**Coin System**

**Brief description**

Coin system is such a management system in java which tells the shopkeeper that how much change for money he has to return back to customer. The system also keeps the track of the coins available in the stock. Shopkeeper can add and check the status of stock in real time.

**Classes**

Server\_Coins.java:-

Server class is the class which has a main method that creates an instance of the remote object implementation, exports the remote object, and then binds that instance to a name in a Java RMI registry. After starting the server, it will show a message “Server is up and running” in the console.

Client\_Coin\_login.java:-

To use the system as RMI there has to be a client in our case its Client\_Coin\_login.java file to use system on a machine. Which will ask for username and password which is varun, 12345 respectively. After login successfully second frame (Coin\_Sysytem.java) will appear.

Implement\_Coin.java:-

Implement class is having all the calculation in order to find the answer of each functionality. This class is having 3 method

1. addValue :-

addValue takes the values from textfield and adds the value to an array that array represents the stock.

1. exchange:-

exchange method calculates the numbers of coins to be given back to customer.

1. exchange2

To return back the calculated result from the exchange method in the label exchange2 is used.

Interface\_Coin.java: -

It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.

**Code Snippets**

Server: -

**public** **class** Server\_Coin {

**public** **static** **void** main(String args[]) **throws** Exception {

String unpass = "varun12345";

LocateRegistry.*createRegistry*(1099);

System.***out***.println("Server is up and running");

Implement\_Coin n1 = **new** Implement\_Coin();

Naming.*rebind*(unpass, n1);

}

}

Interface: -

**interface** Interface\_Coin **extends** Remote {

**public** **int**[] ***coins*** = **new** **int**[4];

**public** **int**[] ***resultcoins*** = **new** **int**[4];

**public** **int**[] addValue(**int**[] variable) **throws** RemoteException;

**public** **int**[] exchange(Double result) **throws** RemoteException;

**public** **int**[] exchange2(**int**[] coins) **throws** RemoteException;

}

Implement: -

**public** **class** Implement\_Coin **extends** UnicastRemoteObject **implements** Interface\_Coin {

**private** **int** counter;

Console console = System.*console*();

**protected** Implement\_Coin() **throws** RemoteException {

counter = 0;

}

// adding coins in the stock array

**public** **int**[] addValue(**int**[] veri) **throws** RemoteException {

**for** (**int** i = 0; i < ***coins***.length; i++) {

***coins***[i] += veri[i];

}

**return** ***coins***;

}

// calculate the result of coins and returns the result back to

// Client\_Coin\_System.java class

**public** **int**[] exchange(Double result) **throws** RemoteException {

**int**[] resultcoins = {0,0,0,0};

**double** roundOff = (**double**) Math.*round*(result \* 100) / 100;

**int** amt = (**int**) (roundOff \* 100);

**int**[] cents = { 25, 10, 5, 1 };

**int** count = 0;

**for** (**int** i = 0; i < ***coins***.length; i++) {

**if** (***coins***[i] > 0 ) {

count = amt / cents[i];

amt = amt % cents[i];

resultcoins[i] = count;

}

**else** {

System.***out***.println("You ran out of " + cents[i] + " cents");

}

}

**return** resultcoins;

}

// updating the stocks

**public** **int**[] exchange2(**int**[] resultArray) {

**for** (**int** j = 0; j < ***coins***.length; j++) {

**if** ((***coins***[j] > 0)) {

***coins***[j] -= resultArray[j];

}

}

**return** ***coins***;

}

Login: -

String un = textField.getText();

**int** pass = Integer.*parseInt*(textField\_1.getText());

**try** {

String url = "rmi:///";

Interface\_Coin inter = (Interface\_Coin) Naming.*lookup*(url +un+pass);

//inter.logcheck();

Coin\_System c = **new** Coin\_System();

c.frame.setVisible(**true**);

} **catch** (Exception e1) {

e1.printStackTrace();

}

frame.dispose();

**Calculate: -**

Double bill = Double.*parseDouble*(textField.getText());

Double cash = Double.*parseDouble*(textField\_1.getText());

Double result = cash - bill;

// Label lblNewLabel\_4 will show the cents to return back

lblNewLabel\_4.setText("" + String.*format*("%.2f", result)); // here Implement\_Coin.java class is called to calculate the result and to display the result to its proper place

**try** {

resultarr = c.exchange(result);

**for** (**int** i = 0; i < resultLable.length; i++) {

resultLable[i].setText("" + resultarr[i]);

}

resultarr2 = c.exchange2(resultarr);

**for** (**int** i = 0; i < Lablename.length; i++) {

// resultLable[i].setText("" +resultarr2[i]);

**if**(resultarr2[i]>-1){

Lablename[i].setText("" + resultarr2[i]);

}

**else**{

System.***out***.println("Not enough money to return back!!");

}

}

} **catch** (RemoteException e1) {

e1.printStackTrace();

}

**Add Stock: -**

// accepting the values from text fields through array

JTextField[] textFieldname = { textField\_2, textField\_3, textField\_4, textField\_5 };

**int**[] veri\_value = **new** **int**[4];

**for** (**int** i = 0; i < textFieldname.length; i++) {

**int** virtualDirectories = Integer.*parseInt*(textFieldname[i].getText());

veri\_value[i] = virtualDirectories;

}

// here Implement\_Coin.java class is called to pass the array and add coins in the stock

**try** {

arr = c.addValue(veri\_value);

**for** (**int** i = 0; i < Lablename.length; i++) {

Lablename[i].setText("" + arr[i]);

}

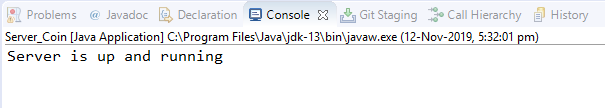
} **catch** (RemoteException e1) {

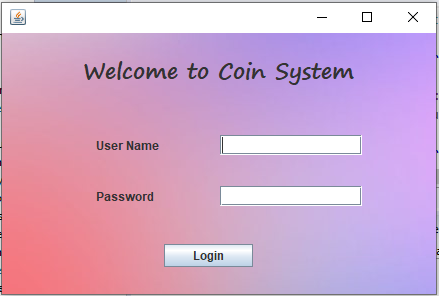
e1.printStackTrace();

}

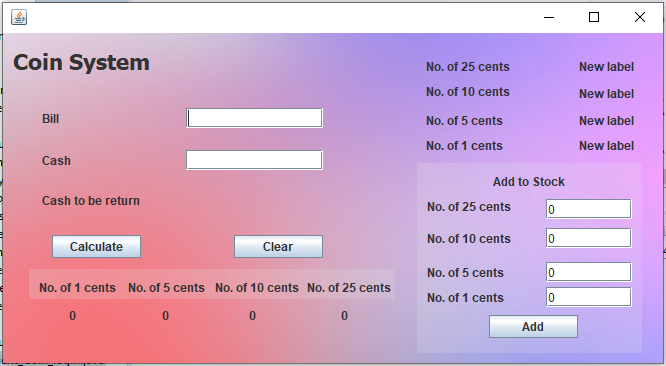
**Screenshots**

**1.Start the server**

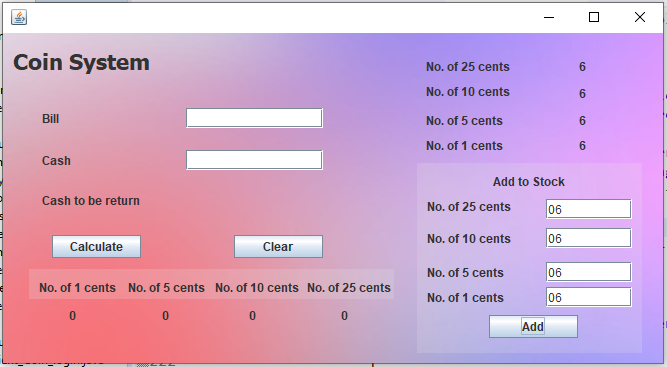


**2. run Client\_Coin\_login.javan**

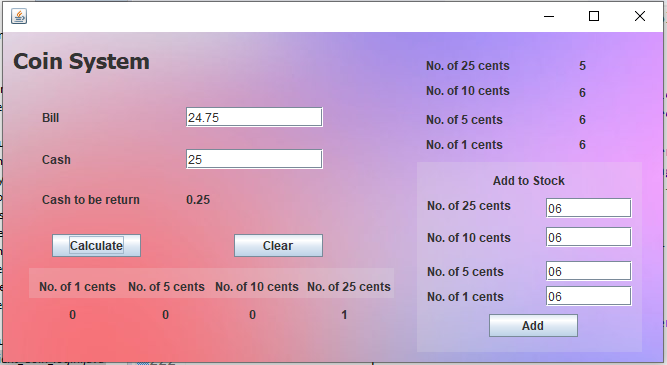
1. **Second Frame**



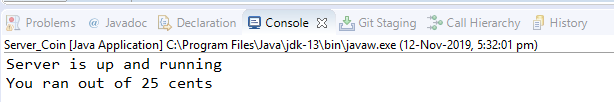
1. **Add stocks**

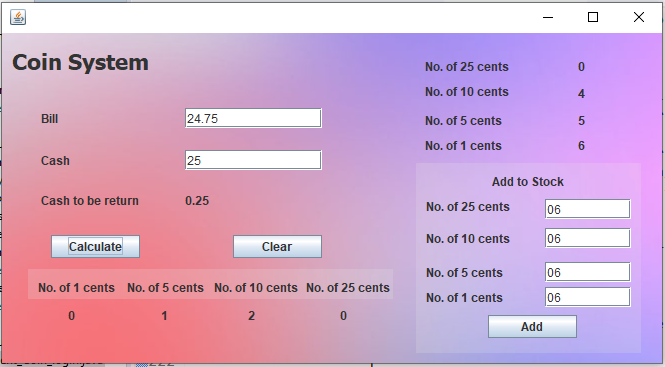


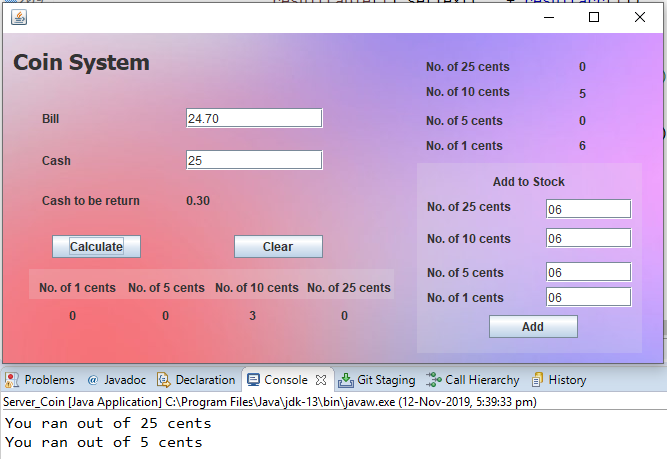
1. **Calculate button**



1. **Error message**







**Conclusions: -**

My Project, The Coin system is a Change management system which will help shopkeepers whoever will use this system with Change that has to be returned to the customer if the Customer gives more cash.

I have encountered a few problems during the development of this project which were mostly of Programming perspective. Some of them are:

* I faced difficulty in getting the data off Textfield which had to be stored in the Array.
* There were difficulties in Implementing RMI on the System as the server was not running. And also configuration of server and client was difficult.

After studying and learning more about these problems, I was successful in overcoming them.

In all, developing this project was a great experience and I was able to learn functionalities in Java like RMI, Array list, Swing and Frame.