램 종료 버튼*

return\_coin.setBounds(355,343,50,50);

manager\_bu.setBounds(20,348,229,172);

exit\_bu.setBounds(380,525,100,50);

**this**.add(return\_coin);

**this**.add(manager\_bu);

**this**.add(exit\_bu);

exit\_bu.addActionListener(**new** ActionListener() { *// 버튼을 누를 시 coin.txt drink.txt*

**public** void actionPerformed(ActionEvent e) { *// 저장 후 프로그램 종료*

FileIo fileio = **new** FileIo();

fileio.CoinWrite(stack\_top);

fileio.DrinkWrite(drinkNames, Atprice, at\_pd\_count);

System.exit(0);

}

});

manager\_bu.addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

managerbutton\_method(stack\_top, stack\_money, CoinNames, machine\_totalmoney, drinkNames, at\_pd\_count, Atprice);

}

});

}

*// 비밀번호 입력 메소드 passwd.txt에서 비밀번호를 가져와 입력받은 비밀번호와 대조 후 MYframe클래스(관리자모드) 불러오기*

**private** void managerbutton\_method(AtomicInteger stack\_top[], AtomicInteger stack\_money[][], String[] CoinNames, AtomicInteger machine\_totalmoney,

String [] drinkNames, AtomicInteger[] at\_pd\_count, AtomicInteger Atprice[]){

String passwd\_input = JOptionPane.showInputDialog(**null**, "비밀번호를 입력하세요.");

**try** {

BufferedReader reader = **new** BufferedReader(**new** FileReader("src/DB/passwd.txt"));

String strtmp = **null**;

strtmp = reader.readLine();

reader.close();

**if**(passwd\_input != **null** && passwd\_input.equals(strtmp)) {

**new** MyFrame(stack\_top, stack\_money, CoinNames, machine\_totalmoney, drinkNames, at\_pd\_count, Atprice);

}

**else** {

JOptionPane.showMessageDialog(**null**, "PASSWORD IS WRONG!");

}

} **catch** (FileNotFoundException e1) {

JOptionPane.showMessageDialog(**null**, "FILE DATA OPEN ERROR", "ERROR", JOptionPane.ERROR\_MESSAGE);

} **catch** (IOException e1) {

JOptionPane.showMessageDialog(**null**, "FILE READ ERROR", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

}

}

Coinpanel의 가장 큰 기능은 화폐 투입 및 반환이다. 지폐를 어떤방식으로 처리할까 하다가 생각한 방법이 stack을 사용한 방법이었다. 코인 버튼을 누를 시 지폐 스택 stack\_money에 push하는 방법으로 했다. 여기서 고민이 많이 됐던 것이 1000원의 지폐 투입 한도이다. 지폐는 3000원의 제한이 있지만 만약 1000원 3장을 투입하고 1000원 이상의 물품을 구입하면 그 상태에서 1000원을 더 넣을 수 있는지가 요구 기능에서는 명료하게 적혀있지 않아 난황이었다. 필자는 1000원을 다시 넣을 수 있는 기준을 물품을 다 사고 잔돈이 0원이거나 아니면 반환 버튼을 눌렀을 때만 1000원 count를 초기화 하게 설정했다.

반환 기능은 최대한 적은 coin을 주는 것이 맞다고 판단해 그리디 알고리즘으로 기능을구현했다. 1000원 stack부터 pop하고 1000원을 pop할 수 없을 때 그다음 높은 화폐를 pop하는 방식으로 적용했다.

관리자모드는 passwd를 입력하게 했으며 db폴더의 passwd.txt에서 대조하여 맞으면 관리자 모드 Frame인 MyFrame class를 생성했다. program exit에서는 모든 파일을 저장하고 프로그램을 종료하는 기능을 넣었다.

**2.4 ProductPanel**

*/\**

*\* 음료 이름, 음료 갯수, 음료 가격, 구매 버튼을 구현한 JPanel class*

*\*/*

**package** **swing**;

**import** **java.awt.Color**;

**import** **javax.swing.Timer**;

**import** **java.awt.event.ActionEvent**;

**import** **java.awt.event.ActionListener**;

**import** **java.util.LinkedList**;

**import** **java.util.concurrent.atomic.AtomicInteger**;

**import** **javax.swing.ImageIcon**;

**import** **javax.swing.JButton**;

**import** **javax.swing.JLabel**;

**import** **javax.swing.JPanel**;

**import** **javax.swing.event.ChangeEvent**;

**import** **javax.swing.event.ChangeListener**;

**class** **ProductPanel** **extends** JPanel {

**private** LinkedList<Integer> list; *// 음료 재고 linkedlist*

**private** StackClass stackClass; *// stack push pop class*

**private** JButton buyButton; *// 구매 버튼*

ProductPanel(String drinkName, ImageIcon imageIcon, AtomicInteger price, AtomicInteger at\_pd\_count,JLabel statusLa, JLabel money\_status,

AtomicInteger money, AtomicInteger stack\_money[][], AtomicInteger top[], AtomicInteger machine\_totalmoney, int Coin[]

,AtomicInteger charge\_1000\_count){

list = **new** LinkedList<Integer>(); *// 음료 재고를 linkedlist로 구현*

**for**(int i=0;i<at\_pd\_count.get();i++) { *// 음료 재고 갯수만큼 list.add*

list.add(1);

}

**this**.setBackground(**new** Color(255,0,0,0));

**this**.setLayout(**null**);

JLabel imageLabel = **new** JLabel(imageIcon); *// 음료 사진*

buyButton = **new** JButton(); *// 음료 구매 버튼*

buyButton.setOpaque(**true**);

JLabel drinkLabel = **new** JLabel(drinkName,JLabel.CENTER);

JLabel price\_Label = **new** JLabel(Integer.toString(price.get())+"원",JLabel.CENTER);

Timer timer = **new** Timer(100, **new** ActionListener() { *// 0.1초 단위로 button refresh()*

**public** void actionPerformed(ActionEvent e) {

button\_refresh(money, price);

}

});

int imageX = 10; *// 이미지의 가로 위치*

int imageY = 10; *// 이미지의 세로 위치*

int imageWidth = 80; *// 이미지의 폭*

int imageHeight = 200; *// 이미지의 높이*

imageLabel.setBounds(imageX, imageY, imageWidth, imageHeight);

int buttonX = 10; *// 버튼의 가로 위치*

int buttonY = 270; *// 버튼의 세로 위치*

int buttonWidth = 80; *// 버튼의 폭*

int buttonHeight = 30; *// 버튼의 높이*

buyButton.setBounds(buttonX, buttonY, buttonWidth, buttonHeight);

int drinkLabelX = 10; *// 음료 이름의 가로 위치*

int drinkLabelY = 180; *// 음료 이름의 세로 위치*

int drinkLabelWidth = 80; *// 음료 이름의 폭*

int drinkLabelHeight = 20; *// 음료 이름의 높이*

drinkLabel.setBounds(drinkLabelX, drinkLabelY, drinkLabelWidth, drinkLabelHeight);

int priceLabelX = 10; *// 가격의 가로 위치*

int priceLabelY = 200; *// 가격의 세로 위치*

int priceLabelWidth = 80; *// 가격의 폭*

int priceLabelHeight = 20; *// 가격의 높이*

price\_Label.setBounds(priceLabelX, priceLabelY, priceLabelWidth, priceLabelHeight);

add(imageLabel);

add(drinkLabel);

add(price\_Label);

add(buyButton);

stackClass = **new** StackClass();

timer.start();

buyButton.addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) { *// 물품을 구매하고 잔돈을 반환할 수 없을 때 확인 method*

**if**( stackClass.is\_return(stack\_money, top, machine\_totalmoney, Coin, money, price)==**false**) {

statusLa.setText("잔돈이 충분하지 않아 구매할 수 없습니다");

}

**else** { *// 잔돈을 반환할 수 있으면*

**if**(list.size()>0) {

**if**(money.get()>=price.get()) { *// 상품 재고가 있고 투입 금액이 상품 가격보다 많으면 구매*

list.remove(); *// 재고 하나 삭제*

at\_pd\_count.set(list.size()); *// 재고 count set*

statusLa.setText(drinkName + "이(가) 구매되었습니다."); *// 물품 구매 label set*

money.addAndGet(-(price.get())); *// 투입 금액 변*

money\_status.setText(String.valueOf(money.get())); *// 투입 금액 label set*

**new** FileIo().SalesWrite(drinkName,price.get()); *// 월별 매출 데이터 파일에 write*

**if**(money.get()==0) { *// 투입금액이 0원이면 1000원count 초기*

charge\_1000\_count.set(0);

}

}

**else** {

statusLa.setText("not enough money");

}

}

**else** {

statusLa.setText(drinkName + "이(가) 품절되었습니다.");

}

}

}

});

}

void button\_refresh(AtomicInteger money, AtomicInteger price ) *// 구매버튼 stauts refresh 메소드*

{

**if**(list.size()==0) {

buyButton.setText("sold out");

buyButton.setBackground(Color.RED);

}

**else** **if**(price.get()<=money.get()) {

buyButton.setText("Buy");

buyButton.setBackground(Color.GREEN);

}

**else** {

buyButton.setBackground(Color.black);

buyButton.setText("Default");

}

}

}

productpanel은 위에서 말했다싶이 상품 이름, 가격, 이미지 label과 구매 button, 재고를 관리하는 class이다.

음료 재고는 제너릭 linkded list를 이용했다. drink.txt 에서 읽은 재고 값을 at\_pd\_count로 받고 for문으로 at\_pd\_count 만큼 list에 add(1)을 했다. drinkname과 price도 마찬가지로 파일에서 읽어 온 것을 생성자로 받아서 Jlabel화 했다. 구매 button은 하단에 refresh method를 구현해서 timeclass로 0.1초마다 버튼의 status(색깔, text)을 refresh()하게 했다.

fefresh() method 구현은 크게 세가지 상태로 했는데 list.size==0(재고 없음) 일때, price(상품가격)이 투입 금액보다 작거나 같을 때, 기본 Default 설정 세가지로 구현했다.

Button Action 기능은 다음과 같이 구현했다. 먼저 stackclass를 사용하여 isreturn method로 물품을 구입하고 잔돈을 반환할 수 있는지에 대한 것을 확인한다. (is\_return method는 아래 stackclass code review 에서 설명하겠다.) 잔돈이 충분하지 않으면 잔돈을 반환할 수 없다는 status label을 set 한다. 잔돈이 충분하고 재고도 있으며 투입금액이 상품 금액에 상응하면 물품 구매로 이루어지고 list를 remove하고 atpdcout도 set한다. status label에는 “drinkname+ 이 구매되었습니다.” 를 set하고 money- 상품가격을 한후 money 값을 money status에 set한다. 여기서 만약 물품을 구매하고 투입금액이 0원이 되면 1000원 count 변수를 0으로 수정한다. 그리고 FileIo class의 Saleswrite method(매출 file write)로 구매 기록을 쓴다. 투입금액이 충분하지 않을 때와, 재고가 없을 때는 statusla에 “not enough money” 와 “drinkname 이 품절되었습니다.”를 set한다.

**2.5 MyFrame**

*/\**

*\* 관리자 모드 Frame class*

*\*/*

**package** **swing**;

**import** **java.awt.Color**;

**import** **java.awt.GridLayout**;

**import** **java.awt.event.ActionEvent**;

**import** **java.awt.event.ActionListener**;

**import** **java.io.FileOutputStream**;

**import** **java.io.OutputStream**;

**import** **java.util.concurrent.atomic.AtomicInteger**;

**import** **javax.swing.JButton**;

**import** **javax.swing.JFrame**;

**import** **javax.swing.JOptionPane**;

**class** **MyFrame** **extends** JFrame {

**static** **final** char special\_chars[] = {'!', '@', '#', '$', '%', '^', '&', '\*', '+', '-'};*//특수문자*

**static** **final** char numbers[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'}; *// 비밀번호 숫자*

**public** MyFrame(AtomicInteger[] stack\_top, AtomicInteger[][] stack\_money, String CoinNames[],

AtomicInteger machine\_totalmoney, String [] drinkNames, AtomicInteger[] at\_pd\_count,

AtomicInteger Atprice[]) {

setTitle("관리자 모드");

setSize(500, 500);

setVisible(**true**);

setResizable(**false**);

GridLayout grid = **new** GridLayout(3, 2);

setBackground(Color.pink);

setLayout(grid);

JButton ManagerBu[] = **new** JButton[6];

ManagerBu[0] = **new** JButton("음료 현황 및 변경");

ManagerBu[1] = **new** JButton("화폐현황 및 수금");

ManagerBu[2] = **new** JButton("NULL");

ManagerBu[3] = **new** JButton("관리자 비밀번호 변경");

ManagerBu[4] = **new** JButton("일별/월별 매출");

ManagerBu[5] = **new** JButton("종료");

**for** (int i = 0; i < 6; i++) {

add(ManagerBu[i]);

}

ManagerBu[0].addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

**new** DrinkPanel(drinkNames, at\_pd\_count,Atprice);

}

});

ManagerBu[1].addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

**new** CollectMoneyFrame(stack\_top, stack\_money, CoinNames, machine\_totalmoney);

}

});

ManagerBu[3].addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

change\_passwd\_method();

}

});

ManagerBu[4].addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

**new** SalesFrame();

}

});

ManagerBu[5].addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

dispose();

}

});

}

*// 비밀번호에 특수문자 포함 여부 검사*

**private** boolean hasSpecialChar(String password, char[] special\_chars) {

**for** (char special\_char : special\_chars) {

**if** (password.contains(String.valueOf(special\_char))) {

**return** **true**;

}

}

**return** **false**;

}

*// 비밀번호 숫자 포함 여부 검사*

**private** boolean hasNumber(String password, char[] numbers) {

**for** (char number : numbers) {

**if** (password.contains(String.valueOf(number))) {

**return** **true**;

}

}

**return** **false**;

}

*// 비밀번호 변경 method*

**private** void change\_passwd\_method()

{ *// 새 비밀번호 입력*

String change\_passwd = JOptionPane.showInputDialog(**null**,

"변경할 비밀번호를 입력하세요.(8자 이상, 숫자 하나 이상 또는 특수문자 하나 이상 포함)");

**if** (change\_passwd != **null**) {

*// 새 비밀번호 확인 받기*

String change\_passwd\_confirm = JOptionPane.showInputDialog(**null**,

"변경할 비밀번호를 다시 입력하세요.");

*// 비밀번호 변경 조건 확인*

**if** (change\_passwd.equals(change\_passwd\_confirm) && change\_passwd.length() >= 8 &&

(hasNumber(change\_passwd, numbers) || hasSpecialChar(change\_passwd, special\_chars))) {

**try** {

*// 새 비밀번호 저장*

OutputStream output = **new** FileOutputStream("src/DB/passwd.txt");

output.write(change\_passwd.getBytes());

output.close();

*// 변경 완료 메시지 표시*

JOptionPane.showMessageDialog(**null**, "비밀번호가 성공적으로 변경되었습니다.");

} **catch** (Exception e1) {

e1.printStackTrace();

}

} **else** {

*// 예외처리*

JOptionPane.showMessageDialog(**null**, "비밀번호가 조건을 만족하지 않습니다. 다시 입력해주세요.");

}

}

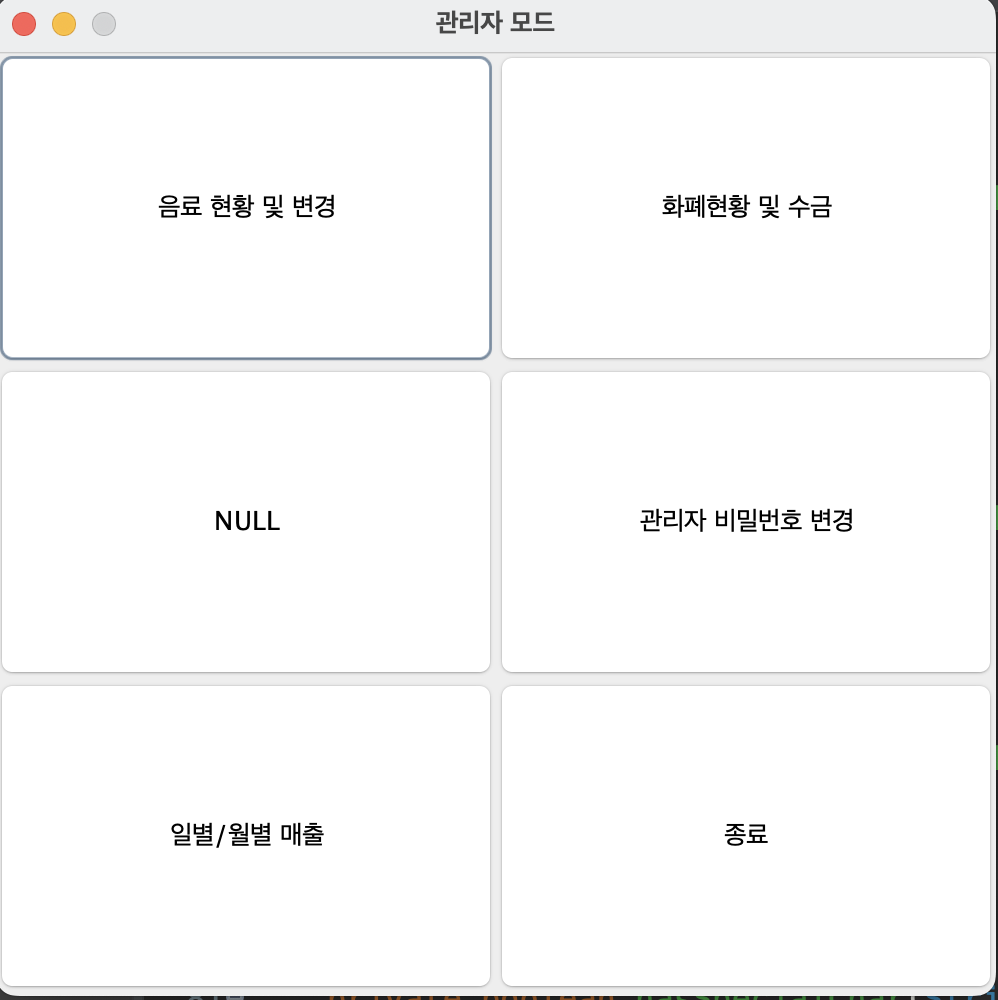
}

}

MyFrame은 관리자 모드 Jframe이다. Coinpanel class에 있는 관리자모드 button으로비밀번호가 맞을시 MyFrame이 생성된다. GUI는 다음과 같다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명



각 버튼들은 각자의 class or 기능method를 생성한다. (NULL 버튼은 grid 때문에 임의로 만들었다. 액션이 있는 버튼은 아니다.)

각 기능 class는 뒤에 설명하겠다. 여기서 관리자 비밀번호 변경은 현재 class에 method로 구현했다. 비밀번호 변경 버튼을 누르면 change\_passwd\_method를 실행한다.

JoptionPane으로 변경할 비밀번호를 입력 받는다. 그 후 변경할 비밀번호를 다시 입력받는다. 만약 변경할 비밀번호와 다시 입력한 비밀번호가 같고 비밀번호의 길이가 8자 이상이고 특수문자 포함 확인 method hasSpecialChar 와 숫자 포함 확인 method hasNumber 둘 중 하나라도 true이면 DB/passwd.txt 에 접근해서 새로운 비밀번호를 write한다.

**2.6 CollectMoneyFrame**

*/\**

*\* 관리자 모드 화폐 현황 및 수금 JFrame class*

*\*/*

**package** **swing**;

**import** **java.awt.Color**;

**import** **java.awt.GridLayout**;

**import** **java.awt.event.ActionEvent**;

**import** **java.awt.event.ActionListener**;

**import** **java.util.concurrent.atomic.AtomicInteger**;

**import** **javax.swing.JButton**;

**import** **javax.swing.JFrame**;

**import** **javax.swing.JLabel**;

**import** **javax.swing.JOptionPane**;

**class** **CollectMoneyFrame** **extends** JFrame {

**private** int Coin[] = {10,50,100,500,1000};

**final** int MAXSIZE=15;

**public** CollectMoneyFrame(AtomicInteger[] stack\_top, AtomicInteger[][] stack\_money, String CoinNames[]

,AtomicInteger machine\_totalmoney){

setTitle("수금");

setSize(500, 500);

setVisible(**true**);

setResizable(**false**);

GridLayout grid = **new** GridLayout(6, 4);

setBackground(Color.pink);

setLayout(grid);

StackClass sc = **new** StackClass();

JLabel CoinNa[] = **new** JLabel[5]; *// 화폐 이름 JLabel*

JLabel Coincount[] = **new** JLabel[5]; *// 화폐 갯수 JLabel*

JButton AddMoney[] = **new** JButton[5]; *// 화폐 추가 button*

JButton SubMoney[] = **new** JButton[5]; *// 화폐 빼기 button*

JLabel TotalMoney = **new** JLabel("총액: "+Integer.toString(machine\_totalmoney.get())); *// 현재 기계 안에화폐 총액 JLabel*

JButton exit = **new** JButton("수금 및 EXIT"); *// 종료 button*

AtomicInteger collectmoney = **new** AtomicInteger(0); *// 수금액 count 변수*

JLabel collectLa = **new** JLabel("수금액: "+collectmoney.get()+" won"); *// 수금액 JLable*

exit.addActionListener(**new** ActionListener() { *// 종료 버튼 액션 method*

**public** void actionPerformed(ActionEvent e) {

FileIo fileio = **new** FileIo();

fileio.CoinWrite(stack\_top);

JOptionPane.showMessageDialog(**null**, Integer.toString(collectmoney.get())+" won을 수금 및 저장했습니다.");

dispose();

}

});

**for**(int i=0;i<5;i++) { *// JLabel , button 초기화*

int index=i;

CoinNa[i] = **new** JLabel(CoinNames[i]);

Coincount[i]= **new** JLabel(Integer.toString(stack\_top[i].get()+1)+"개");

AddMoney[i] = **new** JButton("+");

SubMoney[i] = **new** JButton("-");

AddMoney[i].addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

**if** (sc.is\_full(stack\_top[index])==-1) { *// 화폐 추가 갯수 예외처리*

JOptionPane.showMessageDialog(**null**, "최대치입니다.", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

**else** {

sc.push(stack\_money[index], stack\_top[index], Coin[index], machine\_totalmoney);

Coincount[index].setText(Integer.toString(stack\_top[index].get()+1)+"개");

TotalMoney.setText("총액: "+Integer.toString(machine\_totalmoney.get()));

collectLa.setText("수금액: "+collectmoney.addAndGet(-Coin[index])+" won");

}

}

});

SubMoney[i].addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

**if**(stack\_top[index].get()<=4) { *// coin의 갯수가 5개 초과 일때만 뺄 수 있게 예외처리*

JOptionPane.showMessageDialog(**null**, "최소 5개 이상 있어야 합니다.", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

**else** { *// 스택에서 pop하고 총액과 수금액 count*

sc.pop(stack\_money[index], stack\_top[index],machine\_totalmoney);

Coincount[index].setText(Integer.toString(stack\_top[index].get()+1)+"개");

TotalMoney.setText("총액: "+Integer.toString(machine\_totalmoney.get()));

collectLa.setText("수금액: "+collectmoney.addAndGet(Coin[index])+" won");

}

}

});

**this**.add(CoinNa[i]);

**this**.add(Coincount[i]);

**this**.add(AddMoney[i]);

**this**.add(SubMoney[i]);

}

**this**.add(TotalMoney);

**this**.add(collectLa);

**this**.add(exit);

}

}

CollectMoneyFrame 화폐 현황 및 수금, 추가를 할 수 있는 class이다.

GUI는 다음과 같다.

텍스트, 스크린샷, 번호, 디자인이(가) 표시된 사진

자동 생성된 설명

각 화폐의 원, 갯수 Jlable, 화폐를 추가할 수 있는 +버튼과 화폐를 수금할 수 있는 -버튼, 화폐의 총액을 나타내는 Label, + -버튼으로 수금액을 표현하는 Jlabel, save&exit 버튼으로 이루어져 있다.

+, - 버튼을 누르면 stack money[] 에 push or pop을 진행한다. 그리고 coin의 value만큼 collectmoney 변수 vlaue를 변경하고 수금액 jlabel을 set한다. +는 stackmoney의 maxsize만큼 push할 수 있고 그 이상을 넘을시에는 joptionPane 으로 오류메세지를 출력해주는 예외처리를 했다. ‘-‘ 버튼은 화폐가 최소 5개가 남아야 한다고 설정하여 예외처리를 했다. 아래 사진은 예외처리 구현 gui다.

텍스트, 스크린샷, 운영 체제, 번호이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 운영 체제, 폰트이(가) 표시된 사진

자동 생성된 설명  
10원이 50개인 상태에서 ‘+’버튼을 눌렀을 때와, 5개 있을 때 ‘-‘ 버튼을 눌렀을 때GUI

수금 및 Exit button을 눌렀을 때 Fileio class를 사용하여 Coin.txt의 화폐의 갯수를 저장하고 수금액을 Joption pane으로 출력 후 Frame을 종료한다.

텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

자동 생성된 설명 1000원 ‘-‘버튼을 4번 누르고 exit버튼을 눌렀을 때 사진처럼 Joptionpane으로 출력한다. 만약 수금을 하지 않고 화폐의 갯수만 추가 시키면 음수 값을 수금했다고 출력된다.

**2.7 DrinkPanel**

*/\**

*\* 상품 재고 및 음료 현황, 음료 이름, 음료 가격을 관리하는 JFrame class*

*\*/*

**package** **swing**;

**import** **java.awt.GridLayout**;

**import** **java.awt.event.ActionEvent**;

**import** **java.awt.event.ActionListener**;

**import** **java.util.concurrent.atomic.AtomicInteger**;

**import** **java.util.\***;

**import** **javax.swing.JButton**;

**import** **javax.swing.JFrame**;

**import** **javax.swing.JLabel**;

**import** **javax.swing.JOptionPane**;

**import** **javax.swing.SwingUtilities**;

**class** **DrinkPanel** **extends** JFrame {

**static** **final** int MAX\_SIZE = 15;

**static** **final** int MIN\_SIZE = 3;

**public** DrinkPanel(String [] drinkNames, AtomicInteger[] at\_pd\_count,

AtomicInteger Atprice[]) {

setTitle("음료 현황 및 변경");

setSize(800, 500);

setVisible(**true**);

GridLayout grid = **new** GridLayout(6, 7);

setLayout(grid);

setResizable(**false**);

FileIo fileio = **new** FileIo();

JLabel[] DrinkNaLa = **new** JLabel[5]; *// 음료 이름 Label*

JLabel[] pdcountLa = **new** JLabel[5]; *// 음료 재고 Label*

JLabel[] priceLa = **new** JLabel[5]; *// 가격 Label*

JButton[] sum\_product = **new** JButton[5]; *// 재고 추가 버튼*

JButton[] sub\_product = **new** JButton[5]; *// 재고 반출 버튼*

JButton[] change\_drink\_na = **new** JButton[5]; *// 음료 이름 변경 버튼*

JButton[] change\_drink\_price = **new** JButton[5]; *// 음료 가격 변경 버튼*

JButton exit = **new** JButton("save and exit"); *// 종료 버튼*

**for**(int i=0;i<5;i++) {

int index=i;

DrinkNaLa[i] = **new** JLabel((drinkNames[i]),JLabel.CENTER);

pdcountLa[i]= **new** JLabel((Integer.toString(at\_pd\_count[i].get())+"개"),JLabel.CENTER);

priceLa[i] = **new** JLabel((Integer.toString(Atprice[i].get())+"won"),JLabel.CENTER);

sum\_product[i]= **new** JButton("+");

sub\_product[i]= **new** JButton("-");

change\_drink\_na[i] = **new** JButton("상품 이름 변경");

change\_drink\_price[i] = **new** JButton("상품 가격 변경");

sum\_product[i].addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

**if**(at\_pd\_count[index].get() == MAX\_SIZE) { *// max size 예외처리*

JOptionPane.showMessageDialog(**null**, "재고를 더 넣을 수 없습니다.");

}

**else** {

at\_pd\_count[index].addAndGet(1); *// at\_pd\_count[i] ++*

pdcountLa[index].setText((Integer.toString(at\_pd\_count[index].get())+"개"));

}

}

});

sub\_product[i].addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

**if**(at\_pd\_count[index].get()<=MIN\_SIZE) { *// minsize 예외처리*

JOptionPane.showMessageDialog(**null**, "재고는 최소 3개 이상이어야만 됩니다.");

}

**else** {

at\_pd\_count[index].addAndGet(-1);

pdcountLa[index].setText((Integer.toString(at\_pd\_count[index].get())+"개"));

}

}

});

change\_drink\_na[i].addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

String tmp = JOptionPane.showInputDialog(**null**,"음료 이름을 입력하세요.");

drinkNames[index] = tmp;

DrinkNaLa[index].setText(drinkNames[index]);

}

});

change\_drink\_price[i].addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

String tmp = JOptionPane.showInputDialog(**null**,"음료 가격을 입력하세요.(숫자만)");

**try** {

Atprice[index].set(Integer.parseInt(tmp));

priceLa[index].setText(Integer.toString(Atprice[index].get())+"won");

}

**catch**(NumberFormatException e1){

JOptionPane.showMessageDialog(**null**, "숫자만 입력해 주세요");

}

}

});

}

exit.addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

fileio.DrinkWrite(drinkNames, Atprice, at\_pd\_count);

JOptionPane.showMessageDialog(**null**, "FILE SAVE");

**new** ContentPanEx().invalidate();

dispose();

}

});

**this**.add(**new** JLabel(("이름"),JLabel.CENTER));

**this**.add(**new** JLabel(("재고"),JLabel.CENTER));

**this**.add(**new** JLabel(("가격"),JLabel.CENTER));

**this**.add(**new** JLabel(("재고 투입"),JLabel.CENTER));

**this**.add(**new** JLabel(("재고 반출"),JLabel.CENTER));

**this**.add(**new** JLabel((""),JLabel.CENTER));

**this**.add(exit);

**for**(int i=0;i<5;i++) {

**this**.add(DrinkNaLa[i]);

**this**.add(pdcountLa[i]);

**this**.add(priceLa[i]);

**this**.add(sum\_product[i]);

**this**.add(sub\_product[i]);

**this**.add(change\_drink\_na[i]);

**this**.add(change\_drink\_price[i]);

}

}

}

상품 재고, 상품 이름 변경, 상품 가격 가격을 관리하는 Jframe class이다.

GUI는 아래와 같다.

텍스트, 스크린샷, 번호, 도표이(가) 표시된 사진

자동 생성된 설명

사진처럼 상품 이름, 재고, 상품 가격을 Jlabel, 재고 투입과 재고 반출 Button, 각 상품의 이름 변경과 가격 변경 버튼, exit버튼으로 이루어져 있다.

텍스트, 스크린샷, 로고이(가) 표시된 사진

자동 생성된 설명재고 투입 버튼 기능은 위의 수금 class와 마찬가지로 + 와 – 버튼을 누르면 재고 변수 atpdcount을 변동하고 재고 label을 set한다. 음료의 최소 재고는 3개로 설정했으므로 3개 이하일 때 재고 반출 button을 누르면 Joptionpane으로 오류가 출력되는 예외처리를 했다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

상품 이름을 변경버튼을 누르면 showinputdialog로 입력을 받고 drinkname[]을 변경한다. 상품 가격 변경도 마찬가지인데 숫자가 아닌 문자를 입력하면Numberformatecption으로 예외처리를 했다.

변경을 다 하고 Exit Button을 누르면 fileIo로 drink.txt에 쓰기를 하고 저장한다. 그리고 contentpaneEx를 재실행 위해 invaildata()를 사용했다.

* 1. **SalesFrame**

*/\**

*\* 일별 월별 매출관리 Jframe class*

*\*/*

**package** **swing**;

**import** **java.awt.BorderLayout**;

**import** **java.awt.FlowLayout**;

**import** **java.awt.event.ActionEvent**;

**import** **java.awt.event.ActionListener**;

**import** **java.io.BufferedReader**;

**import** **java.io.FileNotFoundException**;

**import** **java.io.FileReader**;

**import** **java.io.IOException**;

**import** **java.time.LocalDate**;

**import** **javax.swing.JButton**;

**import** **javax.swing.JComboBox**;

**import** **javax.swing.JFrame**;

**import** **javax.swing.JLabel**;

**import** **javax.swing.JOptionPane**;

**import** **javax.swing.JScrollPane**;

**import** **javax.swing.JTable**;

**import** **javax.swing.table.DefaultTableModel**;

**class** **SalesFrame** **extends** JFrame {

**private** DefaultTableModel model;

**private** JTable table;

*//Jtabel 로 구현*

**public** SalesFrame()

{

String colName[] = {"DATE","TIME","DRINK","PRICE"}; *// table에 목록*

String[] year = **new** String[(LocalDate.now()).getYear() - 2000 + 2]; *// 2000년 부터 현재 년도*

String month[] = {"월","01","02","03","04","05","06","07","08","09","10","11","12"};

String day[] = **new** String[32];

day[0] = "일";

day[1] = "01";

day[2] = "02";

day[3] = "03";

day[4] = "04";

day[5] = "05";

day[6] = "06";

day[7] = "07";

day[8] = "08";

day[9] = "09";

year[0]="연";

int index=1;

**for**(int i=10;i<=31;i++) {

day[i]= String.valueOf(i);

}

**for**(int i=2000; i<=(LocalDate.now()).getYear(); i++) {

year[index]= String.valueOf(i);

index++;

}

setTitle("일/월 매출");

setSize(800, 500);

setVisible(**true**);

setResizable(**false**);

setLayout(**new** FlowLayout());

model = **new** DefaultTableModel(colName, 0);

table = **new** JTable(model);

add(**new** JScrollPane(table),BorderLayout.CENTER);

JComboBox yearCombo= **new** JComboBox(year); *// 연도 선택 combobox*

JComboBox monthCombo = **new** JComboBox(month); *// 월 선택 combobox*

JComboBox dayCombo = **new** JComboBox(day); *// 일 선택 combobox*

JButton selectBu = **new** JButton("조회"); *// datafile 선택 버*

JButton exitBu = **new** JButton("exit");

JLabel monthLa = **new** JLabel("월매출"); *// 일매출 label*

JLabel dayLa = **new** JLabel("일매출"); *// 월매출 label*

selectBu.addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

String selectyear = yearCombo.getSelectedItem().toString();

String selectmonth =monthCombo.getSelectedItem().toString();

String selectday = dayCombo.getSelectedItem().toString();

String selectdate = selectyear+"-"+selectmonth+"-"+selectday;

model.setRowCount(0);

SalesRead(selectyear+"-"+selectmonth); *// YYYY-MM.txt 읽어오는 method*

int columnIndex = 3; *// 4번째 열의 인덱스*

int month\_total = 0;

int day\_total=0; *// 읽어온 data를 jtable 에 구현*

**for** (int row = 0; row < model.getRowCount(); row++) {

String value = (String) model.getValueAt(row, columnIndex);

month\_total+= Integer.parseInt(value);

}

**for**(int row= 0; row < model.getRowCount(); row++) {

String valuedate = (String) model.getValueAt(row, 0);

**if**(selectdate.equals(valuedate)==**true**) {

String value = (String) model.getValueAt(row, columnIndex);

day\_total+=Integer.parseInt(value);

}

}

*// 월매출 set*

monthLa.setText(selectmonth+"월: "+Integer.toString(month\_total)+"원");

*// 일매출 set*

dayLa.setText(selectday+"일: "+Integer.toString(day\_total)+"원");

}

});

exitBu.addActionListener(**new** ActionListener() {

**public** void actionPerformed(ActionEvent e) {

dispose();

}

});

**this**.add(yearCombo,BorderLayout.EAST);

**this**.add(monthCombo,BorderLayout.EAST);

**this**.add(dayCombo,BorderLayout.EAST);

**this**.add(selectBu,BorderLayout.EAST);

**this**.add(exitBu,BorderLayout.SOUTH);

**this**.add(monthLa,BorderLayout.NORTH);

**this**.add(dayLa,BorderLayout.NORTH);

}

*// 매출 datafile 읽어오는 method*

void SalesRead(String date){

String filename= "src/DB/"+date+".txt"; *// datafile name*

*//System.out.println(filename);*

**try** {

BufferedReader reader = **new** BufferedReader(**new** FileReader(filename));

String strtmp;

**while** ((strtmp = reader.readLine()) != **null**) {

String[] parts = strtmp.split("\t");

model.addRow(parts);

}

reader.close();

} **catch**(FileNotFoundException e){

JOptionPane.showMessageDialog(**null**, "FILE DATA OPEN ERROR", "ERROR", JOptionPane.ERROR\_MESSAGE);

} **catch** (IOException e) {

JOptionPane.showMessageDialog(**null**, "FILE READ ERROR", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

}

}

SalesFrame은 물품을 구매했을 때 write한 file을 Jtable에 값을 읽어와 매출 기록과, 일매출, 월매출을 확인할 수 있는 Jframe class이다. GUI는 다음과 같다.

텍스트, 소프트웨어, 컴퓨터, 운영 체제이(가) 표시된 사진

자동 생성된 설명

Jtable, combobox 연, 월, 일, 조회 button, exti button, 월매출, 일매출 Jlable로 구성되어있다.

Combobox에서 연 월 일을 선택 후 조회 button을 클릭하면 Combobox value값을 읽어와 Filename으로 string 처리 후 해당 년,월 데이터 파일을 Jtable로 불러온다. 그리고 아래 Jlabel 에 해당 월 매출과 일 매출이 갱신된다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

조회 버튼을 누를시 Date, Time, Drinkname, price 순으로 table이 갱신된다.

Jtable에는 해당월 매출 기록이 전부 표시가 된다.

exit에는 해당 월 매출 77380원, 해당 일 매출 40030원을 확인할 수 있다.

( 5월 매출 31일만 6월 매출 file밖에 없으므로 확인을 부탁드립니다.)

**2.9 FileIo**

*/\**

*\* 파일 입출력 method 를 구현한 class*

*\*/*

**package** **swing**;

**import** **java.io.BufferedReader**;

**import** **java.io.BufferedWriter**;

**import** **java.io.File**;

**import** **java.io.FileNotFoundException**;

**import** **java.io.FileReader**;

**import** **java.io.FileWriter**;

**import** **java.io.IOException**;

**import** **java.time.LocalDate**;

**import** **java.time.LocalTime**;

**import** **java.time.format.DateTimeFormatter**;

**import** **java.util.concurrent.atomic.AtomicInteger**;

**import** **javax.swing.JOptionPane**;

**class** **FileIo** {

**private** **static** **final** int EXIT\_ON\_CLOSE = 0;

void CoinWrite(AtomicInteger[] stack\_top)

{

**try** {

*// 1. 파일 객체 생성*

File file = **new** File("src/DB/coin.txt");

FileWriter fw = **new** FileWriter(file);

BufferedWriter writer = **new** BufferedWriter(fw);

**for**(int i=0; i<5;i++) {

writer.write(Integer.toString(stack\_top[i].get()+1)+"\t");

}

writer.close();

} **catch**(FileNotFoundException e){

System.out.println("File open error");

JOptionPane.showMessageDialog(**null**, "FILE DATA OPEN ERROR", "ERROR", JOptionPane.ERROR\_MESSAGE);

System.exit(EXIT\_ON\_CLOSE);

} **catch** (IOException e) {

JOptionPane.showMessageDialog(**null**, "FILE READ ERROR", "ERROR", JOptionPane.ERROR\_MESSAGE);

System.exit(EXIT\_ON\_CLOSE);

}

}

void CoinRead(AtomicInteger[] stack\_top, AtomicInteger[][] stack\_money,

int Coin[] , AtomicInteger machine\_totalmoney)

{

**try** {

BufferedReader reader = **new** BufferedReader(**new** FileReader("src/DB/coin.txt"));

String strtmp = **null**;

strtmp = reader.readLine();

String[] parts = strtmp.split("\t");

**for**(int i=0;i<5;i++) {

int count = Integer.parseInt(parts[i].trim());

**for**(int j=0;j<count;j++) {

StackClass stackclass = **new** StackClass();

stackclass.push(stack\_money[i], stack\_top[i], Coin[i], machine\_totalmoney);

}

}

reader.close();

} **catch**(FileNotFoundException e){

System.out.println("File open error");

JOptionPane.showMessageDialog(**null**, "FILE DATA OPEN ERROR", "ERROR", JOptionPane.ERROR\_MESSAGE);

System.exit(EXIT\_ON\_CLOSE);

} **catch** (IOException e) {

JOptionPane.showMessageDialog(**null**, "FILE READ ERROR", "ERROR", JOptionPane.ERROR\_MESSAGE);

System.exit(EXIT\_ON\_CLOSE);

}

}

void DrinkRead(String [] drinkNames, AtomicInteger[] at\_pd\_count,

AtomicInteger Atprice[])

{

**try** {

BufferedReader reader = **new** BufferedReader(**new** FileReader("src/DB/drink.txt"));

String strtmp = **null**;

int index = 0;

**while** ((strtmp = reader.readLine()) != **null** && index < drinkNames.length) {

String[] parts = strtmp.split("\t");

**if** (parts.length == 3) {

drinkNames[index] = parts[0].trim();

Atprice[index] = **new** AtomicInteger(Integer.parseInt(parts[1].trim()));

at\_pd\_count[index]= **new** AtomicInteger(Integer.parseInt(parts[2].trim()));

index++;

}

}

reader.close();

} **catch** (FileNotFoundException e) {

System.out.println("File open error");

JOptionPane.showMessageDialog(**null**, "FILE DATA OPEN ERROR", "ERROR", JOptionPane.ERROR\_MESSAGE);

System.exit(EXIT\_ON\_CLOSE);

} **catch** (IOException e) {

System.out.println("File read error");

JOptionPane.showMessageDialog(**null**, "FILE READ ERROR", "ERROR", JOptionPane.ERROR\_MESSAGE);

System.exit(EXIT\_ON\_CLOSE);

}

}

void DrinkWrite(String []drinkNames, AtomicInteger[] price, AtomicInteger[] at\_pd\_count)

{

**try** {

*// 1. 파일 객체 생성*

File file = **new** File("src/DB/drink.txt");

*// 3. Buffer를 사용해서 File에 write할 수 있는 BufferedWriter 생성*

FileWriter fw = **new** FileWriter(file);

BufferedWriter writer = **new** BufferedWriter(fw);

*// 4. 파일에 쓰기*

**for**(int i=0;i<5;i++) {

writer.write(drinkNames[i]+"\t"+price[i].get()+"\t"+at\_pd\_count[i].get());

writer.newLine();

}

writer.close();

} **catch**(FileNotFoundException e){

System.out.println("File open error");

JOptionPane.showMessageDialog(**null**, "FILE DATA OPEN ERROR", "ERROR", JOptionPane.ERROR\_MESSAGE);

System.exit(EXIT\_ON\_CLOSE);

} **catch** (IOException e) {

JOptionPane.showMessageDialog(**null**, "FILE WRITE ERROR", "ERROR", JOptionPane.ERROR\_MESSAGE);

System.exit(EXIT\_ON\_CLOSE);

}

}

*// 매출 기록 write method*

void SalesWrite(String drinkname, int price)

{

LocalDate nowdate = LocalDate.now();

LocalTime nowtime = LocalTime.now();

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("HH:mm:ss");

String formatedNow = nowtime.format(formatter);

DateTimeFormatter Dateformatter = DateTimeFormatter.ofPattern("yyyy-MM");

String filename ="src/DB/"+nowdate.format(Dateformatter)+".txt"; *// filename 현재시간 설정*

**try** { *// file에 YYYY-MM-DD HH:mm:ss drinkname price를 write*

File file = **new** File(filename); *// 파일이 존재하지 않으면 새 파일을 만들음*

**if**(!file.exists()) file.createNewFile();

FileWriter fw = **new** FileWriter(file,**true**);

BufferedWriter writer = **new** BufferedWriter(fw);

writer.write(nowdate+"\t"+formatedNow+"\t"+drinkname+"\t"+price);

writer.newLine();

writer.close();

} **catch** (IOException e) {

JOptionPane.showMessageDialog(**null**, "FILE WRITE ERROR", "ERROR", JOptionPane.ERROR\_MESSAGE);

System.exit(EXIT\_ON\_CLOSE);

}

}

}

FileIo class는 각종 file의 read, write의 method를 정의 한 클래스이다.

크게 설명할 것이 없지만 Sales Write() 부분만 다른 method부분이라 살짝 다름으로 이 부분만 설명하겠다. Sales Write method는 물품 구매 버튼이 있는 productpanel에서 사용하는 method이다. 물품이 구매되면 data파일에 매출을 한줄씩 기록하는 방식이다. 하그래서 물품을 구매한 년도와 월 시간 값을 구하는 LocalDate, LocalTime을 사용했다. 구한 value값으로 string 으로 file data name을 설정하고 그 name file에 접근하도록 했다. 하지만 만약 저번달 매출 file만 존재하고 이번달 매출 파일이 존재하지 않을 수도 있으니 file.exists()를 사용하여 존재하지 않으면 file data name(YYYY-MM.txt) 를 생성하도록 만들었다.

**2.10 StackClass**

*/\**

*\* money stack 관련 push, pop method class method 구현class*

*\*/*

**package** **swing**;

**import** **java.util.concurrent.atomic.AtomicInteger**;

**import** **javax.swing.JLabel**;

**class** **StackClass** {

**static** **final** int MAX\_STACK = 50 ;

int is\_full(AtomicInteger top) {

**if**(top.get()>=MAX\_STACK-1) {

**return** -1;

}

**else** **return** 1;

}

int is\_empty(AtomicInteger top) {

**if**(top.get()==-1) {

**return** -1;

}

**else** {

**return** 1;

}

}

void push(AtomicInteger[] stack\_money, AtomicInteger top,

int data, AtomicInteger machine\_totalmoney) {

**if** (is\_full(top) == 1) {

top.getAndAdd(1);

stack\_money[top.get()].set(data);

machine\_totalmoney.addAndGet(data);

}

}

int pop(AtomicInteger stack\_money[], AtomicInteger top, AtomicInteger machine\_totalmoney) {

**if**(is\_empty(top)==1) {

top.getAndAdd(-1);

machine\_totalmoney.addAndGet(-(stack\_money[top.get()+1].get()));

**return** stack\_money[top.get()+1].get();

}

**else** **return** 0;

}

*// 물품을 구매할 때 잔돈을 반환해줄 수 있는지 확인해주는 method*

boolean is\_return(AtomicInteger[][] stack\_money, AtomicInteger[] top, AtomicInteger machine\_totalmoney, int[] Coin, AtomicInteger money, AtomicInteger price) {

int tmp\_money = money.get()-price.get();

int index = -1;

**for** (int i = Coin.length - 1; i >= 0; i--) {

**if** (Coin[i] <= tmp\_money) {

index = i;

**break**;

}

}

**for** (int i = index; i >= 0; i--) {

int popcount = 0;

**while** (top[i].get() != -1) {

tmp\_money -= pop(stack\_money[i], top[i], machine\_totalmoney);

popcount++;

}

**for** (int j = 0; j < popcount; j++) {

push(stack\_money[i], top[i], Coin[i], machine\_totalmoney);

}

}

**return** tmp\_money <= 0;

}

}

stackclass는 moneystack에서 push pop method를 사용하기 위해 위해 만든 class이다. 전형적인 stack method이지만 is\_return이라는 method의 쓰임에 대해서 설명하겠다.

is\_return은 물품을 구매하기 전에 잔돈을 반환할 수 있는지 체크하는 method이다. tmp\_money에 투입금액 – 상품금액 으로 설정한다. 가장 큰 화폐 1000원부터 10원까지 tmp\_money에 – 한 후 다시 원래 금액만큼 push를 한다. tmp\_money가 0이 되면 잔돈을 반환할 수 있는 상태이므로 true를 반환하고 아니면 false를 반환한다. 이것으로 잔돈을 반환할 수 있는지 없는지 확인이 가능하다.