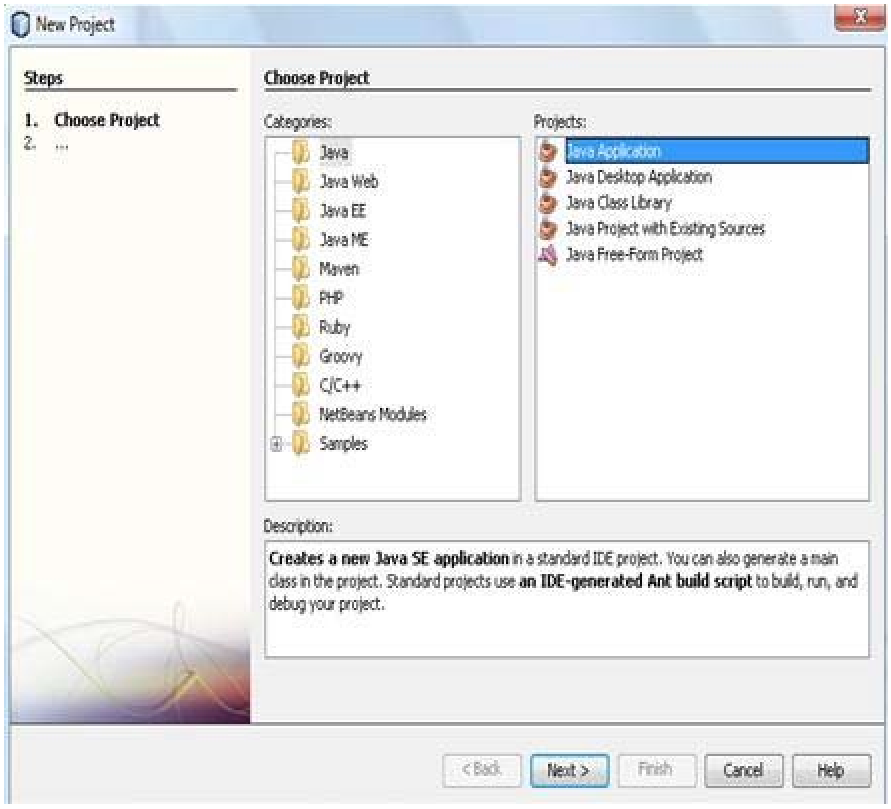
Debugging in NetBeans (Part II)

As we have already learned the basics of debugging in our previous classes, here we will focus on some more features of NetBeans' debugger like **Step Into**, **Step Out**, etc. Just have a look at our first debugging tutorial if you are unfamiliar with the basics of debugging.

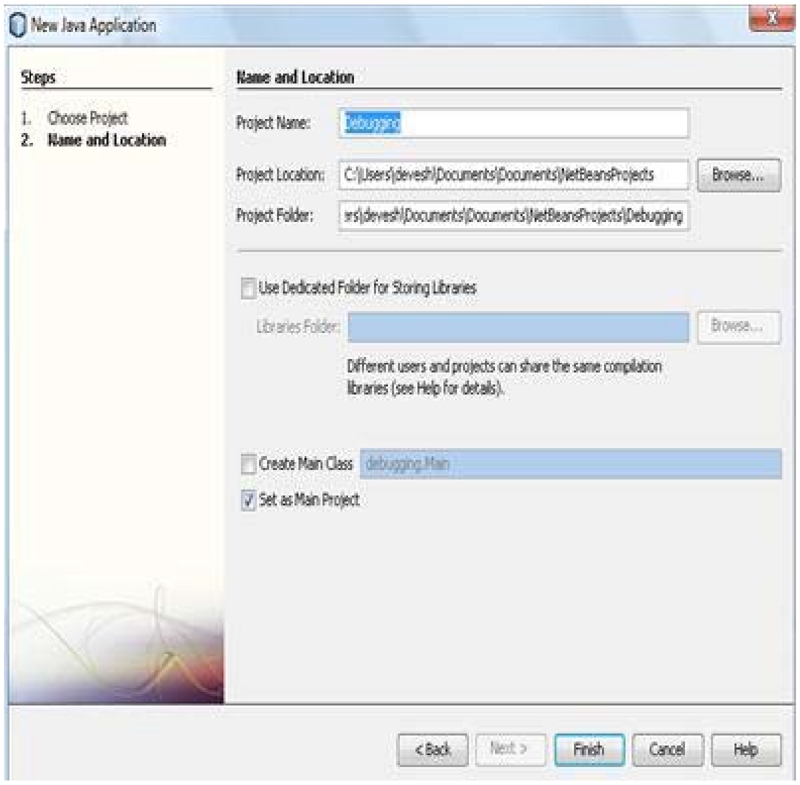
Make sure that NetBeans 6.7 IDE is open on your desktop. Go to **File** on the left top side of the menu bar and select **New Project**.

In the pop menu that appears, select **Java** under categories and **Java Application** under Projects. Click **Next >** button.



Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

Name the project as **"Debugging"** and select appropriate project location. Uncheck the check box **Create Main Class**.


New Java Application

Steps
1. Choose Project
2. Name and Location

Name and Location

Project Name: Debugging

Project Location: C:\Users\devesh\Documents\Documents\NetBeansProjects

Project Folder: ers\devesh\Documents\Documents\NetBeansProjects\Debugging

[ ] Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

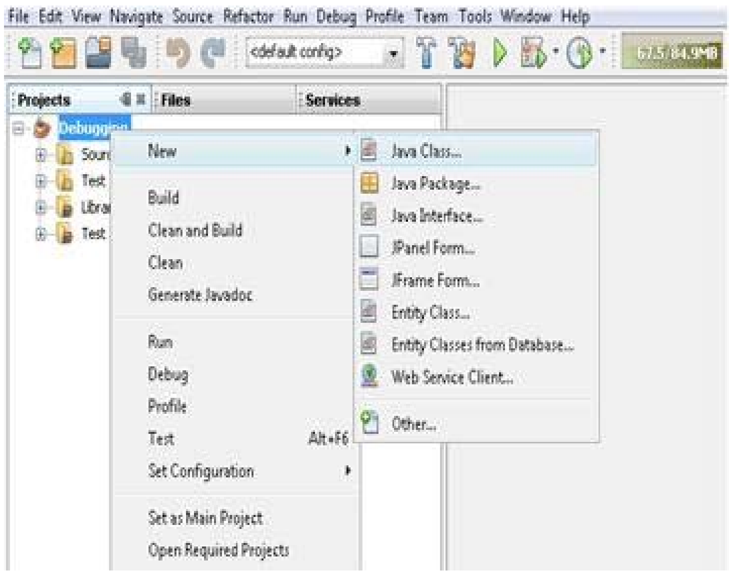
[ ] Create Main Class debugging.Main

[✓] Set as Main Project

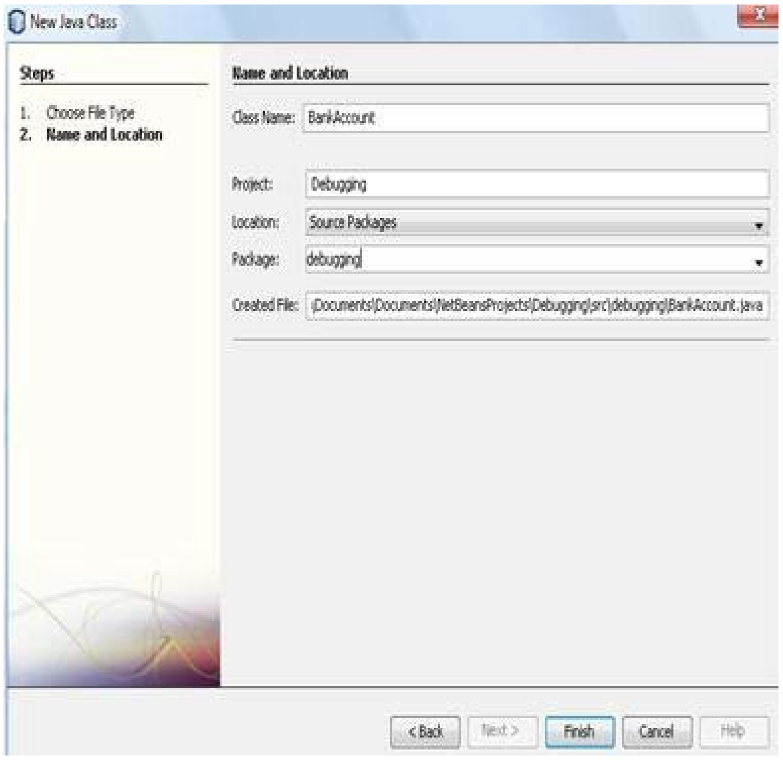
Click on **Finish** button.

Note that a project with the name **Debugging** is being created on the left pane under projects.

Right click on the **Debugging** project and select **New --> Java Class**.



In the pop window that appears, name the class as **BankAccount** and name the package as **debugging**.



Class Name: BankAccount

Project: Debugging

Location: Source Packages

Package: debugging

Created File: \Documents\Documents\NetBeansProjects\Debugging\src\debugging\BankAccount.java

Click on **Finish** button.

Add the following code to the **BankAccount** class.

```
private int accntNo;
private String custName;
private double balance;

/* Constructor for creating a Bank Account with zero balance */
public BankAccount(int newAccntNo, String newCustName)
{
this.accntNo = newAccntNo;
this.custName = newCustName;
this.balance = 0.0;
}

/* Constructor for creating a Bank Account with non-zero balance */
public BankAccount(int newAccntNo, String newCustName, double newBal)
{
this.accntNo = newAccntNo;
this.custName = newCustName;
this.balance = newBal;
}

/* Method to make a deposit to an account */
public void deposit (double amount)
{
/* Validate the amount being deposited */
if (amount > 0.0)
balance += amount;
else
System.out.println("Transaction  Unsuccessful!");
}
```

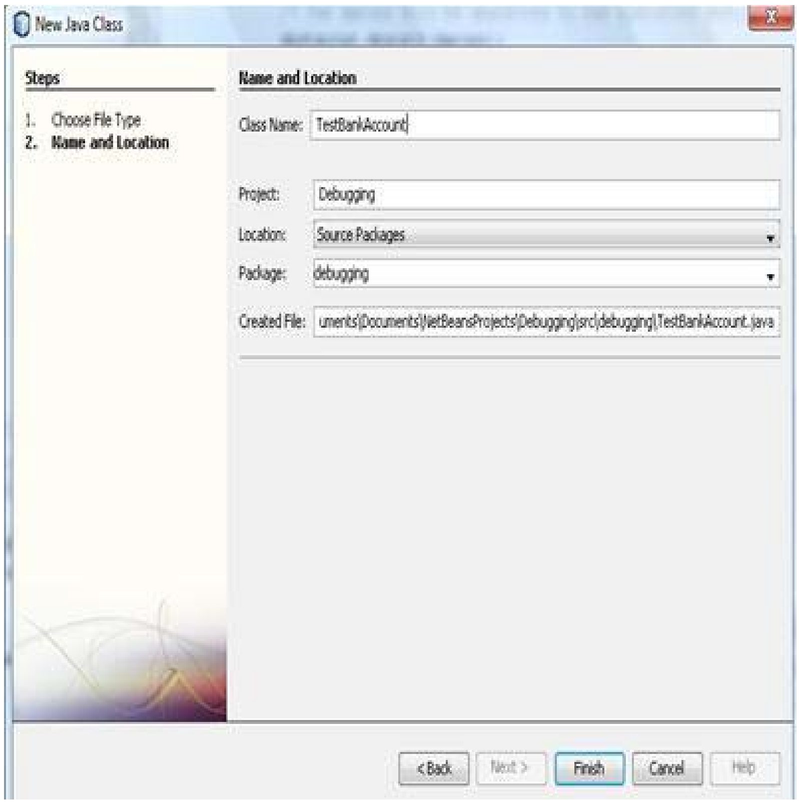
```
/* Method to make a withdrawal from an account */
public void withdraw (double amount)
{
/* Validate the amount being deposited. The balance should be more than the withdrawal amount */
if (amount > 0.0 && amount <= balance)
balance -= amount;
else
System.out.println("Transaction  Unsuccessful!");
}

/* Method to transfer an amount from one account to another */
public void transfer (BankAccount destAccnt, double amount)
{
/* Validate the amount being deposited. The balance should be more than the withdrawal amount */
if (amount > 0.0 && amount >= this.balance)
{
/* The amount will be deposited to the specified destination account */
destAccnt.deposit(amount);

/* The amount will be withdrawn from the source account that has initiated the transfer*/
/* 'this' refers to the calling object (the one whose name appeared in front of the dot in the method call) */
this.withdraw(amount);
}
else
System.out.println("Transaction  Unsuccessful!");
}

/* Method to display the status of the account */
public void display()
{
System.out.println("Account Number: " +accntNo);
System.out.println("Customer Name: " +custName);
System.out.println("Balance: $" +balance);
}
```

It is **highly recommended** that you follow the comments in the code to understand the flow of program.

Note: Remember that you can format your code by right clicking anywhere inside the class and selecting **Format**. It will align your code properly.

Right click on the **Debugging** project and select **New --> Java Class**.

Name the class as **TestBankAccount** and select the package as **debugging**.



Name and Location

Class Name: TestBankAccount

Project: Debugging

Location: Source Packages

Package: debugging

Created File: uments\Documents\NetBeansProjects\Debugging\src\debugging\TestBankAccount.java

Click on **Finish** button.

Add the following code to the **TestBankAccount** class.

```
public static void main(String args [])
{

/* Create a new account with no balance */
BankAccount account1 = new BankAccount (1, "John Doe");

/* Create a new account with non zero balance */ BankAccount
account2 = new BankAccount (2, "Michael", 1000);

System.out.println("Before  Deposit");
System.out.println("--------------");
account1.display(); account1.deposit(100);
System.out.println("After  Deposit");
System.out.println("-------------");
account1.display(); System.out.println("");

System.out.println("Before  Withdrawal");
System.out.println("-----------------");
account2.display();
account2.withdraw(800);
System.out.println("After  Withdrawal");
System.out.println("----------------");
account2.display(); System.out.println("");
System.out.println("After  Transfer");
System.out.println("--------------");
account2.transfer(account1,50);
account1.display();
account2.display();
}
```

Save both the classes and click on the **Run** icon on the top of the menu bar or by right clicking on the project and select **Run**.

The output in the output window at the bottom of the screen looks like this:

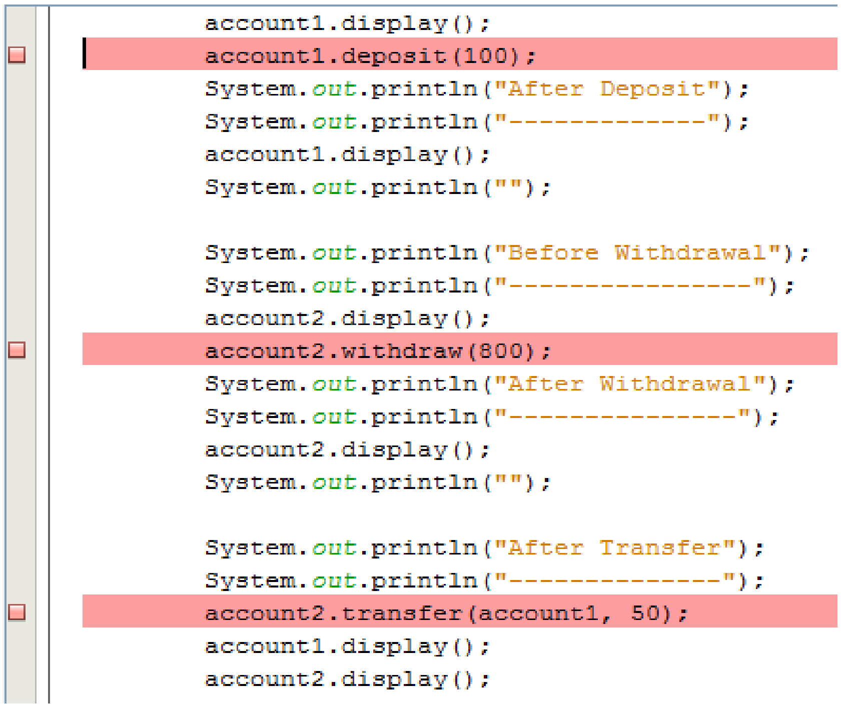
```
Before Deposit
--------------
Account Number: 1
Customer Name: John Doe
Balance: $0.0
After Deposit
-------------
Account Number: 1
Customer Name: John Doe
Balance: $100.0


Before Withdrawal
----------------
Account Number: 2
Customer Name: Michael
Balance: $1000.0
After Withdrawal
---------------
Account Number: 2
Customer Name: Michael
Balance: $200.0


After Transfer
--------------
Transaction Unsuccessful!
Account Number: 1
Customer Name: John Doe
Balance: $100.0
Account Number: 2
Customer Name: Michael
Balance: $200.0
```

Notice that it says "Transaction Unsuccessful" under After Transfer. We need to figure out what went wrong and in which method!

We're now going to track down the cause of the problem using the debugger. Do not solve the problem just by looking at the code. The point of this exercise is to learn how to use the debugger to track down the culprit.We want to step through the program one step at a time, in order to figure out what is wrong with the code. To do this, we first set a **breakpoint** in the code. A **breakpoint** is simply a flag that tells NetBeans "Stop here if you're debugging."

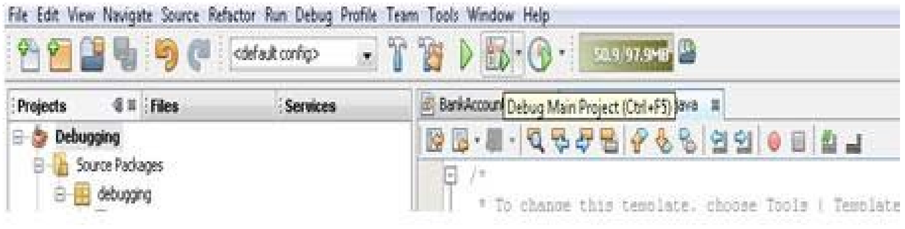
To set a break point you go to the line of code that you wish to set it on and then click on the grey border on the left. This will put a breakpoint on this line.

Select breakpoints on three methods account1.deposit(100), account2.withdraw(800) and account2.transfer(account1,50) in the **TestBankAccount** class by simply clicking on the grey border on the left.



```
        account1.display();
        account1.deposit(100);
        System.out.println("After Deposit");
        System.out.println("-------------");
        account1.display();
        System.out.println("");

        System.out.println("Before Withdrawal");
        System.out.println("----------------");
        account2.display();
        account2.withdraw(800);
        System.out.println("After Withdrawal");
        System.out.println("---------------");
        account2.display();
        System.out.println("");

        System.out.println("After Transfer");
        System.out.println("--------------");
        account2.transfer(account1, 50);
        account1.display();
        account2.display();
```

Click on the Debug Main Project icon on the top of the menu bar.



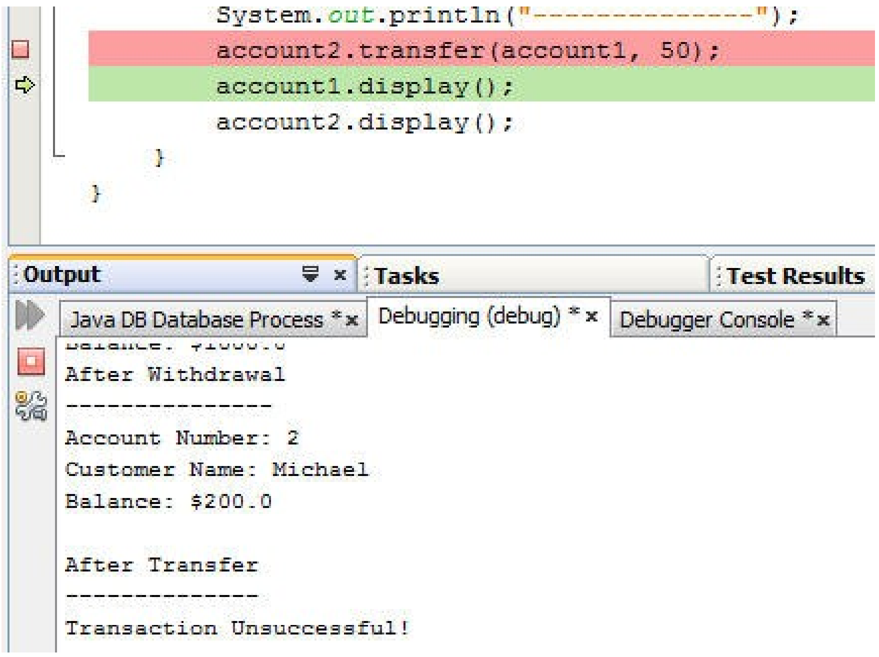
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
50.9/97.9MB
Projects
Files
Services
Debugging
Source Packages
debugging
BankAccount
Debug Main Project (Ctrl+F5)
* To change this template, choose Tools | Template

To walk through the program step by step, you'll use the three icons in the toolbar in the Debug pane of this window, which is the top pane. The three icons are **Step Over**, **Step Into,** and **Step Out**.

The **Step Over** icon executes the method call and moves on to the next line of code so that you don't have to see every detail of every method. The **Step Into** icon takes you into a method call so that you can see the code in the method being executed. The **Step Out** icon takes you out of a method call and back to the caller of the routine. Note that stepping out does ***not*** mean you are skipping the execution of the method--you are merely done with the inspection of the code in the method and thus would like to go back.

First quickly go through the program by clicking on **Step Over** to determine the method of unsuccessful transaction.

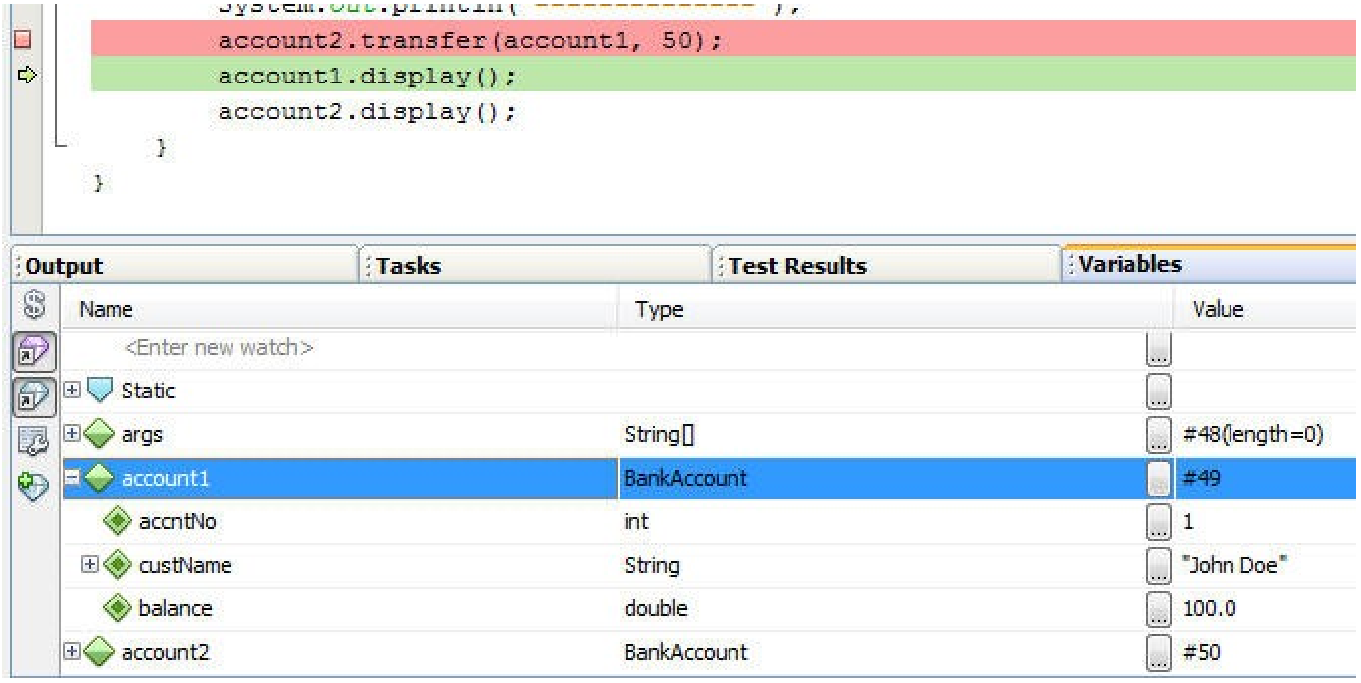
Note that you see "Transaction Unsuccessful" in the Debugging pane at the bottom of the IDE when the program executes account2.transfer(account1,50) method. So, it is now clear that the problem is in the transfer method. We will now look into the transfer method.



```
        System.out.println("---------------");
        account2.transfer(account1, 50);
        account1.display();
        account2.display();
    }
}
```

```
After Withdrawal
---------------
Account Number: 2
Customer Name: Michael
Balance: $200.0

After Transfer
--------------
Transaction Unsuccessful!
```

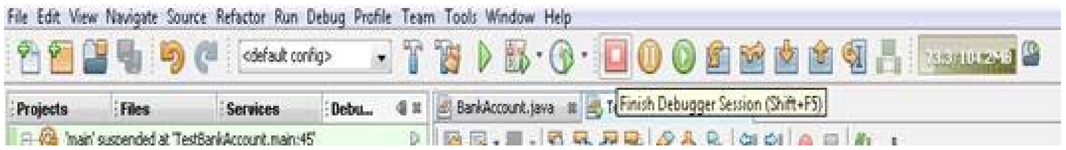
Note: We can also monitor all the variables of the program by clicking on the **Variables** pane on the bottom of the IDE. Expand the '+' sign before any of the account and we can monitor the values of all the variables as the program executes.



```
        account2.transfer(account1, 50);
        account1.display();
        account2.display();
    }
}
```

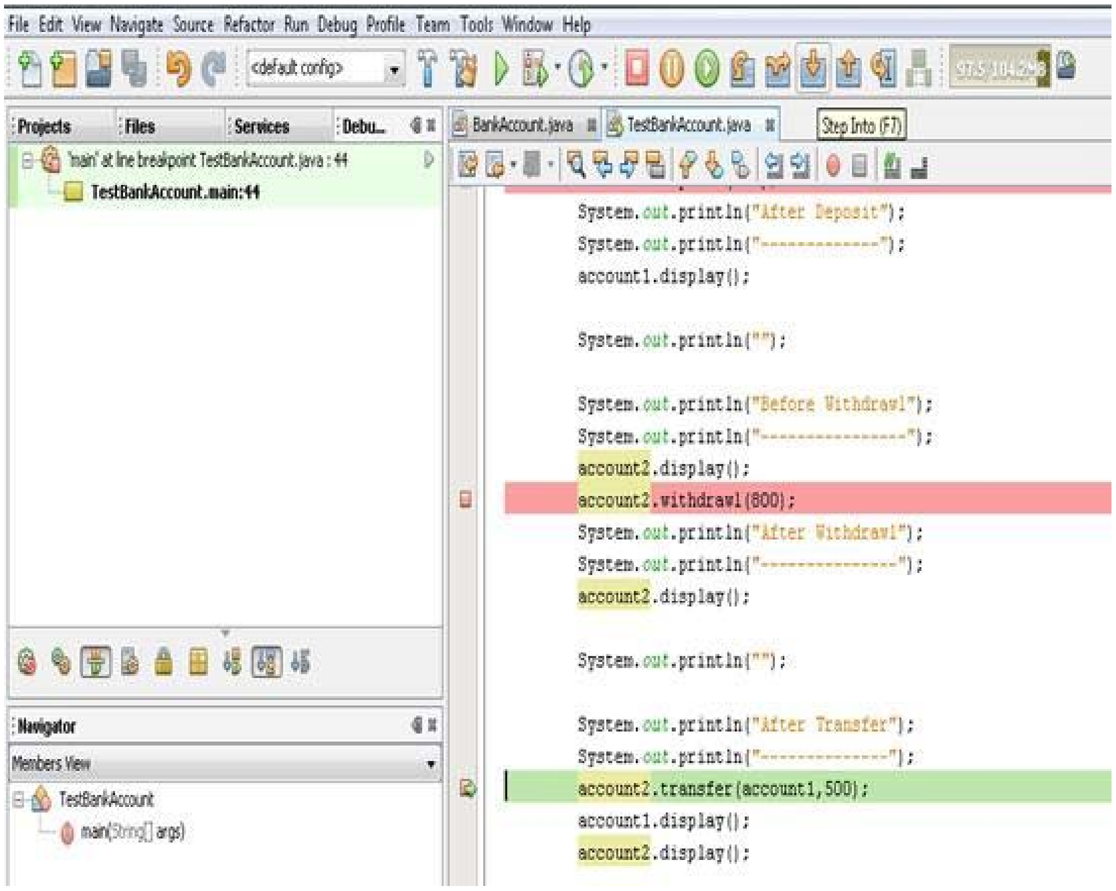
Name	Type	Value
<Enter new watch>		
Static		
args	String[]	#48(length=0)
account1	BankAccount	#49
accntNo	int	1
custName	String	"John Doe"
balance	double	100.0
account2	BankAccount	#50

Stop the debugging by clicking on **Finish** icon on the top of the IDE.



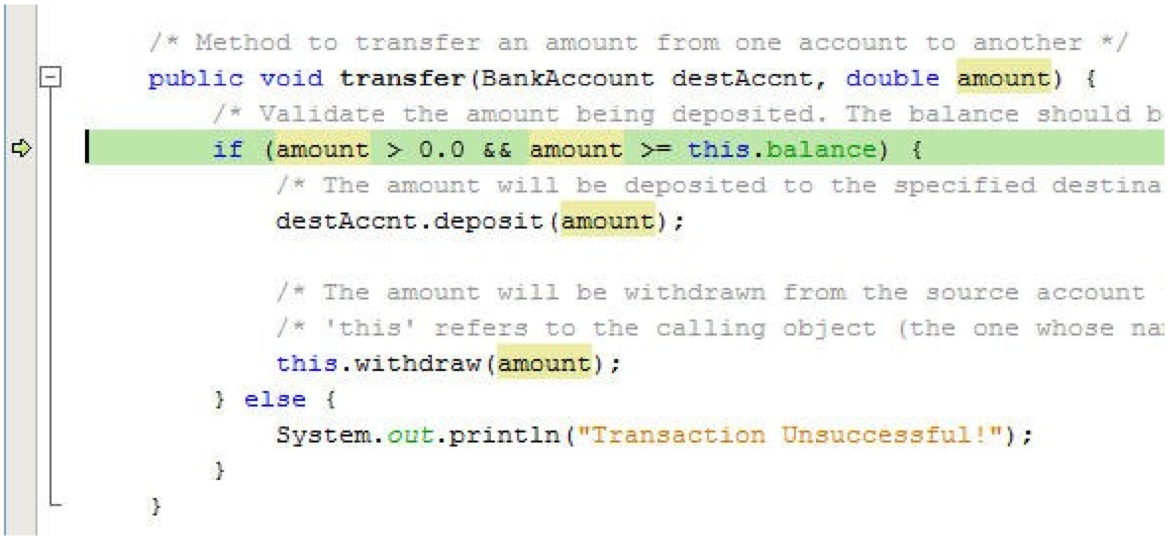
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Projects
Files
Services
Debu...
BankAccount.java
Finish Debugger Session (Shift+F5)
'main' suspended at 'TestBankAccount.main:45'

Click on the **Debug Main Project** icon on the top of the menu bar again and continue **Step Over** until the transfer method. As soon as the green bar reaches on transfer method press **Step Into** icon on the top of the menu bar.



```
        System.out.println("After Deposit");
        System.out.println("------------");
        account1.display();

        System.out.println("");

        System.out.println("Before Withdrawl");
        System.out.println("---------------");
        account2.display();
        account2.withdrawl(800);
        System.out.println("After Withdrawl");
        System.out.println("--------------");
        account2.display();

        System.out.println("");

        System.out.println("After Transfer");
        System.out.println("-------------");
        account2.transfer(account1,500);
        account1.display();
        account2.display();
```

Notice that now the control shifts to the **"transfer"** method in the **BankAccount** class.



```
    /* Method to transfer an amount from one account to another */
    public void transfer(BankAccount destAccnt, double amount) {
        /* Validate the amount being deposited. The balance should b
        if (amount > 0.0 && amount >= this.balance) {
            /* The amount will be deposited to the specified destina
            destAccnt.deposit(amount);

            /* The amount will be withdrawn from the source account
            /* 'this' refers to the calling object (the one whose na
            this.withdraw(amount);
        } else {
            System.out.println("Transaction Unsuccessful!");
        }
    }
```

Note on the 'this' reference: Notice that the **transfer** method uses a reference called **this**. It should be noted that the **this** reference always refer to the object that has invoked the method. In simple words, *it refers to the calling object* (the one whose name appears in front of the dot in the method call).

You can either use **Step Over** or **Step Into** to go through this method as there is no other method call inside this method. So, both of them performs the same operation.

Click on either **Step Over** or **Step Into** and notice that the if part of the if-else statement in the transfer method is not executed. It is because of the fact that the relational operator is coded wrongly. Therefore, else statement executes and thus "Transaction Unsuccessful."

Now that you have figured out the reason behind unsuccessful transaction, you can click on **Finish** icon on the top menu bar to stop the debugging session and fix the bug.

The purpose of this tutorial was to make you familiar with debugging that involve multiple classes and the use of the **Step Into** feature. It is expected that after this tutorial, you will feel more comfortable with debugging in NetBeans.