2021

Report by: Danyal Arif (SP20-BCS-020)

Report to: Ma'am Nusrat Shaheen

OOP-Project Report

Freelance Panda Application

PROJECT DESCRIPTION:

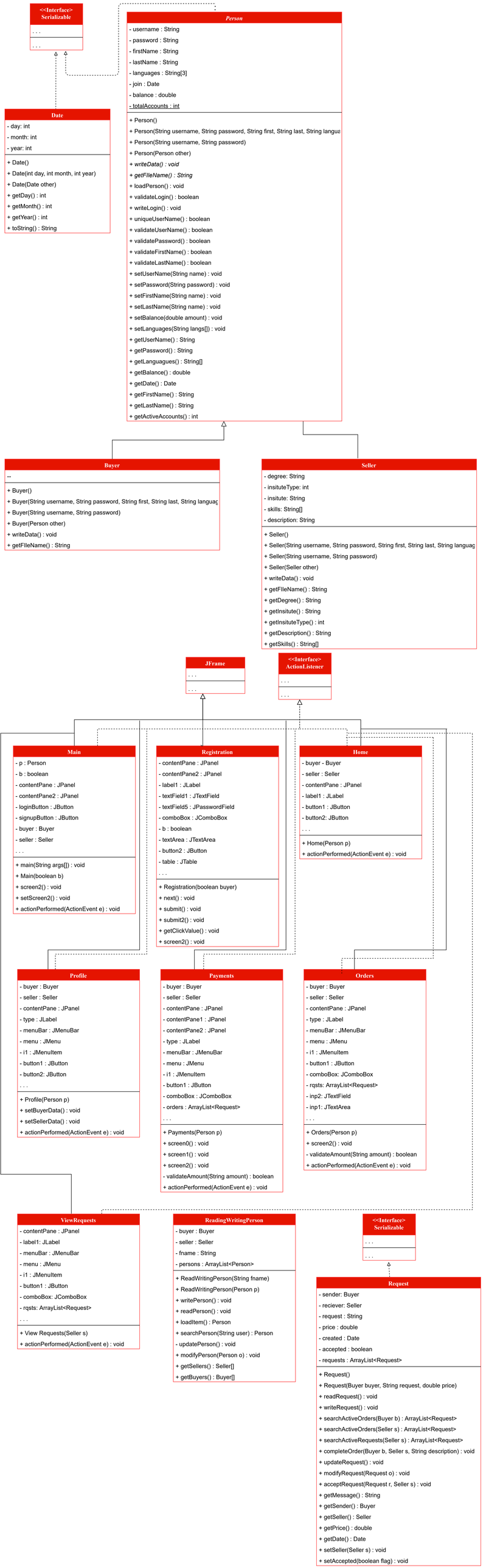
Freelance Panda is a GUI and OOP-based freelancing application in which buyers and sellers can communicate with each other and share their services. Buyers and Sellers can register themselves in the application. To log in, they can use credentials. Some features are common to both account types. Buyers and sellers have access to a home page. Their usernames with a list of other sellers/buyers get displayed on a home page. They have access to the profile section that they can use to view their complete profile. Both buyers and sellers can access orders in which their currently active orders get displayed. A buyer can send a request, which is like a message. In the request section, a buyer can specify their demand. They can post their required work description along with a price. Once sent, the request will get displayed to all available sellers. Sellers can view and accept the buyers' requests. Once it gets accepted, it gets removed from the requests section and, it becomes an order. Buyers and sellers involved in the order can view it on their orders screen. Buyers and Sellers have access to the payments screen in which they can check their available balance. A seller has the option of withdrawing only. A buyer can deposit and withdraw money from/into their account. Buyers can pay sellers. In this screen, a buyer gets the list of all those sellers involved in active order. Once the buyer pays the seller, that order is marked completed and removed. Buyer and Sellers can also restore their forgotten passwords through their username. The front-end of the application got designed using java swing and awt. The backend of the application got built using OOP technologies and concepts. It compromised 13 classes in total.

Concepts/Features:

Concepts implemented in the making of application are as follows:

i. GUI and Event Handling
ii. Classes and Objects
iii. Method and Constructor overloading
iv. Copy Constructors
v. Encapsulation
vi. Composition
vii. Inheritance
viii. Method Overriding
ix. Abstract Classes
x. Polymorphism
xi. Interfaces
xii. Array List
xiii. File Handling
xiv. File Handling with Serialization
xv. Exception and Error Handling

UML diagram:


<<Interface>>
Serializable
. . .
. . .
Person
- username : String
- password : String
- firstName : String
- lastName : String
- languages : String[3]
- join : Date
- balance : double
- totalAccounts : int
+ Person()
+ Person(String username, String password, String first, String last, String langua
+ Person(String username, String password)
+ Person(Person other)
+ writeData() : void
+ getFIleName() : String
+ loadPerson() : void
+ validateLogin() : boolean
+ writeLogin() : void
+ uniqueUserName() : boolean
+ validateUserName() : boolean
+ validatePassword() : boolean
+ validateFirstName() : boolean
+ validateLastName() : boolean
+ setUserName(String name) : void
+ setPassword(String password) : void
+ setFirstName(String name) : void
+ setLastName(String name) : void
+ setBalance(double amount) : void
+ setLanguages(String langs[]) : void
+ getUserName() : String
+ getPassword() : String
+ getLanguagues() : String[]
+ getBalance() : double
+ getDate() : Date
+ getFirstName() : String
+ getLastName() : String
+ getActiveAccounts() : int
Date
- day: int
- month: int
- year: int
+ Date()
+ Date(int day, int month, int year)
+ Date(Date other)
+ getDay() : int
+ getMonth() : int
+ getYear() : int
+ toString() : String
Buyer
--
+ Buyer()
+ Buyer(String username, String password, String first, String last, String languag
+ Buyer(String username, String password)
+ Buyer(Person other)
+ writeData() : void
+ getFIleName() : String
Seller
- degree: String
- insituteType: int
- insitute: String
- skills: String[]
- description: String
+ Seller()
+ Seller(String username, String password, String first, String last, String languag
+ Seller(String username, String password)
+ Seller(Seller other)
+ writeData() : void
+ getFIleName() : String
+ getDegree() : String
+ getInsitute() : String
+ getInsituteType() : int
+ getDescription() : String
+ getSkills() : String[]
JFrame
. . .
. . .
<<Interface>>
ActionListener
. . .
. . .
Main
- p : Person
- b : boolean
- contentPane : JPanel
- contentPane2 : JPanel
- loginButton : JButton
- signupButton : JButton
- buyer : Buyer
- seller : Seller
. . .
+ main(String args[]) : void
+ Main(boolean b)
+ screen2() : void
+ setScreen2() : void
+ actionPerformed(ActionEvent e) : void
Registration
- contentPane : JPanel
- contentPane2 : JPanel
- label1 : JLabel
- textField1 : JTextField
- textField5 : JPasswordField
- comboBox : JComboBox
- b : boolean
- textArea : JTextArea
- button2 : JButton
- table : JTable
. . .
+ Registration(boolean buyer)
+ next() : void
+ submit() : void
+ submit2() : void
+ getClickValue() : void
+ screen2() : void
Home
- buyer - Buyer
- seller : Seller
- contentPane : JPanel
- label1 : JLabel
- button1 : JButton
- button2: JButton
. . .
+ Home(Person p)
+ actionPerformed(ActionEvent e) : void
Profile
- buyer : Buyer
- seller : Seller
- contentPane : JPanel
- type : JLabel
- menuBar : JMenuBar
- menu : JMenu
- i1 : JMenuItem
- button1 : JButton
- button2: JButton
. . .
+ Profile(Person p)
+ setBuyerData() : void
+ setSellerData() : void
+ actionPerformed(ActionEvent e) : void
Payments
- buyer : Buyer
- seller : Seller
- contentPane : JPanel
- contentPane1 : JPanel
- contentPane2 : JPanel
- type : JLabel
- menuBar : JMenuBar
- menu : JMenu
- i1 : JMenuItem
- button1 : JButton
- comboBox : JComboBox
- orders : ArrayList<Request>
. . .
+ Payments(Person p)
+ screen0() : void
+ screen1() : void
+ screen2() : void
- validateAmount(String amount) : boolean
+ actionPerformed(ActionEvent e) : void
Orders
- buyer : Buyer
- seller : Seller
- contentPane : JPanel
- type : JLabel
- menuBar : JMenuBar
- menu : JMenu
- i1 : JMenuItem
- button1 : JButton
- comboBox: JComboBox
- rqsts: ArrayList<Request>
- inp2: JTextField
- inp1: JTextArea
. . .
+ Orders(Person p)
+ screen2() : void
- validateAmount(String amount) : boolean
+ actionPerformed(ActionEvent e) : void
ViewRequests
- contentPane : JPanel
- label1: JLabel
- menuBar : JMenuBar
- menu : JMenu
- i1 : JMenuItem
- button1 : JButton
- comboBox: JComboBox
- rqsts: ArrayList<Request>
. . .
+ View Requests(Seller s)
+ actionPerformed(ActionEvent e) : void
ReadingWritingPerson
- buyer : Buyer
- seller : Seller
- fname : String
- persons : ArrayList<Person>
+ ReadWritingPerson(String fname)
+ ReadWritingPerson(Person p)
+ writePerson() : void
+ readPerson() : void
+ loadItem() : Person
+ searchPerson(String user) : Person
- updatePerson() : void
+ modifyPerson(Person o) : void
+ getSellers() : Seller[]
+ getBuyers() : Buyer[]
<<Interface>>
Serializable
. . .
. . .
Request
- sender: Buyer
- reciever: Seller
- request : String
- price : double
- created : Date
- accepted : boolean
- requests : ArrayList<Request>
+ Request()
+ Request(Buyer buyer, String request, double price)
+ readRequest() : void
+ writeRequest() : void
+ searchActiveOrders(Buyer b) : ArrayList<Request>
+ searchActiveOrders(Seller s) : ArrayList<Request>
+ searchActiveRequests(Seller s) : ArrayList<Request>
+ completeOrder(Buyer b, Seller s, String description) : void
+ updateRequest() : void
+ modifyRequest(Request o) : void
+ acceptRequest(Request r, Seller s) : void
+ getMessage() : String
+ getSender() : Buyer
+ getSeller() : Seller
+ getPrice() : double
+ getDate() : Date
+ setSeller(Seller s) : void
+ setAccepted(boolean flag) : void

CLASSES

I. Person Class:

Person class is an abstract class used for storing/processing common elements of buyers and sellers. This class implements the Serializable interface as the objects of this class are written/read during the execution of project.

Data members:

private String username
private String password
private String firstName
private String lastName
private String languages[]
private Date join
private double balance
private static int *totalAccounts* = 0

These data members hold the basic information of buyer/seller.

Constructors:

public Person()
public Person(String username, String password, String first, String last, String languages[])
public Person(String username, String password)
public Person(Person other)

First constructor will never be called in our program. 2nd constructor sets up the basic data of person. Third is used for validating/loading data and 4th is the copy constructor.

Abstract Methods:

public abstract void writeData();
public abstract String getFileName();

These methods will be filled by its sub-classes and will be called polymorphically.

Methods:

public void LoadPerson()
public boolean validateLogin()
public void writeLogin()
public boolean uniqueUserName()
public boolean validateUserName()
public boolean validatePassword()
public boolean validateFirstName()
public boolean validateLastName()
public String getUserName()
public String getPassword()
public String getFirstName()
public String getLastName()
public Strin g[] getLanguages()
public Date getDate()
public double getBalance()

public void setBalance(double amount)
public void setPassword(String password)
public void setFirstName(String name)
public void setLastName(String name)
public void setUserName(String name)
public void setLanguages(String langs[])
public static int getActiveAccounts()

These methods are responsible for proper working of this class. Each change in class attributes through setters is immediately modified in file as well hence changes are permanent. Moreover, it has the required getters and validation methods. Other methods are for writing/reading login info.

II. Buyer Class:

Buyer class extends the Person class and represents a buyer. Since its super class (Person) implements Serializable, Buyer implements it as well. Its physical objects will be written/read through another class.

Data members:

None

It contains no data members of its own as all its required data members are inherited from its super class.

Constructors:

public Buyer()
public Buyer(String username, String password, String first, String last, String languages[])
public Buyer(String username, String password)
public Buyer(Buyer other)

Its constructors are like the Person class and all of them call their super class.

Methods:

public void writeData()
public String getFileName()

It overrides and defines the abstract methods of its super class. WriteData writes objects of buyers through another class.

III. Seller Class:

Seller class is a blueprint for a seller. Its objects are responsible for representing actual sellers. It extends the Person class.

Data members:

private String degree
private int insituteType
private String insitute
private String skills[]
private String description

It inherits all members of its super class and contains these additional data members required for seller.

Constructors:

public Seller()
public Seller(String username, String password, String first, String last, String languages[], String degree, int insituteType, String insitute, String skills[], String descp)
public Seller(String username, String password)
public Seller(Seller other)

Constructors of this class are designed to initialize the basic data of a seller using the person class and its own data members. All required constructors are included.

Methods:

public void writeData()
public String getFileName()
public String toString()
public String getDescription()
public String[] getSkills()
public String getInsitute()
public int getInsituteType()
public String getDegree()

IV. ReadingWritingPerson Class:

This class is responsible for managing the object writing/reading operations on the person class. It is used to write and read objects of buyers and sellers.

Data members:

private Buyer buyer
private Seller seller
private String fname
private ArrayList<Person> persons

where fname is the name of file to be opened buyer/seller indicates the current person whereas the array is written to the file.

Constructors:

public ReadingWritingPerson(String fname)
public ReadingWritingPerson(Person p)

First constructor opens the input file (sellers.txt/buyers.txt), while the other constructor initializes buyer/seller and opens the respective file.

Methods:

public void writePerson()
public void readPerson()
public Person LoadItem()
public Person searchPerson(String user)
private void updatePerson()
public void modifyPerson(Person o)
public Seller [] getSellers()

public Buyer [] getBuyers()

These methods are responsible for reading/writing/searching/modifying both sellers' and buyers' objects in their respective files.

V. Date Class:

Date class is used to storing dates; for example, the date of a person signing up.

Data members:

private int day
private int month
private int year

Constructors and Methods:

public Date ()
public Date (int day, int month, int year)
public Date (Date other)
public int getDay()
public int getMonth()
public int getYear()
public String toString()

VI. Home Class:

This class is used for displaying and managing the home screen of buyer/seller. It handles both the front-end and back-end of all elements related to home. This class extends JFrame as it represents a frame. Moreover, it implements the ActionListener interface since there are multiple events involved.

Data members:

private Buyer buyer
private Seller seller
private JPanel contentPane
private JLabel label1
private JLabel label2
private JLabel label3
private JButton button1
private JButton button2
private JButton button3
private JButton button4
private JButton button5
private JButton button6
private JLabel lblNewLabel
private JLabel type

Most of these data members hold different components that are presented on the screen. The current person is stored in its respective variable, for example, buyer/seller.

Constructor:

public Home (Person p)

It has only one constructor which initializes the person and loads it into the home screen. Moreover, it loads all the GUI components.

Methods:

public void actionPerformed(ActionEvent e)

It has only one method which is responsible for handling all the events.

VII. Profile Class:

This class is responsible for loading and displaying the data of buyer/seller. It can display data of both separately.

Data members:

private JPanel contentPane
private JLabel label1, label2, label3, label4, ans1, ans2, ans3, ans4, ans5
private Buyer buyer
private Seller seller
private JLabel label5
private JLabel ans6
private JLabel label6
private JLabel ans7
private JLabel label7
private JLabel label8
private JLabel label9
private JLabel label10
private JLabel ans8
private JLabel ans9
private JLabel ans10
private JLabel ans11
private JMenuBar menuBar
private JMenu menu
private JMenuItem i1, i2, i3, i4, i5, i6
private JLabel type

These are the core components of the home screen and buyer, seller.

Constructors:

public Profile (Person p)

This constructor loads the data of current person and initializes all the components that will be displaying the data. It then loads the data in these components.

Methods:

public void setBuyerData()
public void setSellerData()
public void actionPerformed (ActionEvent e)

First two functions are accountable for assigning the buyer/seller data to components which is then displayed. Third is used for handling events.

VIII. Registration Class:

Registration class provides the front-end for a buyer/seller to register. Both buyer and seller register through this class. Their entered information is immediately saved in the respective files and their account is created. Both sellers and buyers have different interface. As less information is required from the buyer therefore only one screen exists in case of buyer. However, in case of seller additional data is required hence seller is required to fill information in two screens. This class extends JFrame for GUI components.

Data members:

private Person p
private JPanel contentPane
private JTextField textField1
private JLabel label1
private JLabel label2
private JLabel label3
private JLabel label4
private JTextField textField2
private JTextField textField3
private JPasswordField textField4
private JLabel label5
private JPasswordField textField5
private JTable table
private JLabel label6
private JButton button
private JScrollPane scrollPane
private int langcount
private String languages [] = new String[3]
private JPanel contentPane2
private JTextField inp
private JLabel lbl1, lbl2, lbl3, lbl4, lbl5, lbl6, lbl7
private JTextArea textArea
private JButton button2
private JComboBox comboBox, comboBox2, comboBox3, comboBox4, comboBox5
private boolean b

These members are mostly for the GUI components while some are temporary variables used for storing information.

Constructors:

public Registration (boolean buyer)

“buyer” determines whether a buyer is getting registered or not. If not then seller must be getting registered.

Methods:

public void next ()
public void submit ()
public void submit2 ()
public void getClickValue ()
public void screen2()

next method is used to move to the next form in case of seller, in case of buyer it is replaced by submit which submits the buyer data after validating the entered information. getClickValue is used the receive the currently entered value in the table while screen2 sets the 2nd panel and initializes its data to display which is the second form.

IX. Request Class:

Request class is used for storing orders and request. Requests can be sent by buyers. It is like a sort of message in which a buyer can specify what they want, moreover, they can specify the price of the task. Once sent it is displayed to all active sellers who can view and accept the request. On acceptance it becomes an order which is completed when buyer transfers the amount. This class implements the Serializable interface as its objects will be written and read. This class is responsible for writing reading searching and modifying its objects.

Data members:

private String request
private double price
private Buyer sender
private Seller reciever
private Date created
private boolean accepted
private ArrayList<Request> requests

request holds the message of the request/order. Price holds the price. buyer holds the object of person who is sending the request. while receiver holds the person, who accepts which will initially be null. created stores the date on which request is sent. accepted determines whether request is active or not. and requests holds the objects of this class which is then written.

Constructors:

public Request ()
public Request (Buyer buyer, String request, double price)

First constructor initializes the array. 2nd constructor is used for creating a request.

Methods:

public void readRequest()
public void writeRequest()
public ArrayList<Request> searchActiveOrders(Buyer b)
public ArrayList<Request> searchActiveOrders(Seller s)
public ArrayList<Request> searchActiveRequests()
public void completeOrder(Buyer b, Seller s, String description)
public void updateRequest()
public void modifyRequest(Request o)
public Buyer getSender()
public Seller getSeller()
public double getPrice()
public Date getDate()
public void setSeller(Seller s)
public void setAccepted(boolean flag)
public String getMessage()
public void acceptRequest(Request r, Seller s)

These methods are responsible for writing/reading/modifying objects of this class.

X. Payments Class:

Payments class is used for managing payments. A buyer can deposit and withdraw money in/from his account. Moreover, a buyer can pay a seller with him they have an active order. Payment completes the order. On completion of order sent balance is deducted from the buyer and sent to the respective seller. Seller only has an option for withdrawing. This class extends JFrame and implements ActionListener.

Data members:

```
private Buyer buyer
private Seller seller
private JPanel contentPane, contentPane1, contentPane2
private JTextField textField, textField2
private JMenuBar menuBar
private JMenu menu
private JMenuItem i1, i2, i3, i4, i5, i6
private JButton button1, button2
private JButton button3
private JComboBox comboBox
private JLabel label2
private JLabel label4
private JLabel label5
private String[] descriptions
private JLabel labelHead
private JLabel ans
private ArrayList<Request> orders
private JLabel type
private JButton buttondone
```

Most data members hold the GUI components. orders is used for holding objects of Request class which are all active orders mostly. Description's array is used for holding descriptions of currently active orders.

Constructors:

```
public Payments (Person p)
```

Where p is the person to be loaded.

Methods:

```
public void screen0()
public void screen1()
public void screen2()
public void actionPerformed(ActionEvent e)
private boolean validateAmount(String amount)
```

First 3 methods are used to change the currently active screen. While actionListener manages click events.

XI. Orders Class:

Orders Class is used to display the currently active orders between buyers and sellers. This class also provides the buyer a facility to send requests which can be seen and accepted by every available seller. It extends JFrame and implements ActionListener.

Data members:

```
private JPanel contentPane
```

private JMenuBar menuBar
private JMenu menu
private JMenuItem i1, i2, i3, i4, i5, i6
private Buyer buyer
private Seller seller
private JLabel label1
private JLabel label2
private JLabel lblSelectOrder
private JComboBox comboBox
private JLabel label3
private JLabel label4
private JLabel label5
private JLabel label6
private JLabel label7
private JLabel ans1
private JLabel ans2
private JLabel ans3
private JLabel ans4
private JLabel ans5
private ArrayList<Request> rqsts
private JLabel label8
private JLabel ans6
private JButton button1
private JPanel contentPane2
private JTextField inp2
private JTextArea inp1
private JButton button2
private JLabel type

These members hold GUI components and rqsts arraylist holds the objects of request class.

Constructors:

public Orders (Person p)

where p is the person to be opened. All orders related to that person will be displayed.

Methods:

public void actionPerformed(ActionEvent e)
public void screen2()
private boolean validateAmount(String amount)

XII. ViewRequests Class:

This is a class exclusive to seller only in which a seller can view and accept the sent requests from buyer.

Data members:

private JPanel contentPane
private JMenuBar menuBar
private JMenu menu
private JMenuItem i1, i2, i3, i4, i5, i6
private Seller seller
private JLabel label1

private JLabel label2
private JLabel label3
private JComboBox comboBox
private JLabel label4
private JLabel label5
private JLabel label6
private JLabel label8
private JLabel ans1
private JLabel ans2
private JLabel ans3
private JLabel ans4
private ArrayList<Request> rqsts
private JLabel label7
private JButton button1
private JLabel type

These data members are used to store gui components while the rqsts arraylist holds the currently active requests.

Constructors:

public ViewRequests(Seller s)

where s is the seller to be loaded.

Methods:

public void actionPerformed(ActionEvent e)

XIII. Main Class:

Main Class holds the home screens of application. It contains the main method. Hence it is the starting point of the application. It provides the interface for the person to login into their account. It allows both sellers and buyers to login into their accounts. Through this class they can restore their lost password and, also navigate towards the registration section.

Data members:

private Person p
private JPanel contentPane
private JPanel contentPane2
private JTextField textField1
private JButton button1
private JButton loginButton
private JButton signupButton
private Buyer buyer
private Seller seller
private JPasswordField passwordField
private boolean b
private JLabel lblNewLabel
private JLabel type
private JButton forgotButton, button2
private JLabel label3, label2, label1

Constructors:

public Main (boolean b)

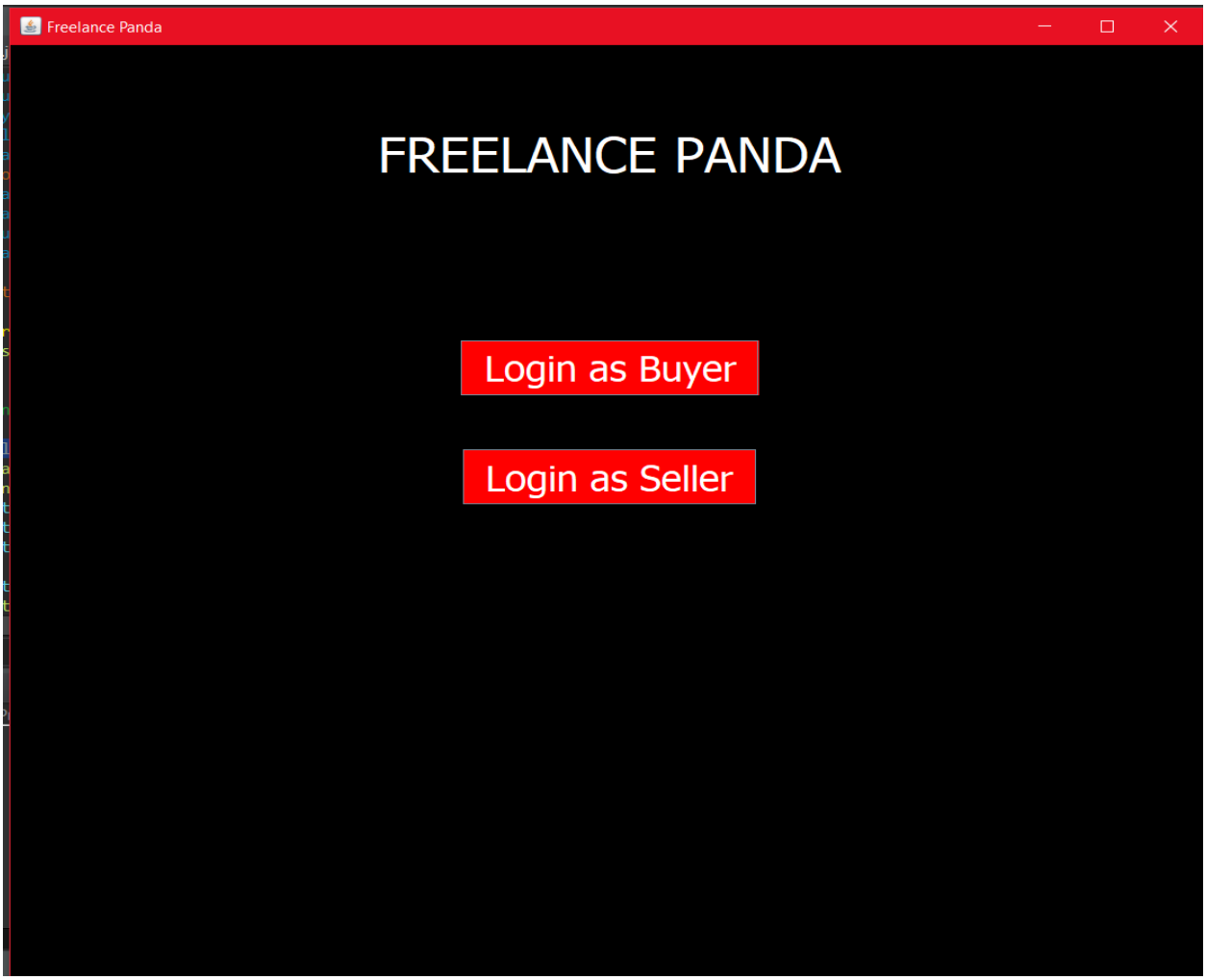
where b determines whether a buyer is about to login or not.

Methods:

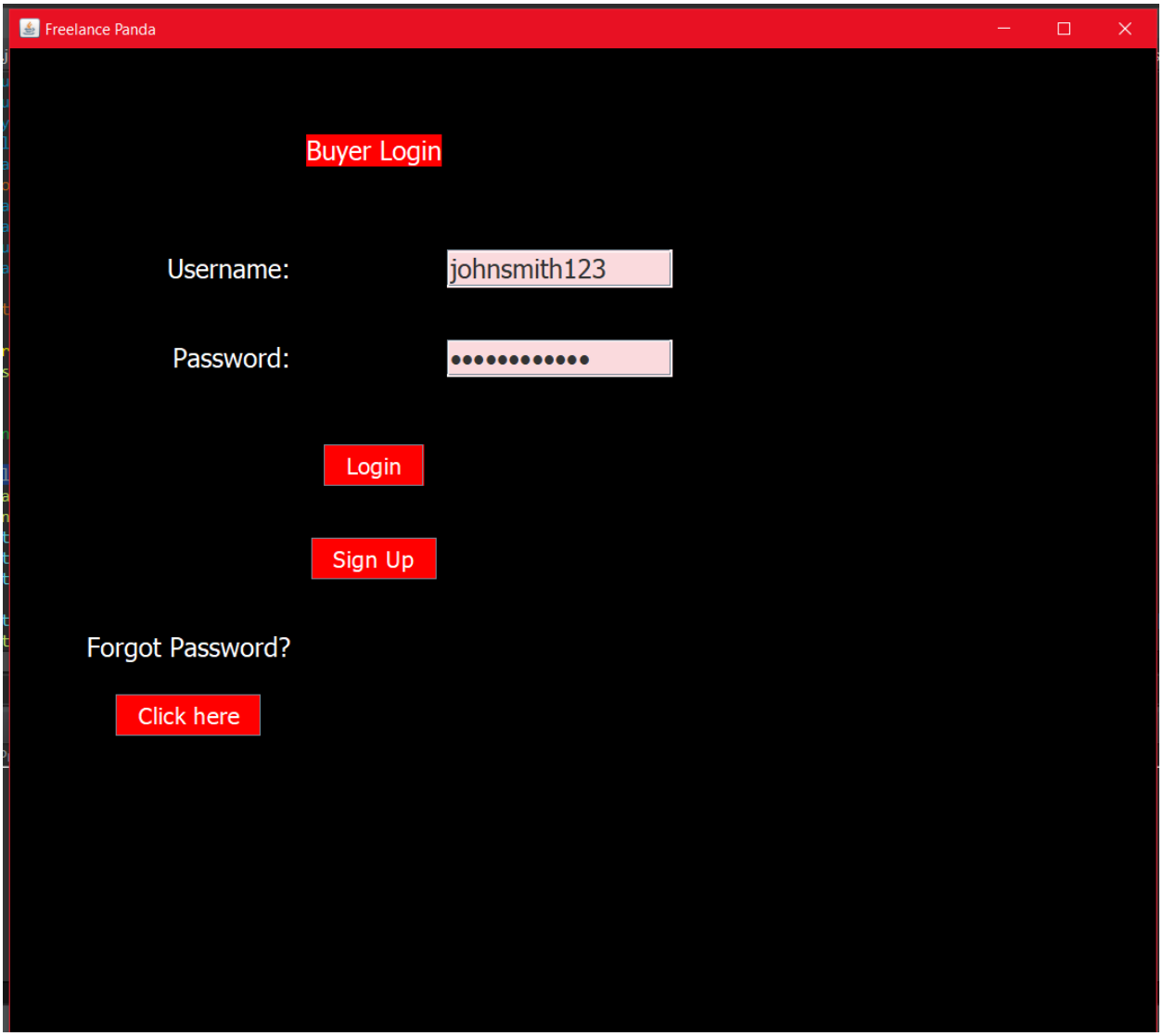
public static void main (String [] args)
public void actionPerformed(ActionEvent e)
public void screen2()
public void setScreen2()

where main is the main method while others are for events and switching screens.

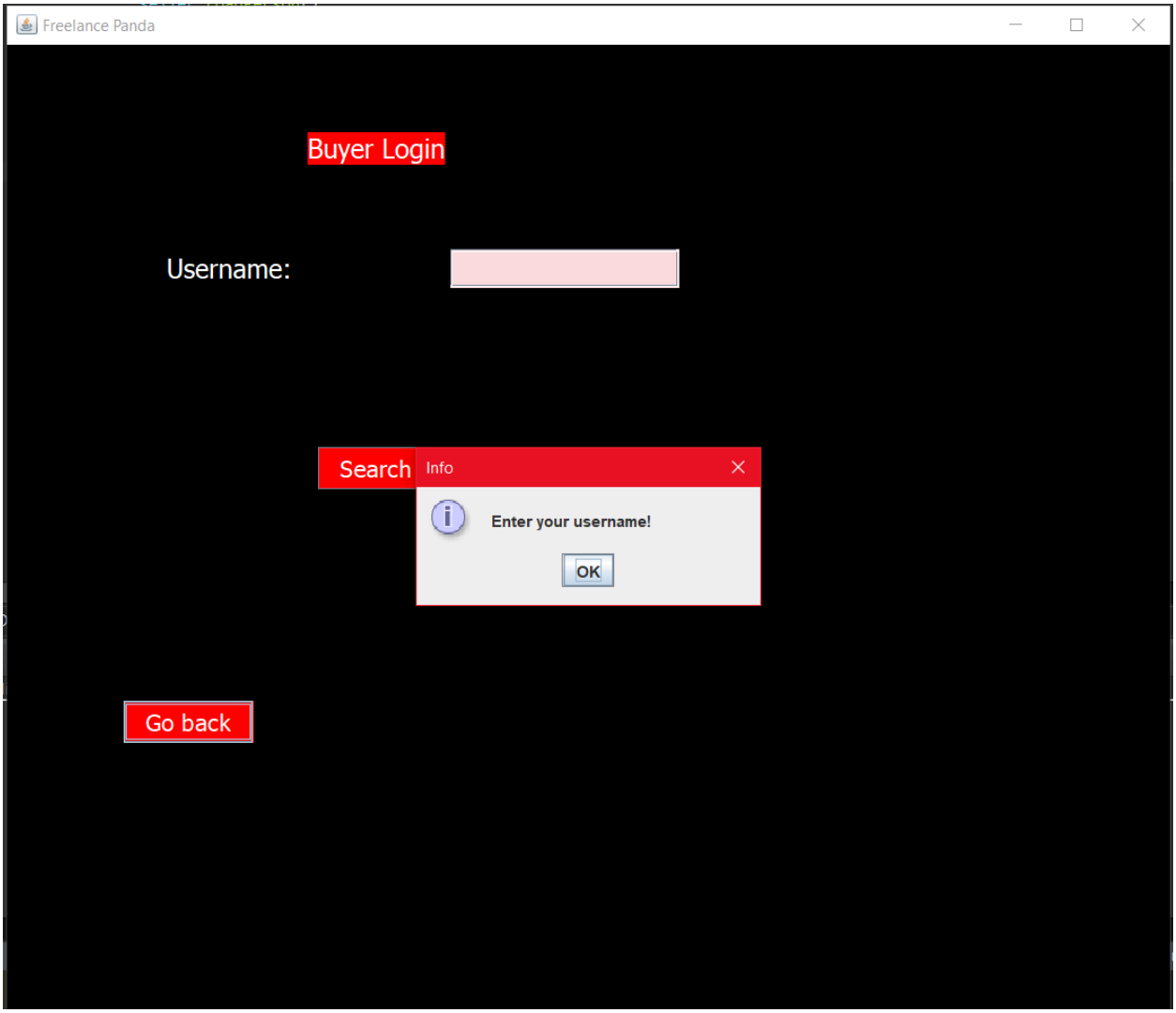
SCREENSHOTS:



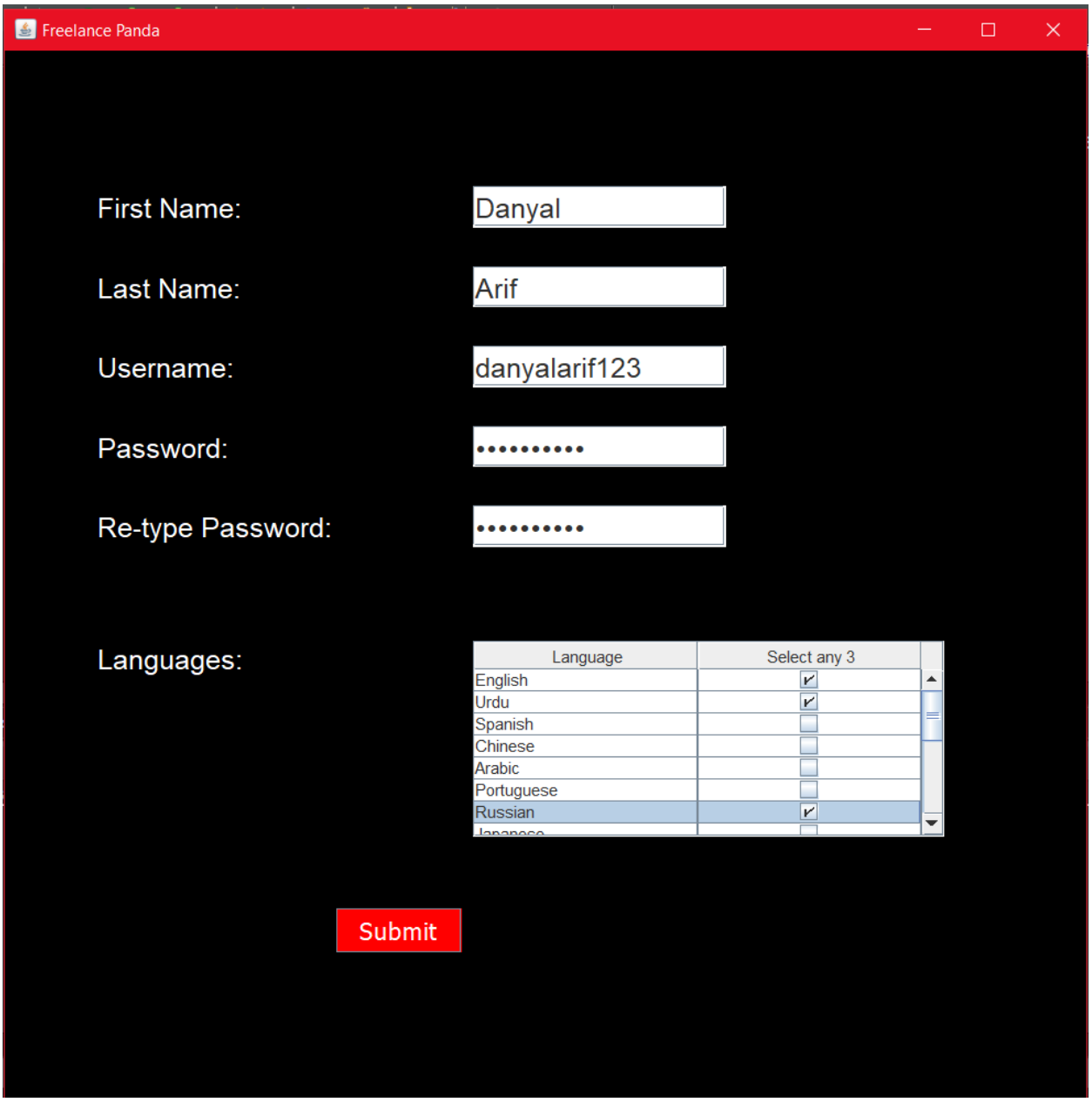
Home Screen.



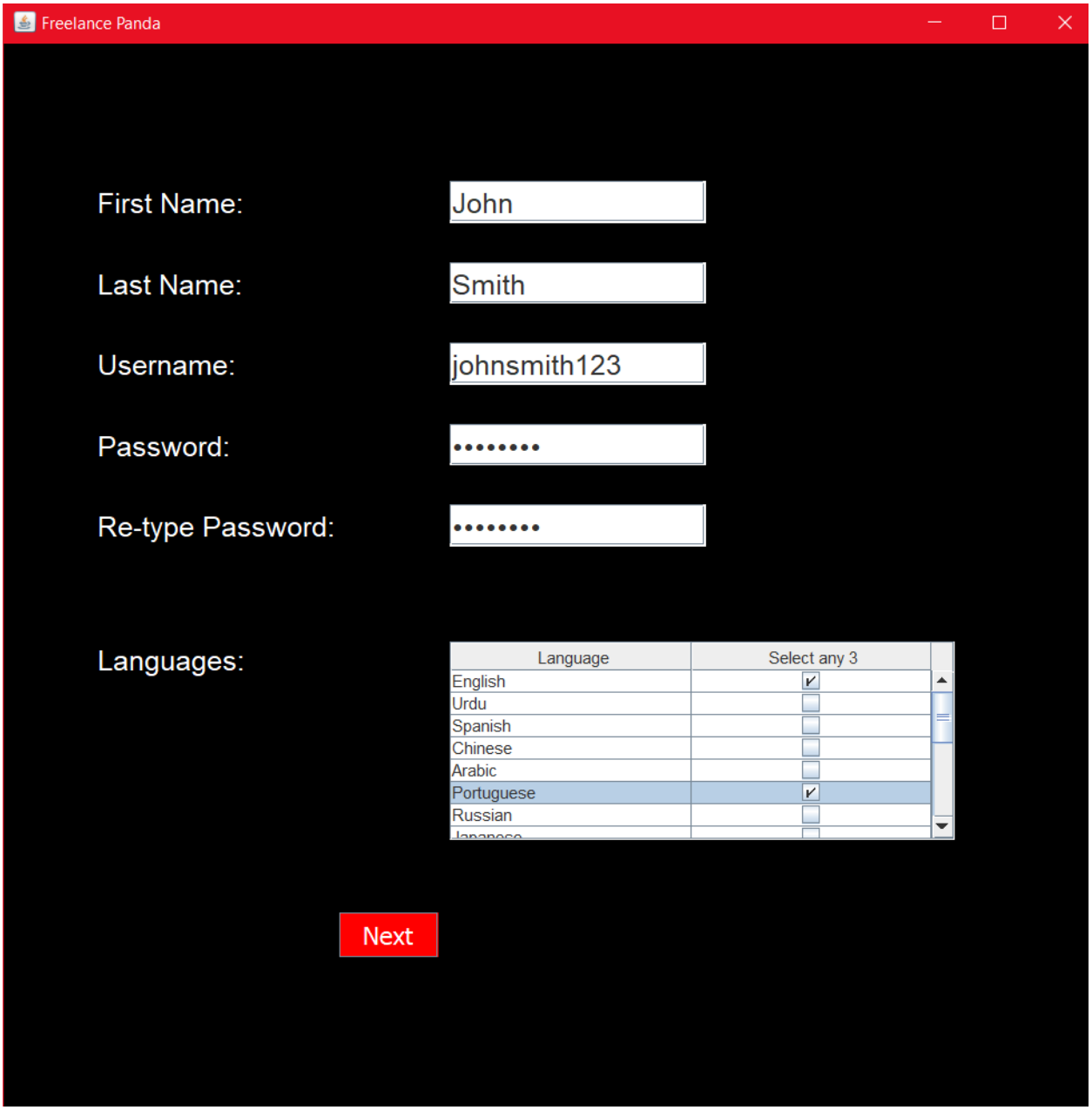
Login Screen.



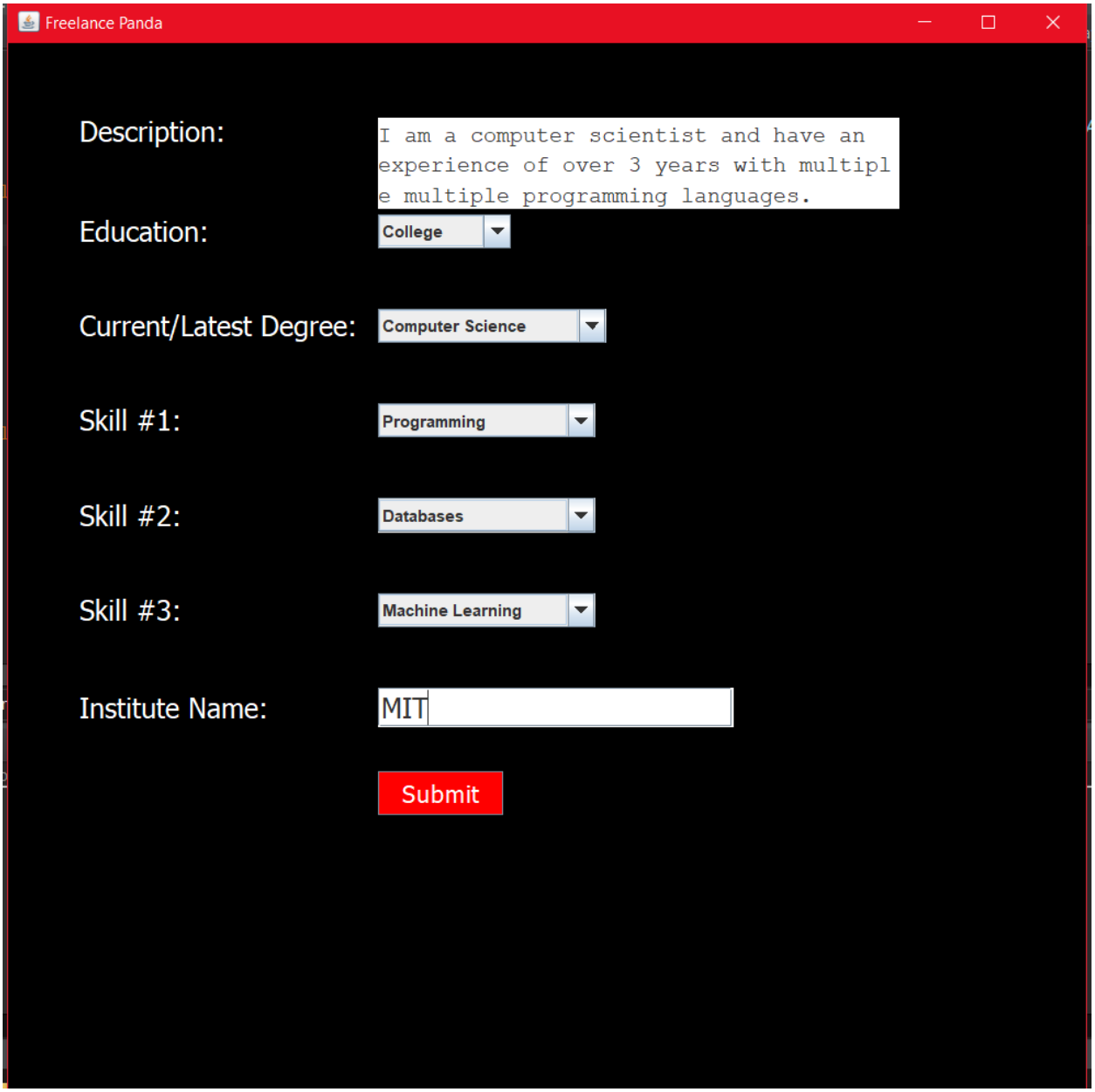
Forget Password Screen.



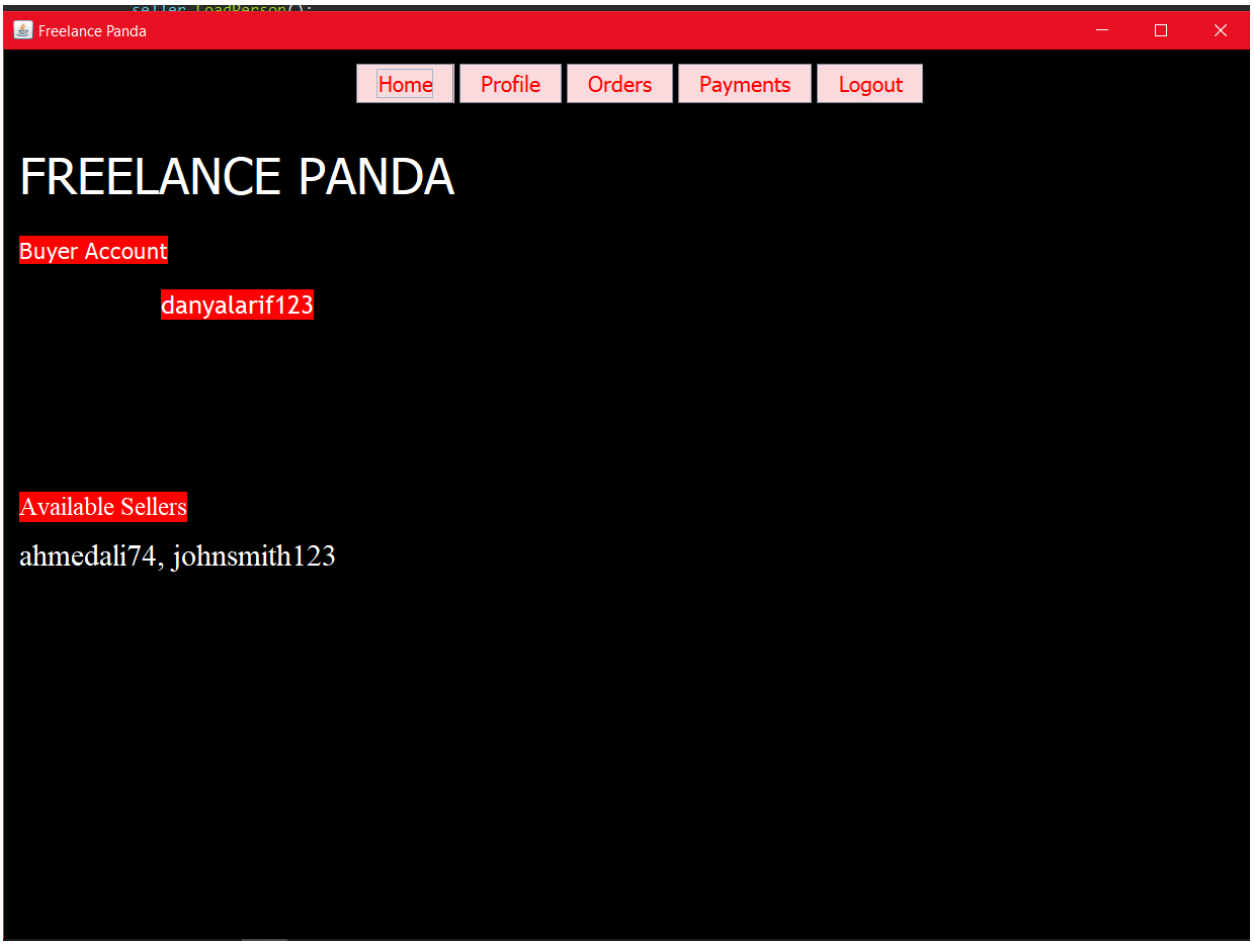
Buyer Registration.



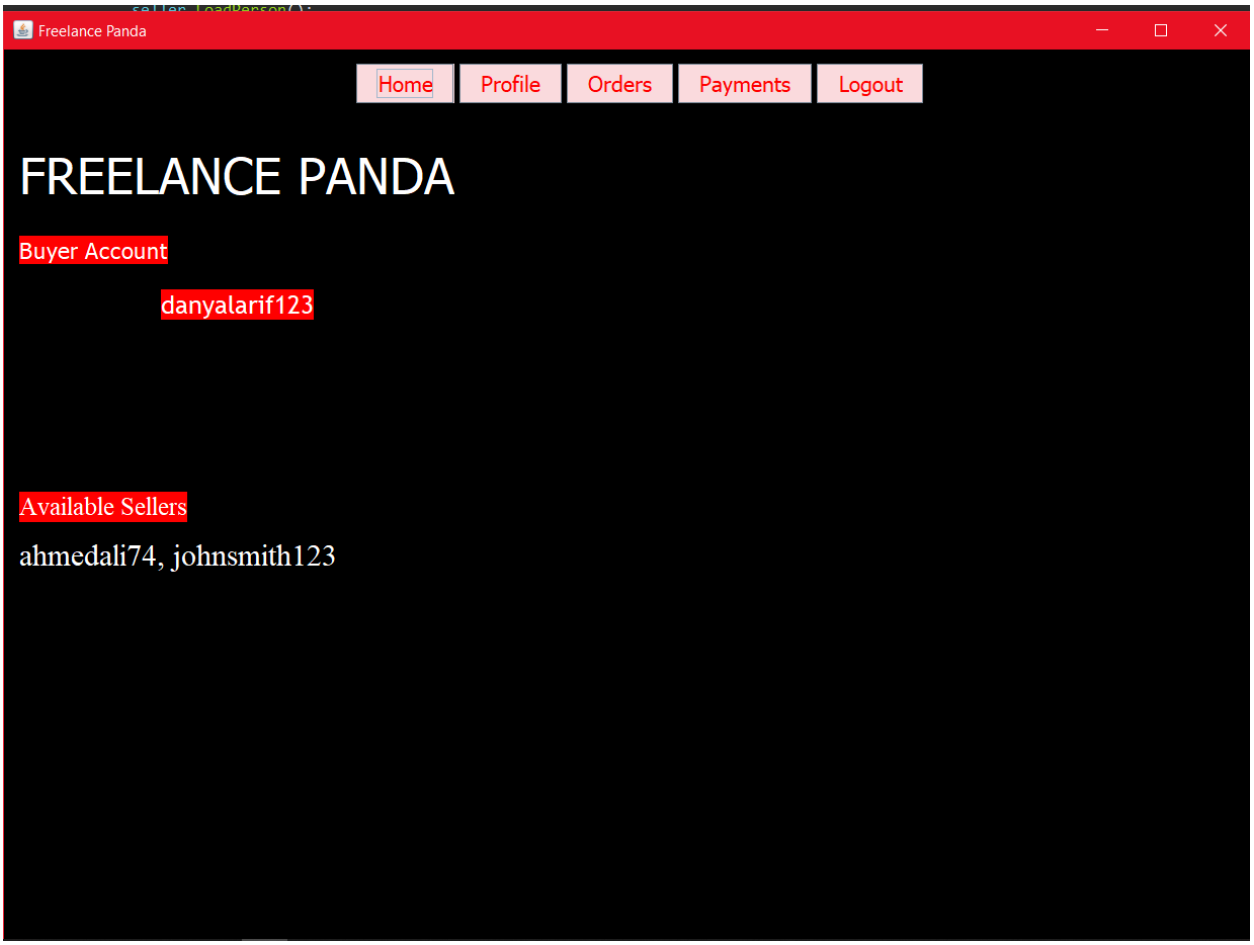
Seller Registration (part 1)



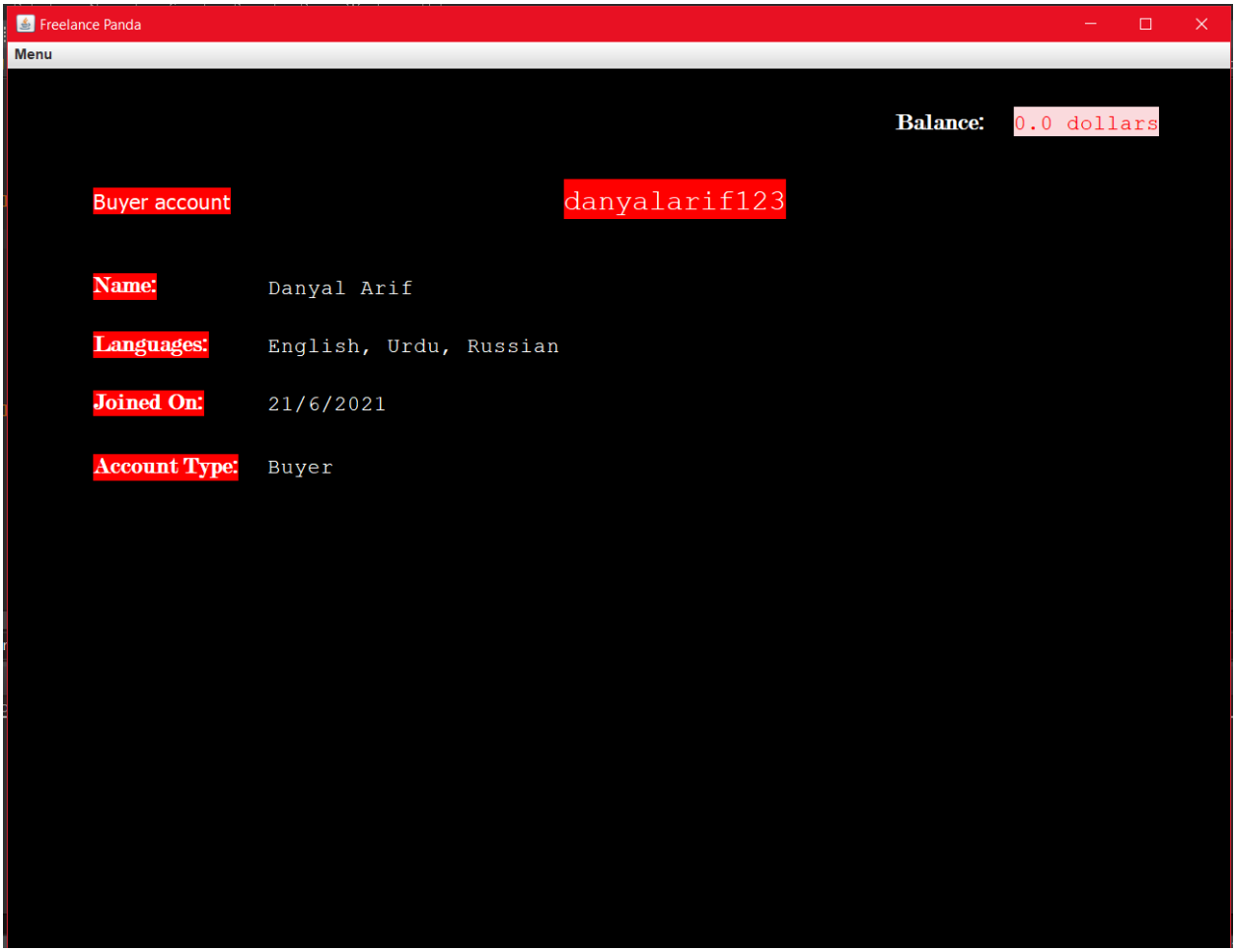
Seller Registration (part 2)



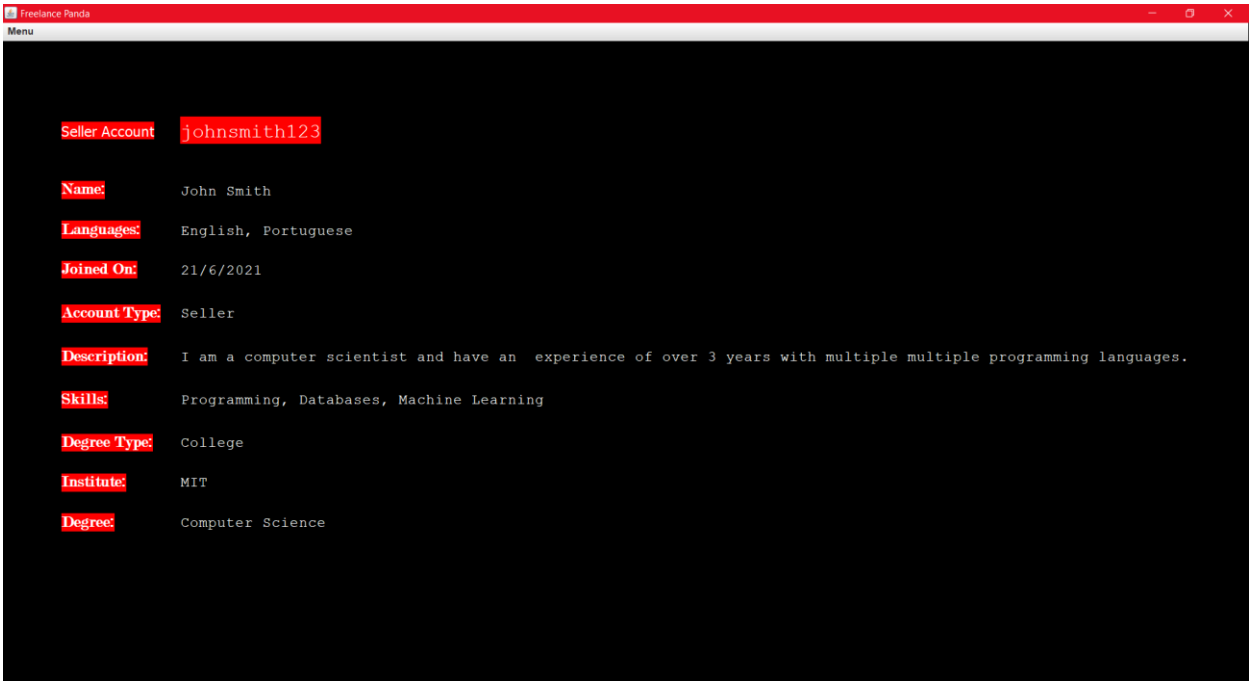
Home Screen (for Buyer).



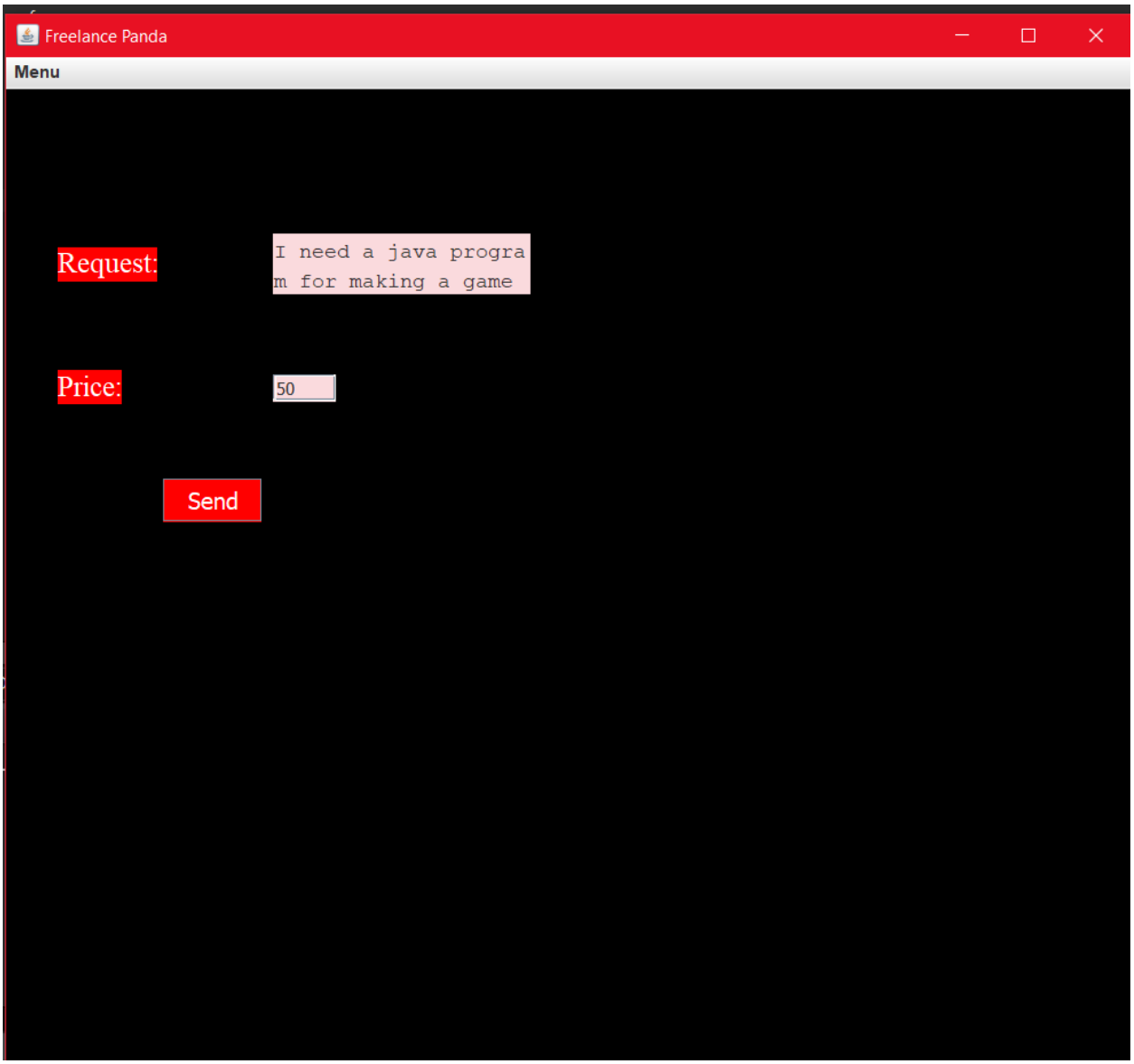
Home Screen (for Seller).



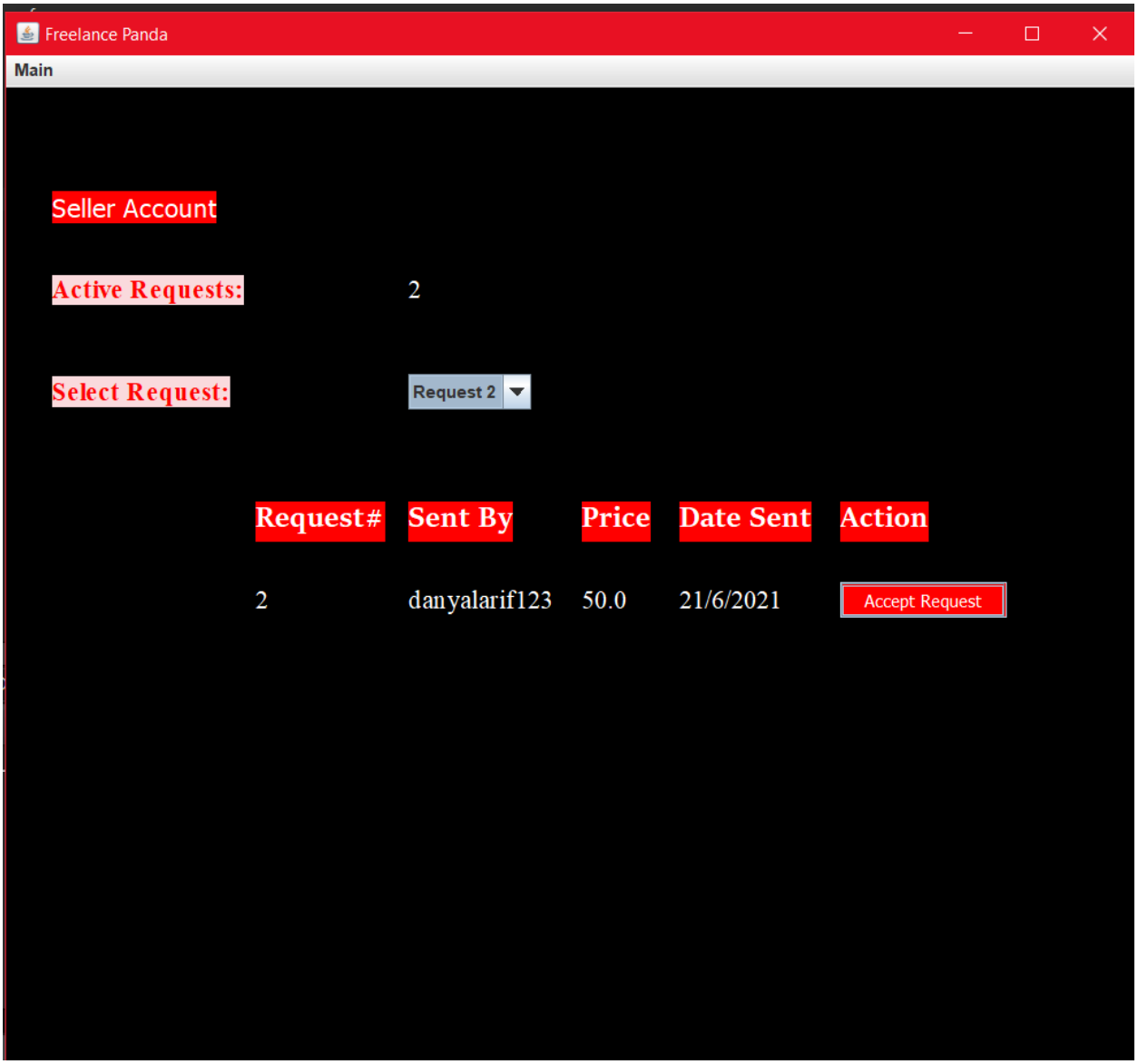
Profile (for Buyer).



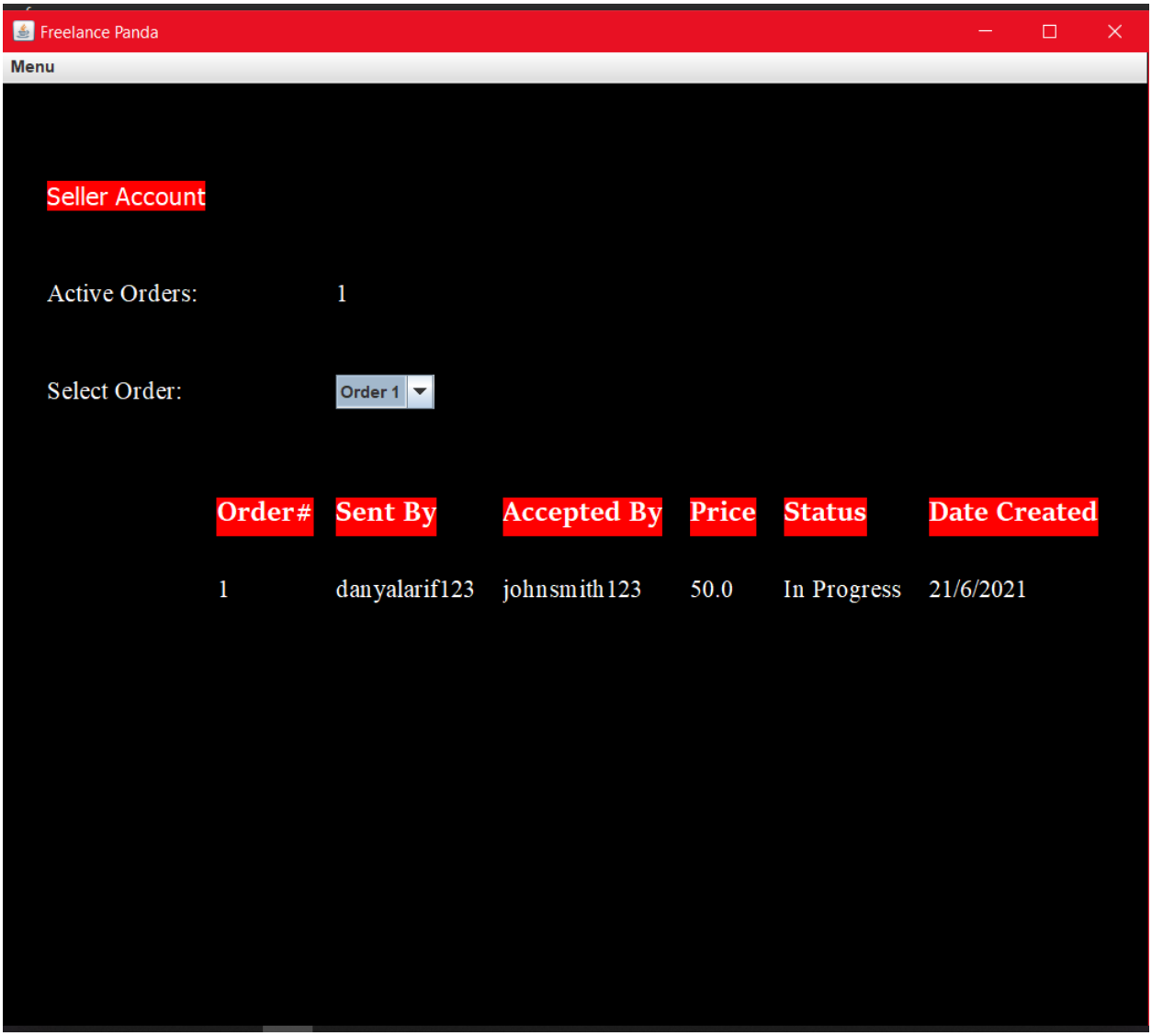
Profile (for Seller).



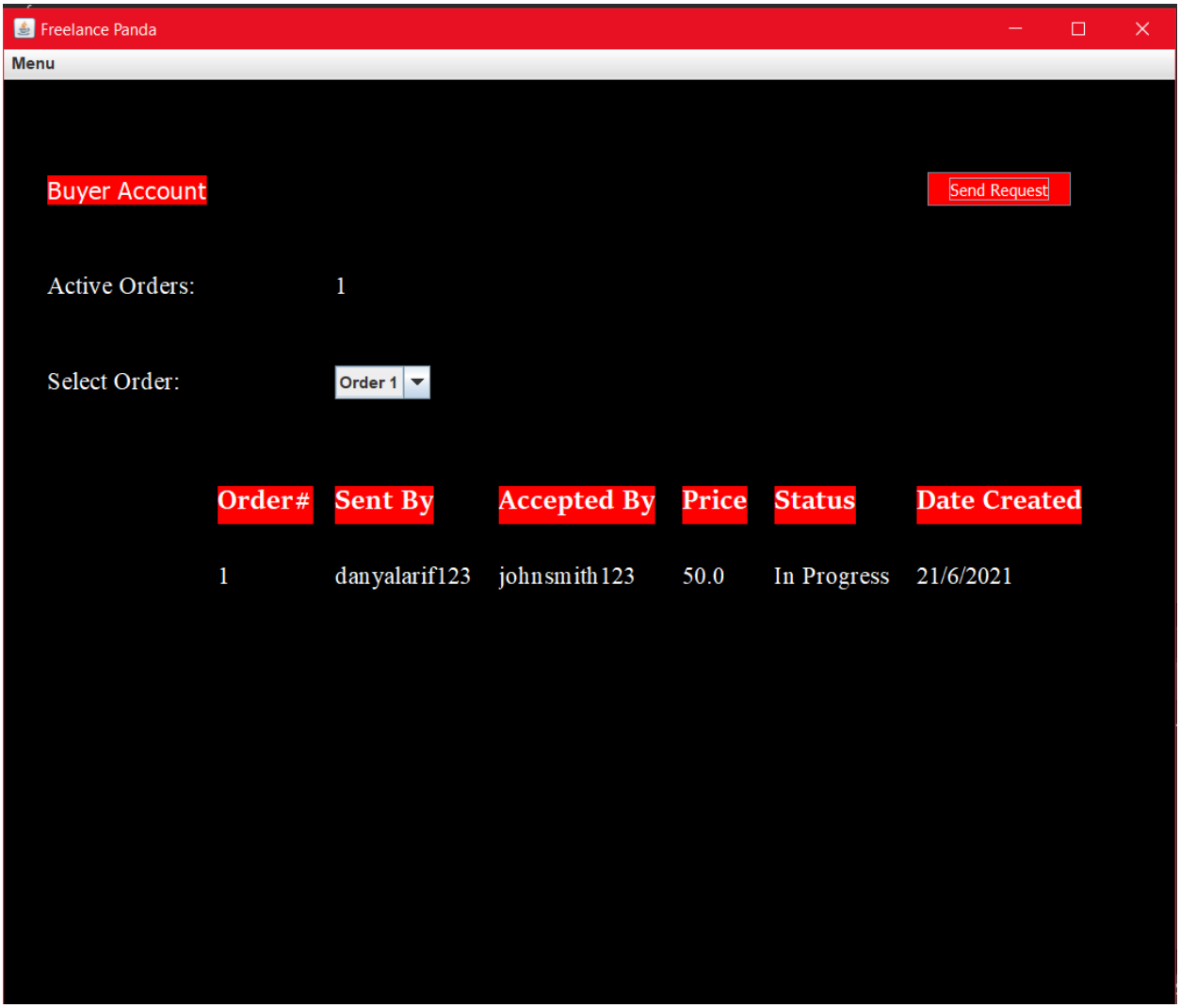
Buyer, Sending Request.



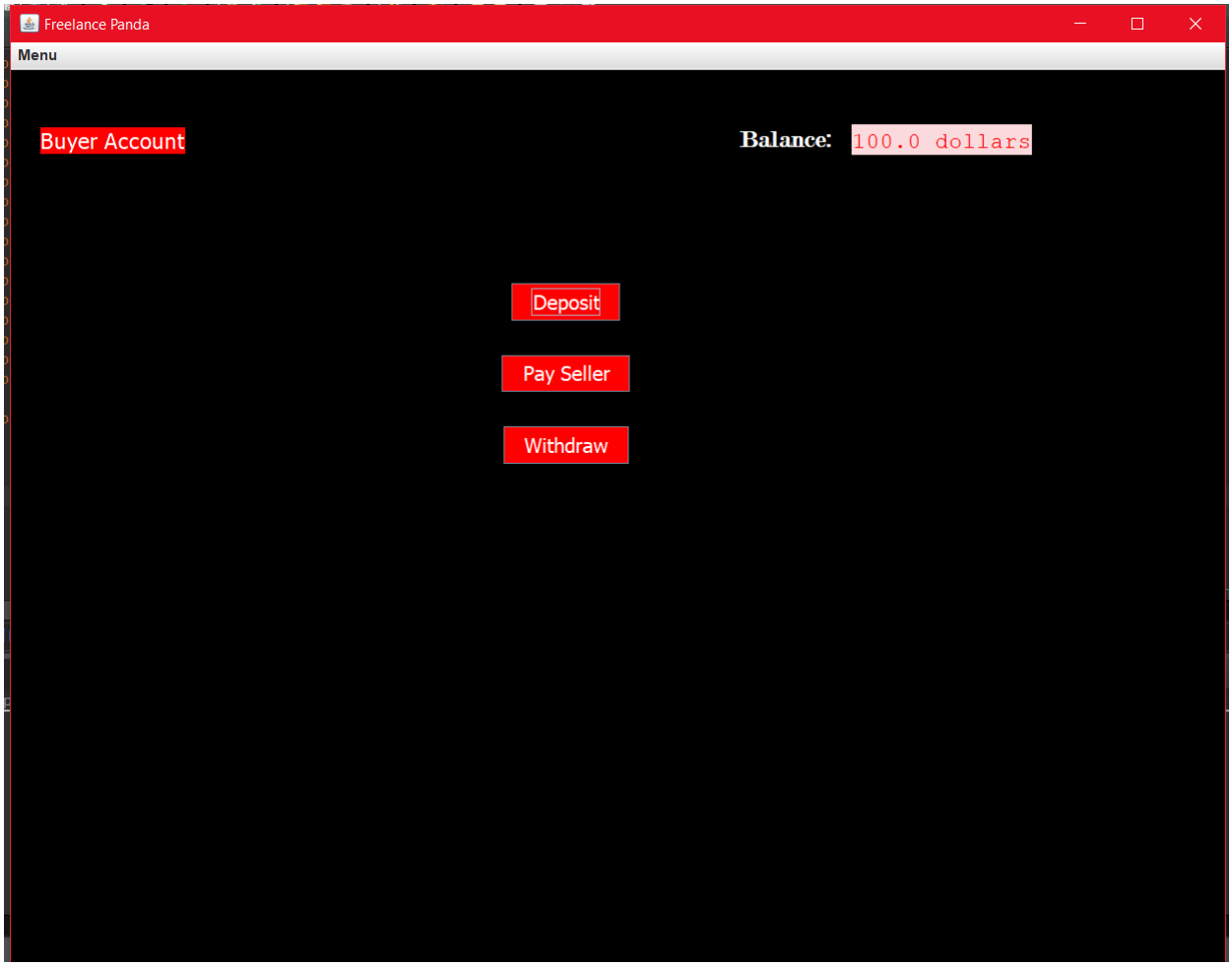
Requests Screen.



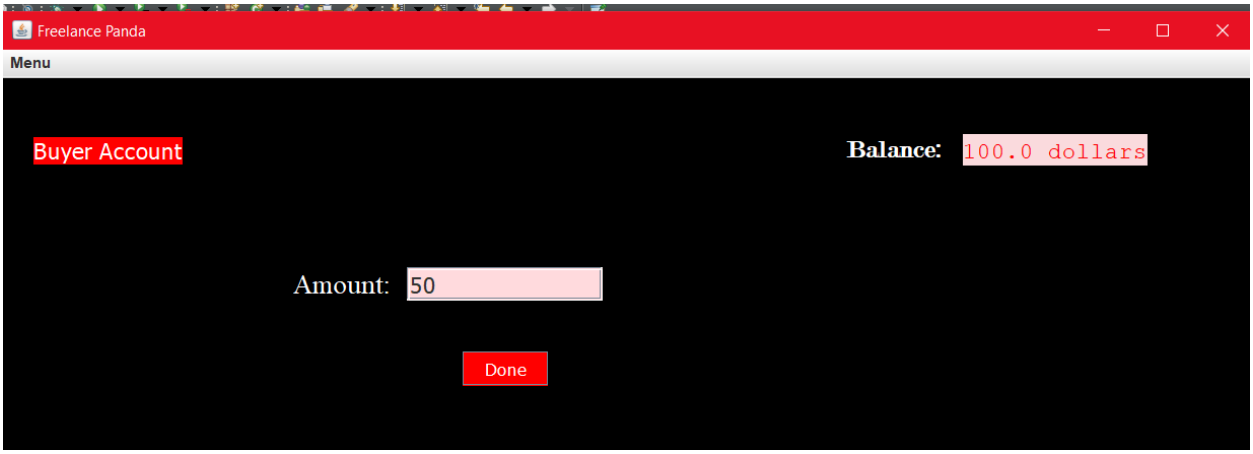
Active Orders (for Seller).



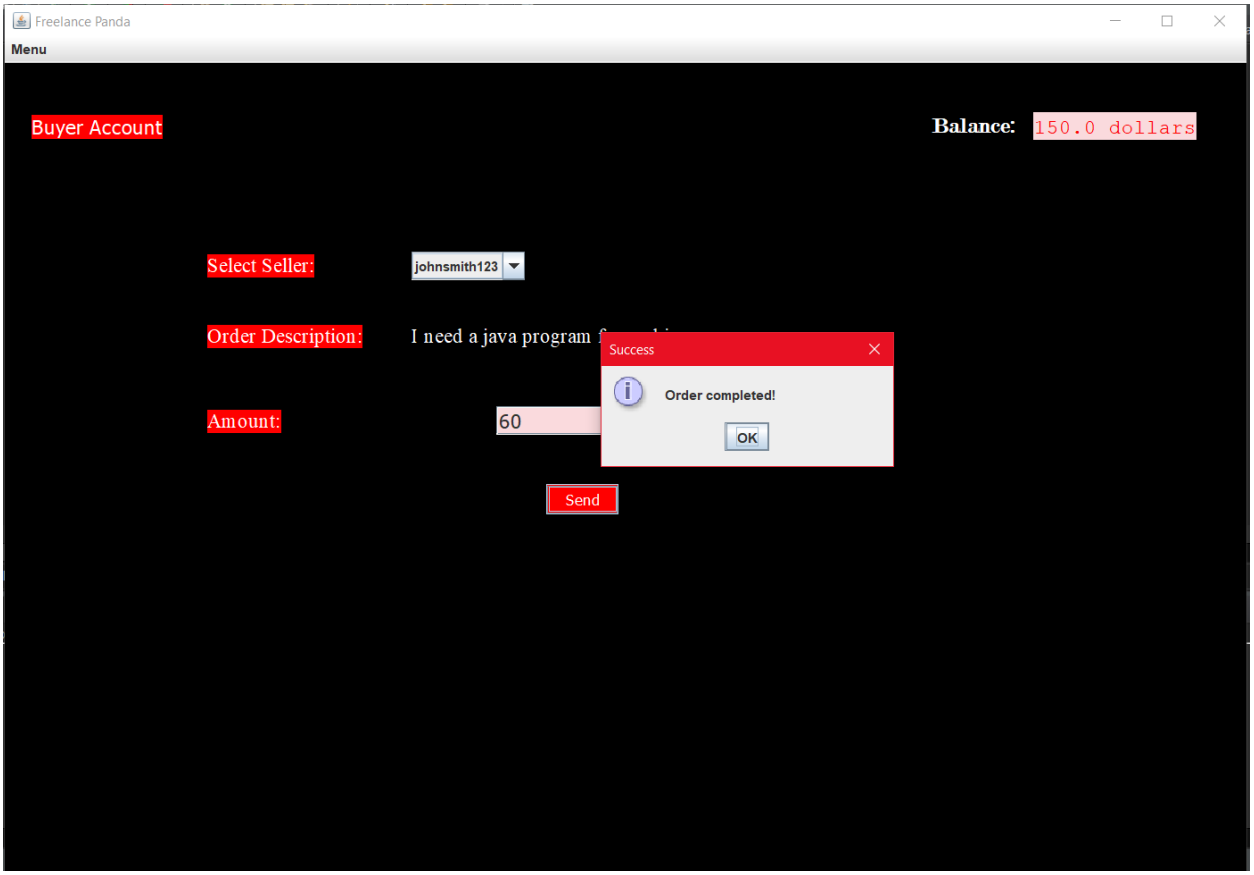
Active Orders (for Buyer).



Payments (for Buyer).



Buyer, Depositing Money.



Buyer, Paying Seller.

Updated Balance, Menu after Transaction.