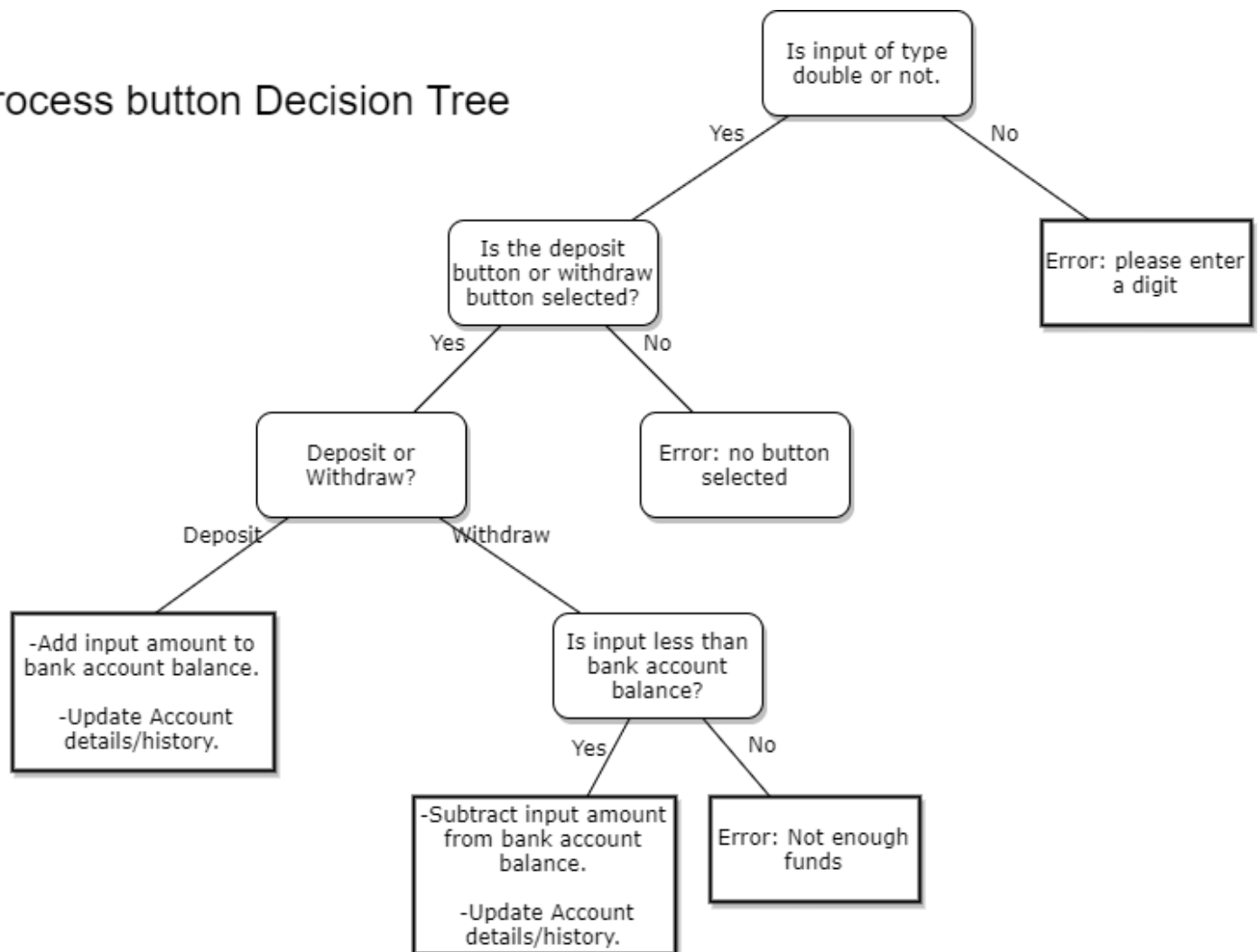


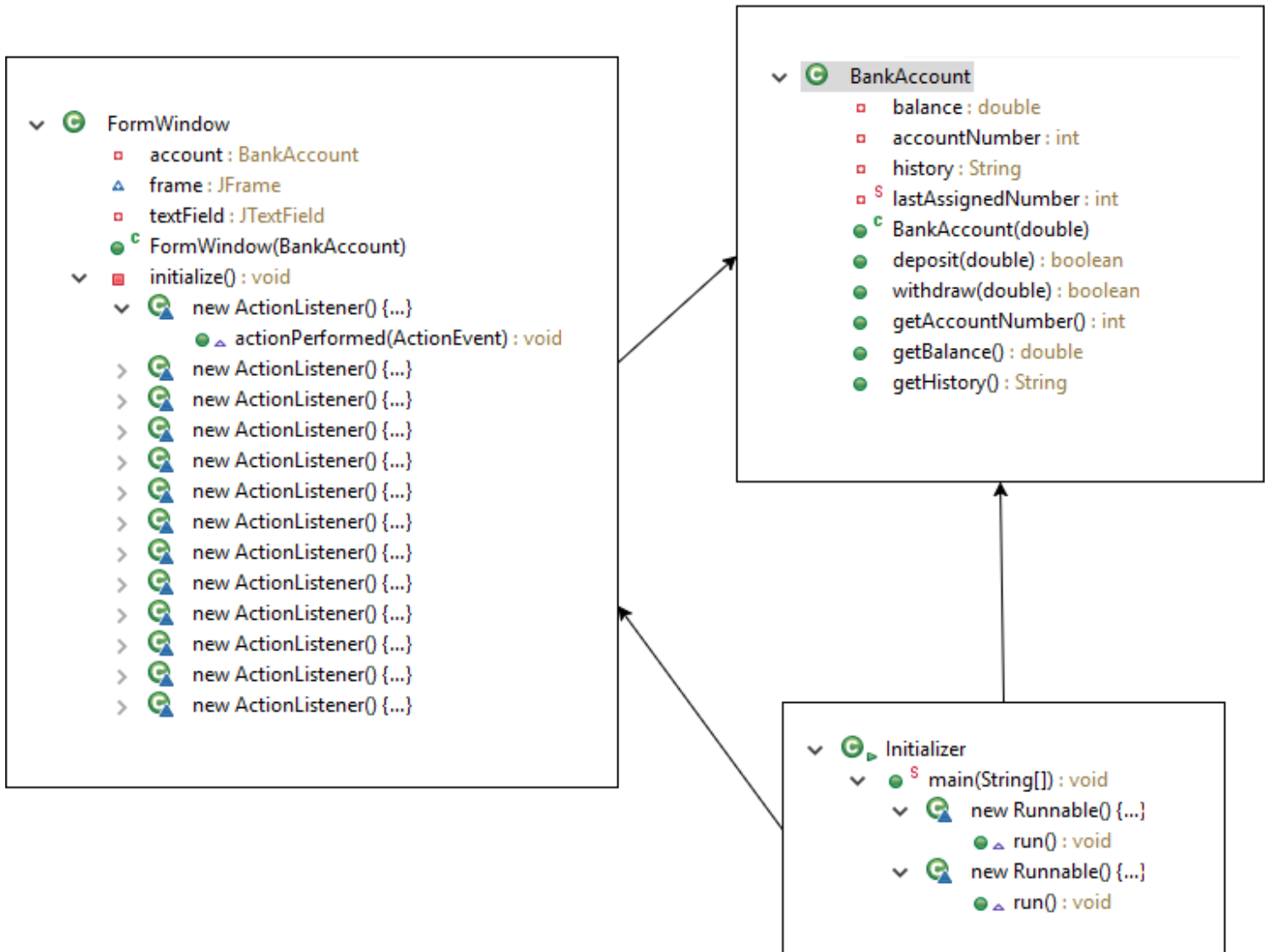
Unit I Project

Faye Lopez








Process button Decision Tree



UML



Source code attached.

- ▼  BankAccountGUI
 - ▼  src
 - ▼  (default package)
 - >  BankAccount.java
 - >  FormWindow.java
 - >  Initializer.java
 - >  JRE System Library [JavaSE

Program Output:

Keith's Bank Account Interface

Account Details
 Account number: 1
 Balance: \$70007000002440.73

History

-Deposited \$8.00.	Balance: \$1395.00
-Deposited \$8.00.	Balance: \$1403.00
-Deposited \$88.00.	Balance: \$1491.00
-Deposited \$8.00.	Balance: \$1499.00
-Deposited \$888.00.	Balance: \$2387.00
-Deposited \$88.00.	Balance: \$2475.00
-Withdrawal \$8.00.	Balance: \$2467.00
-Withdrawal \$8.00.	Balance: \$2459.00
-Withdrawal \$8.00.	Balance: \$2451.00
-Withdrawal \$8.00.	Balance: \$2443.00
-Withdrawal \$2.27.	Balance: \$2440.74
-Deposited \$7000000000.00.	Balance: \$7000002440.73
-Deposited \$70000000000000.00.	Balance: \$70007000002440.73

Amount:

☒ Deposit
☐ Withdraw

Process

7	8	9
4	5	6
1	2	3
0	.	Clear

Test case 1: Try to deposit a negative number.

Result: Program deposits negative funds, logic error.

The screenshot shows a web application titled "Keith's Bank Account Interface". It features two main panels on the left: "Account Details" and "History". The "Account Details" panel shows "Account number: 1" and "Balance: \$-97729.00". The "History" panel shows three transactions: "-Deposited \$-54524.0. Balance: \$-54519.00", "-Deposited \$2424.0. Balance: \$-52095.00", and "-Deposited \$-45634.0. Balance: \$-97729.00". On the right, there is an "Amount:" input field, a "Process" button, and two radio buttons: "Deposit" (selected) and "Withdraw". Below these is a numeric keypad with buttons for digits 0-9, a decimal point, and a "Clear" button.

Keith's Bank Account Interface														
Account Details Account number: 1 Balance: \$-97729.00	Amount: <input type="text"/> <input type="button" value="Process"/>	<input checked="" type="radio"/> Deposit <input type="radio"/> Withdraw												
History -Deposited \$-54524.0. Balance: \$-54519.00 -Deposited \$2424.0. Balance: \$-52095.00 -Deposited \$-45634.0. Balance: \$-97729.00	<table border="1"><tr><td>7</td><td>8</td><td>9</td></tr><tr><td>4</td><td>5</td><td>6</td></tr><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>0</td><td>.</td><td>Clear</td></tr></table>		7	8	9	4	5	6	1	2	3	0	.	Clear
7	8	9												
4	5	6												
1	2	3												
0	.	Clear												

Solution: To fix this, I added two if-statements.

- **First if-statement:** Allows you to deposit only if your input is greater than 0.
- **Second if statement:** Does not deposit if input is less than or equal to 0 and prints error msg.


```

else if (rdbtnDeposit.isSelected()) { //deposit button

    try {
        double amount = Double.parseDouble(textField.getText());
        if (amount > 0) {
            account.deposit(amount);
            textField.setText("");
            listModel.setElementAt("Balance: " + "$" + dFormat.format(account.getBalance()), 1);
            if (account.getBalance() == 0) {
                listModel2.addElement("-Deposited " + "$" + amount + "." + "    Balance: $" + "0.00");
            } else {
                listModel2.addElement("-Deposited " + "$" + amount + "." + "    Balance: $" + dFormat.format(account.getBalance()));
            }
            error.setText("");
        }
        if (amount <= 0) {
            error.setText("Please enter a number greater than 0.");
            textField.setText("");
        }
    } catch (NumberFormatException e1) { //error if input is not of type double
        error.setText("Error: Please only enter a number..");
    }

    if (account.getBalance() == 0) { //sets 'account details' balance
        listModel.setElementAt("Balance: " + "$" + "0.00", 1);
        // listModel2.addElement("-Deposited " + "$" + amount + "." + "    Balance: $" + dFormat.format(keith.getBalance()));
    }
}
}

```


— □ ×

Keith's Bank Account Interface

Account Details

Account number: 1
Balance: \$5.0

History

Please enter a number greater than 0.

Amount:

☒ Deposit
 ☐ Withdraw

7	8	9
4	5	6
1	2	3
0	.	Clear

Test Case 2: Input a very large digit.

Result: Double numbers prints out in scientific notation on left hand side of history for deposit. Prints all digits on right hand side of history for balance.

The screenshot shows a Java Swing window titled "Keith's Bank Account Interface". It contains three main sections:

- Account Details:** Displays "Account number: 1" and "Balance: \$110000000.00".
- History:** A list box showing three entries:
 - Withdrawal \$5.0 Balance: \$0.00
 - Deposited \$1.0E7 Balance: \$10000000.00
 - Deposited \$1.0E8 Balance: \$110000000.00
- Transaction Section:** Includes an "Amount:" input field, a "Process" button, and two radio buttons: "Deposit" (selected) and "Withdraw".
- Numeric Keypad:** A 4x3 grid of buttons containing digits 0-9, a decimal point ".", and a "Clear" button.

```
listModel.setElementAt("Balance: " + "$" + dFormat.format(account.getBalance()), 1);
if (account.getBalance() == 0) {
    listModel2.addElement("-Deposited " + "$" + amount + "." + "      Balance: $" + "0.00");
} else {
    listModel2.addElement("-Deposited " + "$" + amount + "." + "      Balance: $" + dFormat.format(account.getBalance()));
}
error.setText("");
if(amount <= 0) {
    error.setText("Please enter a number greater than 0.");
}
```

A red arrow points from the `dFormat` object in the second line of the `else` block to the `dFormat` object in the first line, highlighting the inconsistency in formatting.

Solution: We can use `decimalFormat` on the *amount* processed like we did with *account.getBalance*.


```

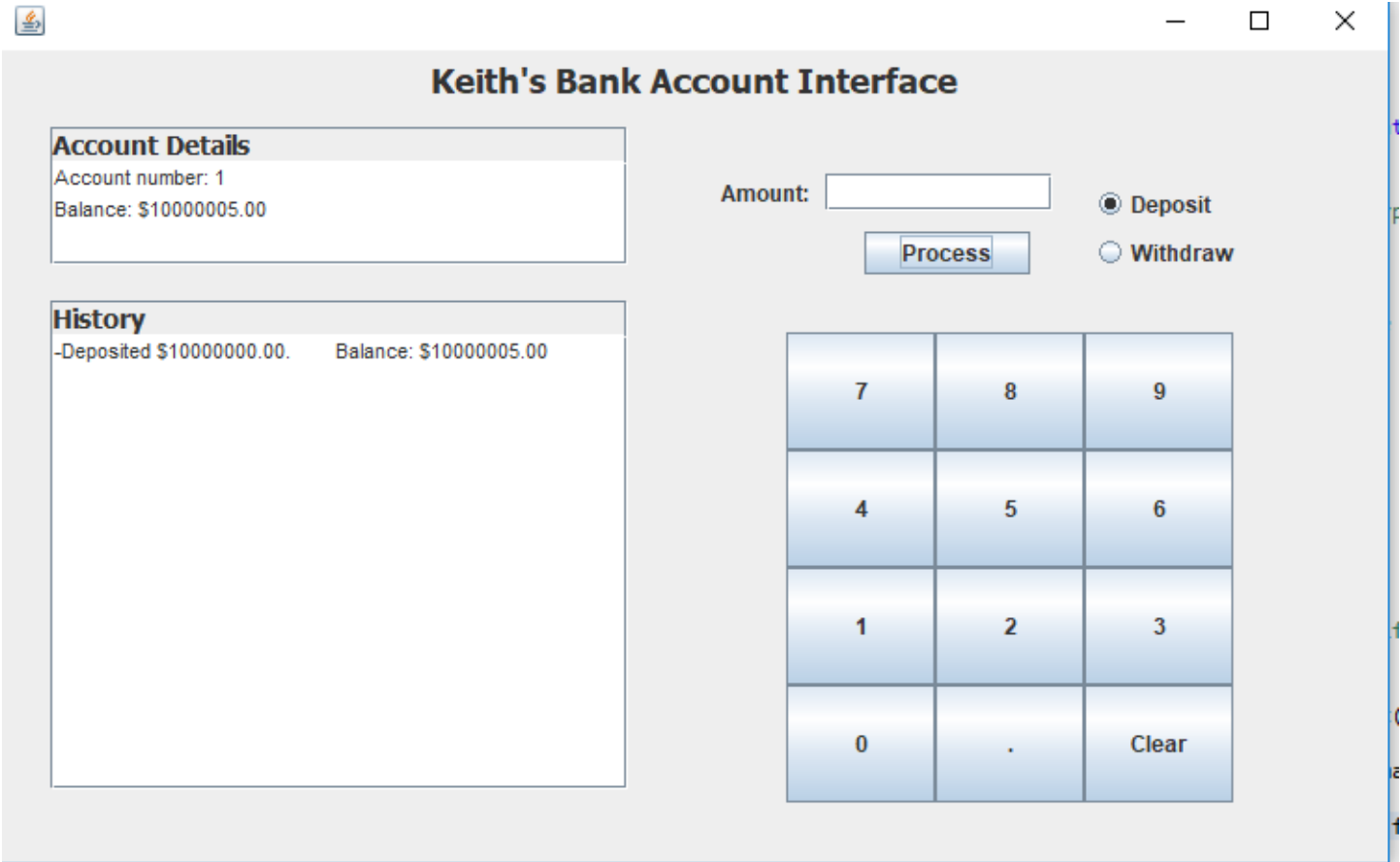
listModel.setElementAt("Balance: " + "$" + dFormat.format(account.getBalance()), 1);
if (account.getBalance() == 0) {
    listModel2.addElement("-Deposited " + "$" + dFormat.format(amount) + "." + "
} else {
    listModel2.addElement("-Deposited " + "$" + dFormat.format(amount) + "." + "
}
error.setText("");

if(amount <= 0) {
    error.setText("Please enter a number greater than 0.");
}

```

Balance: \$" + "0.00");

Balance: \$" + dFormat.format(account.get



Keith's Bank Account Interface

Account Details

Account number: 1
Balance: \$10000005.00

History

-Deposited \$10000000.00. Balance: \$10000005.00

Amount:

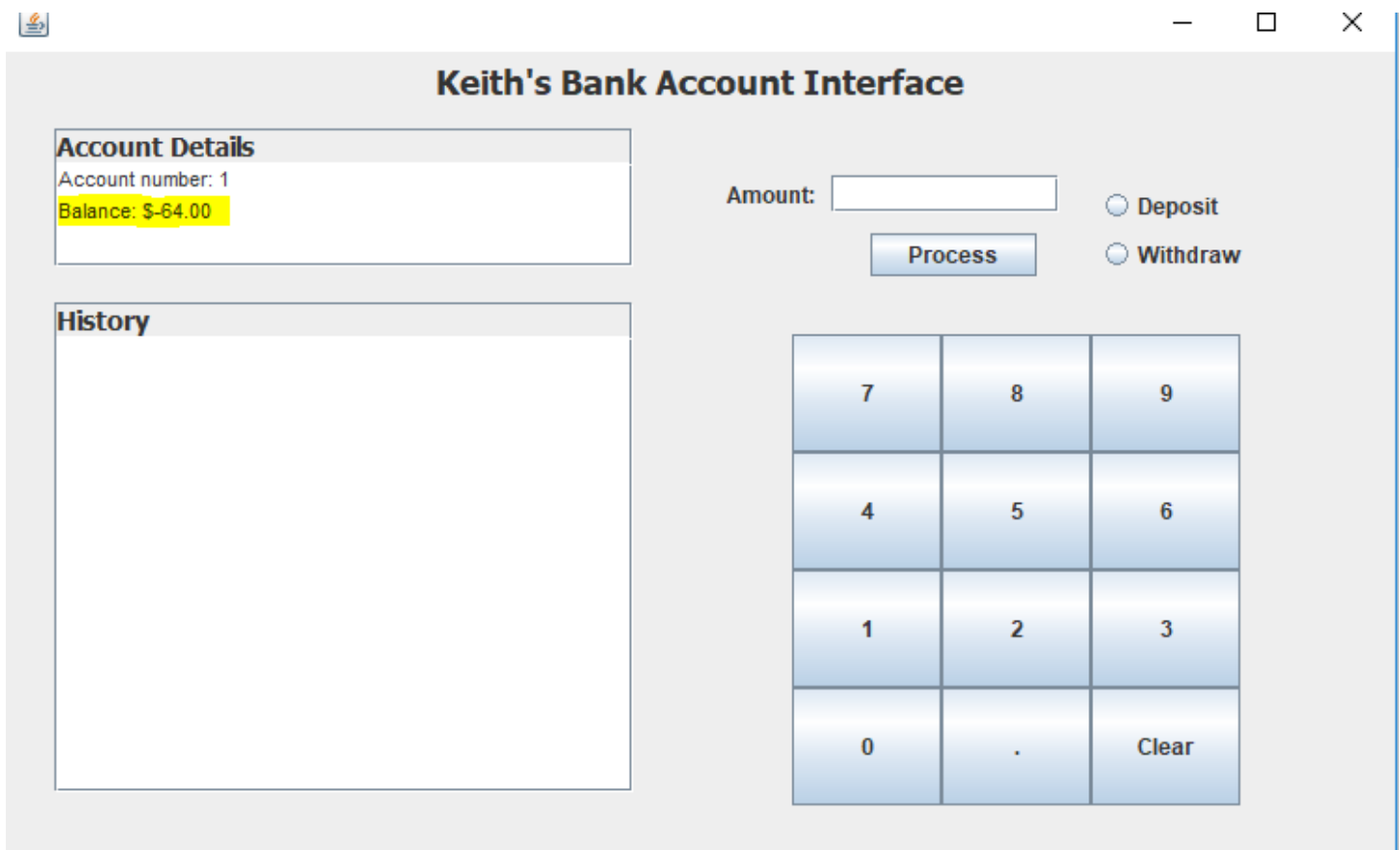
☒ Deposit
☐ Withdraw

7	8	9
4	5	6
1	2	3
0	.	Clear

Test Case 3: Initialize a new *bankaccount* object with a negative double.

Result: The bank account is created with a negative balance.

```
public class Initializer {  
    public static void main(String[] args) {  
  
        BankAccount keith = new BankAccount(-64);  
  
        EventQueue.invokeLater(new Runnable() {  
            public void run() {  
                try {  
                    FormWindow window = new FormWindow(keith);  
                    window.frame.setVisible(true);  
                } catch (Exception e) {  
                    e.printStackTrace();  
                }  
            }  
        });  
    }  
}
```



The image shows a Java Swing window titled "Keith's Bank Account Interface". The window has a light gray background and standard window controls (minimize, maximize, close) in the top right corner. On the left side, there is a panel with two sections: "Account Details" and "History". The "Account Details" section shows "Account number: 1" and "Balance: \$-64.00". The "History" section is an empty text area. On the right side, there is a form for transactions. It includes a label "Amount:" followed by a text input field. Below the input field is a "Process" button. To the right of the input field are two radio buttons: "Deposit" and "Withdraw". Below the form is a numeric keypad with buttons for digits 0-9, a decimal point, and a "Clear" button.

7	8	9
4	5	6
1	2	3
0	.	Clear

Solution: Add an if and else statement in the *bankaccount* constructor to determine if initial deposit is greater than 0. If it is not greater than 0, then the program will automatically set the balance to \$0.00.

```
public BankAccount(double balance) {  
    accountNumber = lastAssignedNumber++;  
    if (balance < 0) {  
        balance = this.balance;}  
    else {  
        this.balance = balance;  
    }  
}
```