5/1/2015

# SSE 554 Project 4

*Encrypting the Bank*

Daryl Ebanks & Josh Deremer

# SSE 554 Project 4

*Encrypting the Bank*

## Introduction

This project is the final installment of the banking concept initially created by Daryl in his first project. Since the inception of the program, multithreading, client-server interaction, and database storage have all been added to the original basic bank program. In this last portion, we will add encryption to the data being transmitted. The data will be encrypted before being sent from the client program to the server program and vice versa. Each direction will have its own encryption method, one created by Josh and the other created by Daryl. Once again, we will begin by discussing some key topics before presenting the integration of encryption into the basic bank program.

## Table of Contents

# Encryption

Encryption is a very important aspect of security with digital information. In many situations, it is undesirable and even dangerous to have one's transmitted data available for the world to read as it is traveling from its source to destination. When information is encrypted, it is not visible to anyone who might try to read it and requires a matching key in order to decrypt and make understandable.

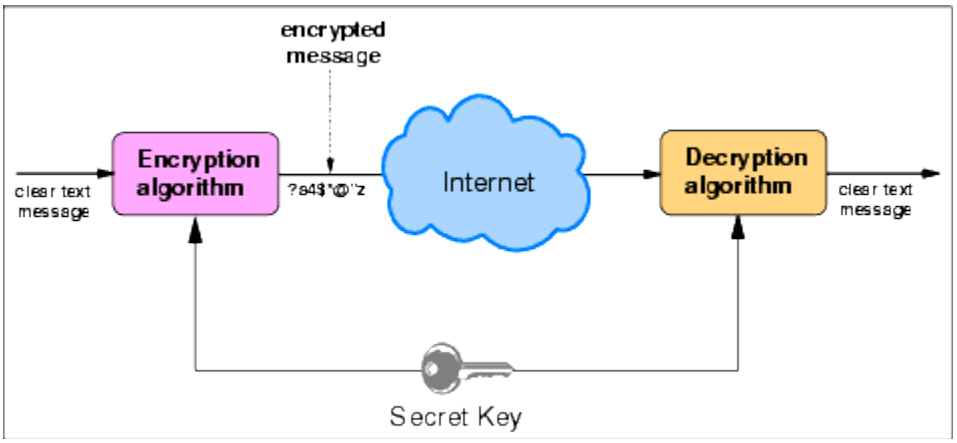
The encryption process involves using a cipher to encode and decode data. Ciphers can be personally created or randomly generated using the utilities available in the Java API. Throughout the years, more and more algorithms have been created to generate stronger and stronger ciphers in order to deter the continuing advancement of cyber thieving technology and computer processing capabilities. Because of this, there is a large number of cipher algorithms available, however, half or more are already outdated and can be easily cracked.

Most ciphers will generally require a cipher key to go along with them. This key can be used to ensure a particular behavior out of the cipher. Any changes in this key will result in changes from the output of the cipher. It is important when encrypting and decrypting data, the same key or matching keys are used, otherwise the data cannot be decrypted properly.

In this project, we make use of two different types of ciphers: symmetric ciphers and public key ciphers.

## Symmetric Ciphers

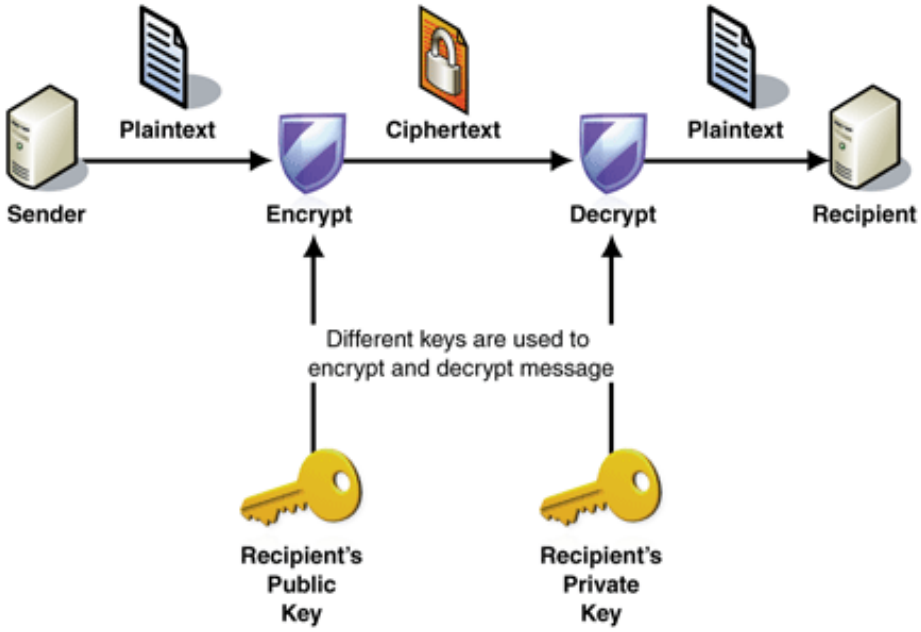
A symmetric cipher is a type of cipher that uses the exact same key for both encryption and decryption. This is a very common encryption method that has a wide variety of cipher algorithms created following this model. The process for a symmetric cipher involves first creating a secret key to be used with both encryption and decryption of data. Then, the user begins data transfer by accepting a plaintext message as the input and the encrypting the message by running it through the encryption algorithm which is governed by the secret key. The encrypted message then travels along the public path where the encryption is required to keep it secure, most often the internet. Once the message reaches its destination, it is run through the decryption algorithm, which is governed by the same secret key as before, and emerges as the original plaintext message. This process can be seen in the image below.




This encryption method is very economical in terms of processing resource usage, however, it requires that the secret key be known by both the sender and receiver of the message. This may be perfectly fine or could pose a serious problem depending on the application. Because of this, other ciphers were created, particularly the public key cipher.

## Public Key Ciphers

The issue mentioned earlier with the symmetric encryption cipher where both the sender and receiver need to know the secret key is known as key distribution. In order for one user to send a message to another user, the secondary user must have the same key the first user uses to encrypt the message. However, the first user cannot just send that new key through an unsecure channel, otherwise the purpose of encrypting the method in the first place is defeated.




This is where public key cryptography comes in. With a public key cipher, the users can have a key pair consisting of a public key and a matching private key. The public key can be access and used by anyone to encrypt a message and send it to the user with the private key. Only the private key can then decrypt the message. Therefore, one

user can simply publish their public key anywhere. The second user can grab the public key and use it to encrypt their message. Then, the second user sends the encrypted message to the first user who decrypts the message using the matching private key.

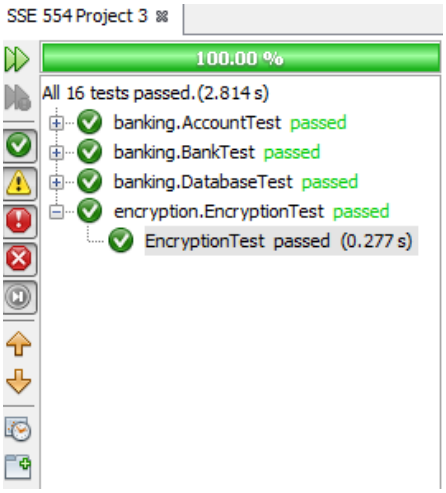
However, as fantastic as that may sound, there are still drawbacks to using public key ciphers. First and foremost is that public key algorithms are much slower than symmetric key algorithms. Because of this, the transfer of large quantities of data is not conceivable for public key ciphers. In order to find a feasible use for ciphers, we combine the two.

# Testing

The testing for this project was very simple. The objective was to create an encryption scheme that could encrypt and decrypt text. The resulting unit test simply creates a string, encrypts it, compares it to the original, decrypts it and compares it again. A lot of time was spent attempting to devise a method of generating a symmetric key on separate machines without explicit communication. When these methods failed, or were too obvious to provide adequate security, they were abandoned for the public/symmetric method described in the book.

The methods passed the test.

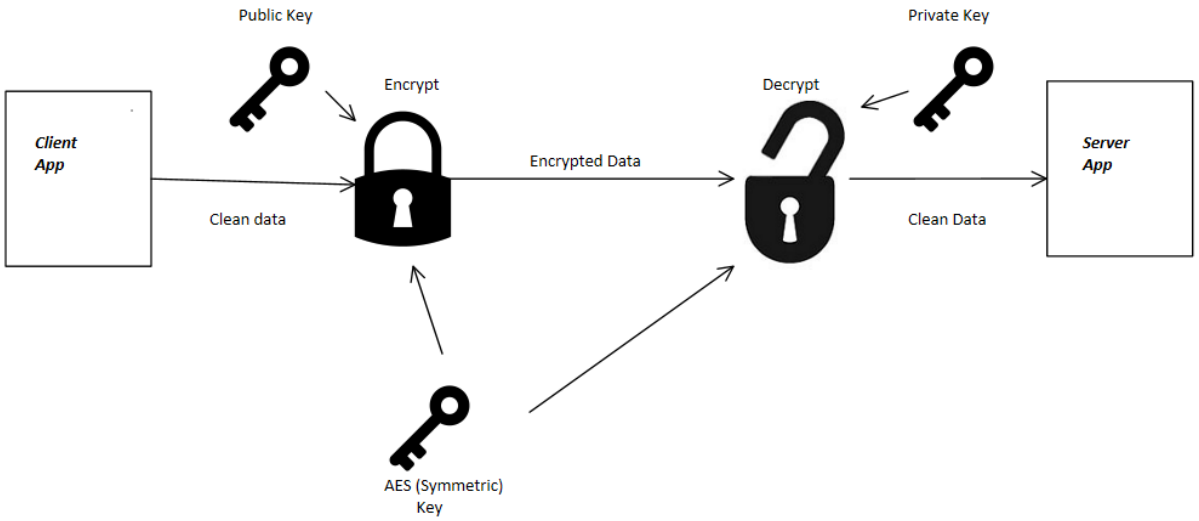
```
public class EncryptionTest {

    public EncryptionTest() {
    }

     @Test
     public void EncryptionTest() {

         String original = "If you want to increase your sucess rate,"
                 + "double your failure rate.";

         EncryptionUtility decrypter = new EncryptionUtility();
         EncryptionUtility encryptor = new EncryptionUtilty(decrypter.getPublicKey());

         byte[] wrappedKey = encryptor.wrapSymmetricKey();
         byte[] encryptedMessage = encryptor.Encrypt(original);

         assertFalse(original.getBytes() == encryptedMessage);

         decrypter.unwrapKey(wrappedKey);
         String decryptedMessage = decrypter.Decrypt(encryptedMessage);

         assertEquals(original, decryptedMessage);
     }
}
```




# Description of Source

## Client to Server Encryption

The client to server encryption method makes use of both the symmetric cipher and the public key cipher. By using the public key cipher first, the server can create a key pair and send the pubic key out to the client. The client then creates a symmetric key and uses the public key to encrypt the symmetric key and send the encrypted key back to the server. Next, the server unencrypts the symmetric key. Now, both the server and the client have a matching symmetric key and can send and receive data using the symmetric cipher method. Using this method, the expensive public key encryption is only applied to a small amount of data.




| ClientEncryptor |
| --- |
| -Cipher cipher<br>~byte[] wrappedKey<br>~SecretKey key |
| +ClientEncryptor(Key publicKey)<br>+byte[] encrypt(String input)<br>+byte[] getKey()<br>+Key getClientKey() |

The symmetric cipher was encrypted using the Advanced Encryption Standard (AES) algorithm. A key for the cipher was generated using the SecureRandom generator from the Java API. The public key cipher made use of the RSA algorithm which was invented by Rivest, Shamir and Adleman. This cipher also generated a random key pair using the SecureRandom generator.

Two classes were created to perform the cryptography. The server side class, ServerDecryptor, keeps the private key for the public key cipher and only decrypts messages from the client side application. The client side class, ClientEncryptor, gets the public key for the public key cipher and only encrypts messages to send to the server application. The encryption and decryption methods convert the byte arrays to strings and vice versa to send and receive the information.

| ServerDecryptor |
| --- |
| -Cipher cipher<br>+final int KEYSIZE<br>-Key publicKey<br>-Key privateKey<br>-SecretKey key |
| +ServerDecryptor()<br>+void initCipher(byte[] wrappedkey)<br>+String decrypt(byte[] input)<br>+Key getPublic()<br>+Key getServerKey() |

The ClientEncryptor was initiated in the client application and encrypts every message before sending it out to the server. Meanwhile, the ServerDecryptor is initiated in the server application and is used to decrypt every message coming from the client.

## Server to Client Encryption

The server to client encryption method also uses the same method of encryption; a symmetric encryption key wrapped by a public key. The main difference is the implementation. The server to client encryption code is implemented as a single class that handles both encryption and decryption.

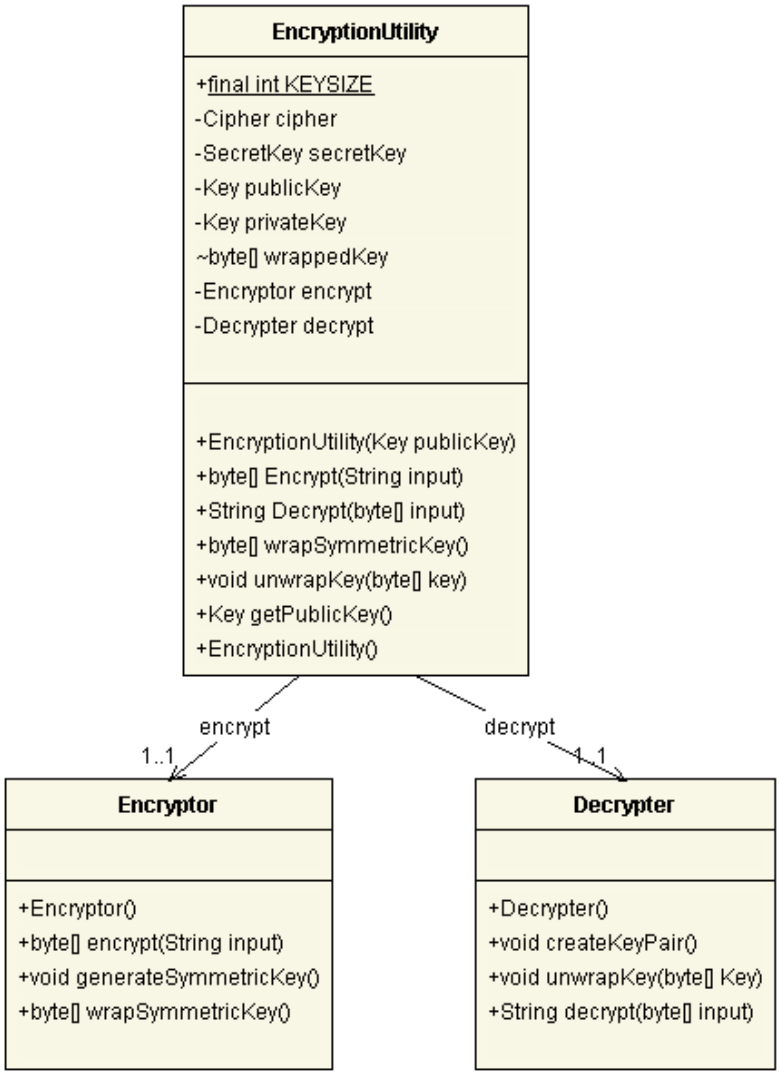
This Encryption Utility class requires the same steps be taken, but some of the steps are bound together for simplicity of use. Each new Encryption Utility can only be used for either encryption or decryption, but it can be used for either.

The class has two helper classes that control separate aspects of the encryption process. The Encryptor and Decrypter, as they are named function as you might expect. These classes are used to ensure that a utility created for encrypting cannot easily be used for decrypting and visa-versa.

The constructor determines how each utility can be used. The Encrypt, Decrypt, wrap Semmetrical Key and unwrap Key methods invoke the methods of the helper classes.

The helper classes themselves contain the logic behind the encryption process. Each of the classes contains the necessary methods for its part of the process.

The diagram describes the relationship between each of the members of the Encryption Utility class.



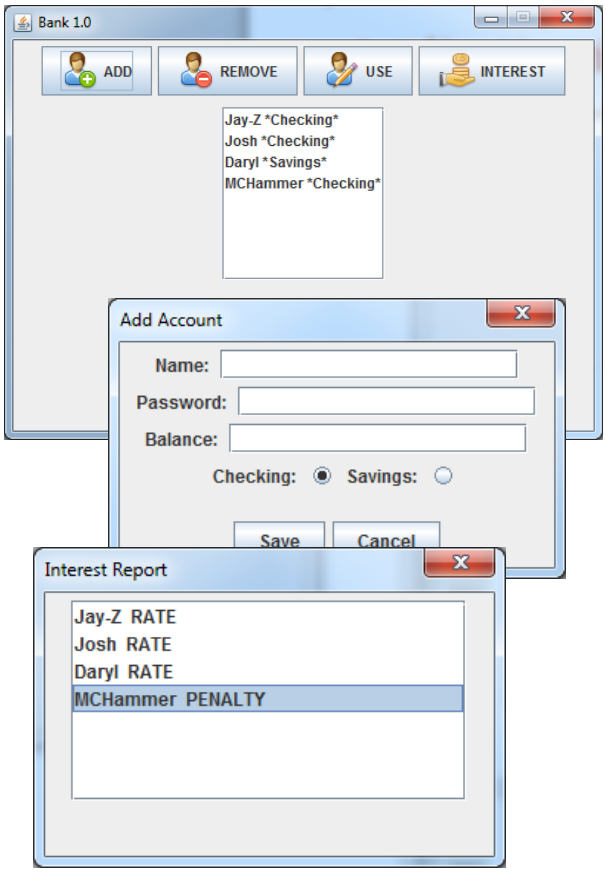
There are small changes in the client and server that were made to accommodate the use of encryption. This is one such example:

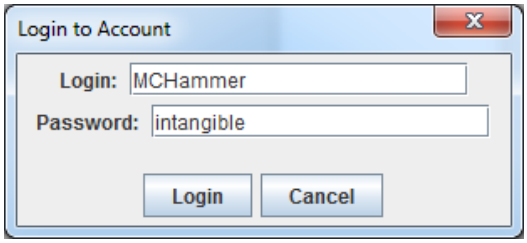
```
switch(todo)
{
        // Add a user
        case 0: complete = add();
        outToClient.writeObject(encryptor.Encrypt(((Boolean)complete).toString()));
        break;

        // Remove a user
        case 1: complete = remove();
        outToClient.writeObject(encryptor.Encrypt(((Boolean)complete).toString()));
        break;
```

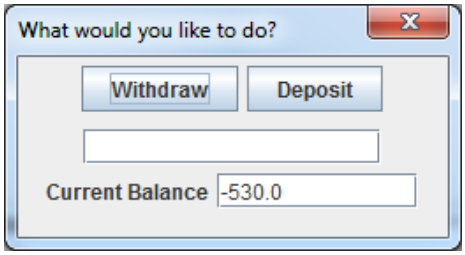
In this example the Encryption Utility encryptor is used to encrypt the message before it is sent across the network.

## Code in Action

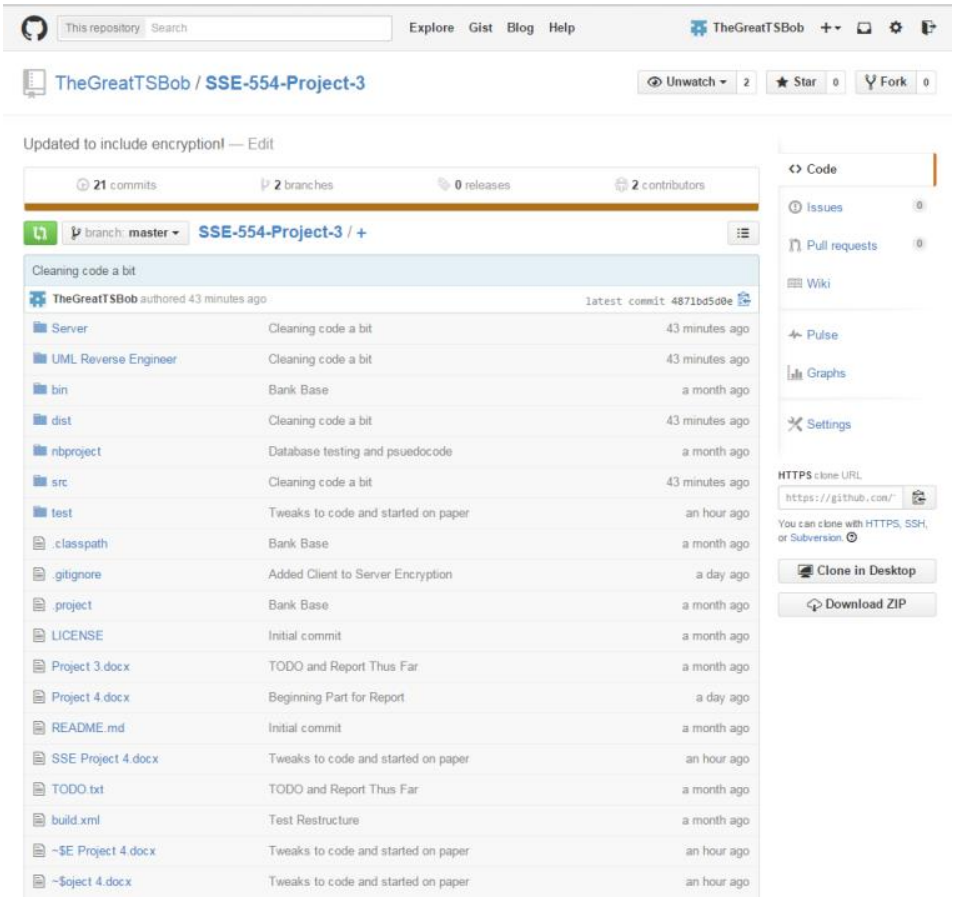
The display and functionality of the code are identical to the previous version. Not much to see here.



# DVCS




The DVCS was instrumental in the collaboration once again. The ability to make changes to code and synchronize these changes with a group makes group work much more feasible and simple.

# Test Code

## Account Test

E:/Newsbin Downloads/Code/SSE-554-Project-3/test/banking/AccountTest.java

```
1  package banking;
2
3  import static org.junit.Assert.*;
4
5  import org.junit.Test;
6
7  public class AccountTest {
8
9          String correctPW = "correct";
10         String incorrectPW = "wrong";
11
12         @Test
13         public void testDeposit() {
14                 CheckingAccount test = new CheckingAccount(100, "Daryl",correctPW);
15                 CheckingAccount test2 = new CheckingAccount(100, "Daryl",correctPW);
16
17
18                 test.deposit(300.00, incorrectPW);
19                 assertEquals(100, test.balance, 0);
20
21                 //tests increasing balance by 100
22                 test.deposit(100.00,correctPW);
23                 assertEquals("Failure - balance should be 200", 200,  test.balance, 0);
24
25                 //No change as deposits cannot be negative
26                 test2.deposit(-100.00,correctPW);
27                 assertEquals(100, test2.balance, 0);
28         }
29
30         @Test
31         public void testWithdraw() {
32                 CheckingAccount test = new CheckingAccount(100, "Daryl",correctPW);
33                 SavingsAccount test2 = new SavingsAccount(100, "Daryl",correctPW);
34
35                 //if the password is incorrect nothing is done
36                 test.withdraw(400, incorrectPW);
37                 assertEquals(100,  test.balance, 0);
38
39                 //tests reducing the balance by 100
40                 test.withdraw(100, correctPW);
41                 assertEquals(0,  test.balance, 0);
42
43                 //No change as withdrawals cannot be negative
44                 test2.withdraw(-100.00, correctPW);
45                 assertEquals(100, test2.balance, 0);
46
47                 //Penalty as max withdraws have been reached
48                 test2.currentWithdrawals = test2.maxWithdrawals;
49                 test2.withdraw(100, correctPW);
50                 assertEquals(90,test2.balance,0);
51
52         }
53
54         @Test
55         public void getBalance() {
56                 CheckingAccount test = new CheckingAccount(100, "Daryl",correctPW);
57
58                 //tests authentication pass and correct return value
59                 assertEquals(100, (double) test.getBalance(correctPW), 0);
60
61                 //tests authentication fail
62                 assertNull(test.getBalance(incorrectPW));
63
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/test/banking/AccountTest.java

```
64          }
65
66          @Test
67          public void testIsOverdrawn() {
68                  CheckingAccount test = new CheckingAccount(0, "Daryl",correctPW);
69
70                  test.balance-=10;
71                  assertTrue(test.isOverdrawn(correctPW));
72          }
73
74          @Test
75          public void testCompoundInterest() {
76                  Account test = new CheckingAccount(0, "Daryl",correctPW);
77                  Account test2 = new CheckingAccount(100, "Daryl",correctPW);
78                  Account test3 = new CheckingAccount( 40, "Daryl",correctPW);
79                  Account test4 = new CheckingAccount(-100, "Daryl",correctPW);
80
81                  assertTrue(test.compoundInterest() == Account.CompoundResult.NONE);
82                  assertEquals(test.balance, 0, 0);
83
84                  assertTrue(test2.compoundInterest() == Account.CompoundResult.RATE);
85                  assertEquals(test2.balance, 110, 0);
86
87                  assertTrue(test3.compoundInterest() == Account.CompoundResult.PENALTY);
88                  assertEquals(test3.balance, 10, 0);
89
90                  assertTrue(test4.compoundInterest() == Account.CompoundResult.PENALTY);
91                  assertEquals(test4.balance, -130, 0);
92
93          }
94
95  }
96
```

# Bank Test

E:/Newsbin Downloads/Code/SSE-554-Project-3/test/banking/BankTest.java

```java
package banking;

import static org.junit.Assert.*;

import java.util.ArrayList;

import org.junit.Test;

import banking.Account.CompoundResult;

public class BankTest {

	String password = "correct";
	String notpassword = "incorrect";

	@Test
	public void addTest() {

		Bank test = new Bank("test");

		int old_size = test.accounts.size();

		test.addAccount(new CheckingAccount(200, "Daryl", password));
		assertEquals(test.accounts.size(), old_size+1);
	}

	@Test
	public void removeTest() {

		Bank test = new Bank("test");
		test.accounts.add(new CheckingAccount(200, "Daryl", password));
		test.accounts.add(new CheckingAccount(300, "Daryl2", password));
		test.accounts.add(new CheckingAccount(400, "Daryl3", password));

		assertEquals(test.accounts.size(), 3);

		test.removeAccount("Daryl3");

		assertEquals(test.accounts.size(), 2);

		test.removeAccount("Daryl2");

		assertEquals(test.accounts.size(), 1);

		test.removeAccount("Daryl4");

		assertEquals(test.accounts.size(), 1);
	}

	@Test
	public void compoundTest() {
		Bank test = new Bank("test");
		test.accounts.add(new SavingsAccount(200, "Daryl", password));
		test.accounts.add(new SavingsAccount(0, "Daryl2", password));
		test.accounts.add(new SavingsAccount(-200, "Daryl3", password));

		assertEquals(test.compoundAccount("Daryl"),CompoundResult.RATE);

		ArrayList<CompoundResult> testList = test.compoundAll();
		ArrayList<CompoundResult> testList2 = new ArrayList<>();

		testList2.add(CompoundResult.RATE);
		testList2.add(CompoundResult.NONE);
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/test/banking/BankTest.java

```
64              testList2.add(CompoundResult.PENALTY);
65
66              for(int i =0; i < testList.size(); i++)
67              {
68                  assertEquals(testList.get(i), testList2.get(i));
69              }
70          }
71
72          @Test
73          public void getBalanceTest() {
74              Bank test = new Bank("test");
75              test.accounts.add(new CheckingAccount(200, "Daryl", password));
76
77              assertEquals(test.getBalance("Daryl", password ),200,0);
78
79          }
80
81          @Test
82          public void withdrawTest() {
83              Bank test = new Bank("test");
84              test.accounts.add(new CheckingAccount(200, "Daryl", password));
85
86              test.withdraw(200,"Daryl", password );
87
88              assertEquals(test.accounts.get(0).balance,0, 0);
89          }
90
91          @Test
92          public void depositTest() {
93              Bank test = new Bank("test");
94              test.accounts.add(new CheckingAccount(200, "Daryl", password));
95
96              test.deposit(200,"Daryl", password );
97
98              assertEquals(test.accounts.get(0).balance,400, 0);
99          }
100
101         @Test
102         public void authenticateTest() {
103             Bank test = new Bank("test");
104             test.accounts.add(new CheckingAccount(200, "DarylTest", password));
105
106             assertTrue(test.authenticateAccount("DarylTest", password));
107             assertFalse(test.authenticateAccount("Daryl", notpassword));
108
109         }
110 }
111
```

# Database Test

E:/Newsbin Downloads/Code/SSE-554-Project-3/test/banking/DatabaseTest.java

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package banking;

import com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException;
import java.sql.*;
import java.util.ArrayList;
import org.junit.Test;

import static org.junit.Assert.*;

/**
 *
 * @author TSBob
 */
public class DatabaseTest {

    @Test(expected= MySQLSyntaxErrorException.class)
    @SuppressWarnings({"CallToPrintStackTrace"})
    public void Test() throws Exception
    {

        Database db = new Database("test");
        Connection conn = db.getConnection("test");

        //connection
        assertTrue(conn != null);

        Statement stat = conn.createStatement();

        //adding to database
        stat.executeUpdate("CREATE TABLE Test (Message CHAR(20))");
        stat.executeUpdate("INSERT INTO Test VALUES ('Hello, World')");


        try (ResultSet result = stat.executeQuery("SELECT * FROM Test"))
        {
            if(result.next())
                assertEquals(result.getString(1), "Hello, World");
        }

        //removing from database
        stat.executeUpdate("DROP TABLE Test");

        //should create an exception because no such table exists
        stat.executeQuery("SELECT * From Test");
    }
    @Test
    public void TestWriteToDatabase() throws Exception
    {
        ArrayList<Account> accounts = new ArrayList<>();
        accounts.add(new CheckingAccount(200,"DarylTest", "password"));

        Database test = new Database("test");
        Connection link = test.getConnection("test");
        Statement stat = link.createStatement();

        stat.executeUpdate("CREATE table Bank("
                + "holder varchar(45),"
                + "password varchar(45),"
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/test/banking/DatabaseTest.java

```
64                      + "balance float,"
65                      + "accounttype varchar(10),"
66                      + "rate float,"
67                      + "withdrawals int,"
68                      + "minbalance double);");
69
70          test.writeToDatabase(accounts);
71
72          ResultSet result = stat.executeQuery("SELECT * from Bank");
73          result.next();
74          assertEquals(result.getString("holder"), "DarylTest");
75          assertEquals(result.getDouble("balance"), 200.0, 0);
76          assertEquals(result.getString("password"), "password");
77
78          stat.executeUpdate("DROP TABLE Bank");
79      }
80
81      @Test
82      public void TestReadFromDatabase() throws Exception
83      {
84          ArrayList<Account> accounts = new ArrayList<>();
85          Database test = new Database("test");
86
87          Connection link = test.getConnection("test");
88          Statement stat = link.createStatement();
89
90          stat.executeUpdate("CREATE table Bank("
91                      + "holder varchar(45),"
92                      + "password varchar(45),"
93                      + "balance float,"
94                      + "accounttype varchar(10),"
95                      + "rate float,"
96                      + "withdrawals int,"
97                      + "minbalance double);");
98
99          stat.executeUpdate("INSERT INTO Bank "
100                     + "(holder, balance, password, accounttype)"
101             + " Value('DarylTest', 400, 'pass', 'Checking')");
102
103         accounts = test.readFromDatabase(accounts);
104
105         for(Account a : accounts)
106         {
107             assertEquals(a.holder, "DarylTest");
108             assertEquals(a.balance, 400, 0);
109             assertEquals(a.password, "pass");
110             assertTrue(a.getClass()== CheckingAccount.class);
111         }
112
113         stat.executeUpdate("DROP TABLE Bank");
114     }
115 }
116
```

# Encryption Test

E:/Newsbin Downloads/Code/SSE-554-Project-3/test/encryption/EncryptionTest.java

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package encryption;
7
8
9  import java.security.NoSuchAlgorithmException;
10 import java.security.SecureRandom;
11 import java.util.Arrays;
12 import javax.crypto.KeyGenerator;
13 import javax.crypto.SecretKey;
14 import org.junit.Test;
15 import static org.junit.Assert.*;
16
17 /**
18  *
19  * @author TSBob
20  */
21 public class EncryptionTest {
22
23     public EncryptionTest() {
24     }
25
26      @Test
27      public void EncryptionTest() {
28
29          String original = "If you want to increase your sucess rate,"
30                  + "double your failure rate.";
31
32          EncryptionUtility decrypter = new EncryptionUtility();
33          EncryptionUtility encryptor = new EncryptionUtility(decrypter.getPublicKey());
34
35          byte[] wrappedKey = encryptor.wrapSymmetricKey();
36          byte[] encryptedMessage = encryptor.Encrypt(original);
37
38          assertFalse(original.getBytes() == encryptedMessage);
39
40          decrypter.unwrapKey(wrappedKey);
41          String decryptedMessage = decrypter.Decrypt(encryptedMessage);
42
43          assertEquals(original, decryptedMessage);
44      }
45 }
46
```

# Source Code

## Account

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/banking/Account.java

```
1  package banking;
4  public abstract class Account {
5
6      protected double balance;
7      protected String holder;
8      protected String password;
9      protected double rate = .2;
10
11     public enum CompoundResult{
12         NONE,
13         RATE,
14         PENALTY
15     };
16
17     public Account(double balance, String holder, String password)
18     {
19         this.balance = balance;
20         this.holder = holder;
           this.password = password;
       }

       protected void deposit(double amount, String password)
       {
           if(!authenticate(password))
               return;

           if(amount > 0)
               balance += amount;
       }

       protected abstract void withdraw(double amount, String password);

       protected Double getBalance(String password)
       {
           if(authenticate(password))
               return balance;

           return null;
       }

       protected Boolean isOverdrawn(String password)
       {
           if(authenticate(password))
               return balance < 0;

           return null;
       }

       protected abstract CompoundResult compoundInterest();


       protected Boolean authenticate(String password)
       {
           if(this.password.compareTo(password) == 0)
               return true;
           return false;
       }

}
```

# Bank

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/banking/Bank.java

```
1  package banking;
2
3  import java.util.ArrayList;
4
5  import banking.Account.CompoundResult;
6  import java.io.IOException;
7  import java.sql.SQLException;
8  import java.util.logging.Level;
9  import java.util.logging.Logger;
10
11 public class Bank {
12
13     protected ArrayList<Account> accounts;
14     public Database database;
15
16     public Bank()
17     {
18         accounts = new ArrayList<>();
19         try {
20             database = new Database();
               database.readFromDatabase(accounts);

           } catch (SQLException | IOException ex) {
               Logger.getLogger(Bank.class.getName()).log(Level.SEVERE, null,
                       ex);
           }
       }

       public void addAccount(Account acc)
       {
           accounts.add(acc);

           /************************************
            *
            *  Consider adding a database refresh when an account is added
            *  removed or changed
            *
            *
            ************************************/

           setData();
       }

       public void removeAccount(String holder)
       {
           int index = getIndex(holder);

           if( index < 0)
               return;

           accounts.remove(index);
           /************************************
            *
            *  Consider adding a database refresh when an account is added
            *  removed or changed
            *
            *
            ************************************/

           setData();
       }

       public int getIndex(String holder)
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/banking/Bank.java

```
        {
            for(Account a : accounts)
            {
                if(a.holder.compareTo(holder) == 0)
                    return accounts.indexOf(a);
            }
            return -1;
        }

        public String getHolder(int index)
        {
            return accounts.get(index).holder;
        }

        public ArrayList<CompoundResult> compoundAll()
        {
            ArrayList<CompoundResult> value = new ArrayList<>();

            for (Account a : accounts)
            {
                value.add(a.compoundInterest());
            }

            /**********************************
             *
             *  Consider adding a database refresh when an account is added
             *  removed or changed
             *
             *
             *********************************/

            setData();

            return value;

        }

        public CompoundResult compoundAccount(String holder)
        {
            int index = getIndex(holder);

            CompoundResult res = accounts.get(index).compoundInterest();

            return res;

        }

        public Double getBalance(String holder, String password)
        {
            int index = getIndex(holder);

            return accounts.get(index).getBalance(password);
        }

        public Boolean authenticateAccount(String holder, String password)
        {
            int index = getIndex(holder);

            if (index < 0)
                return false;

            return accounts.get(index).authenticate(password);
        }
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/banking/Bank.java

```
    public void withdraw(double amount, String holder, String password)
    {
        int index = getIndex(holder);

        accounts.get(index).withdraw(amount, password);

        /************************************
        *
        *  Consider adding a database refresh when an account is added
        *  removed or changed
        *
        *
        ************************************/

        setData();
    }

    public void deposit(double amount, String holder, String password)
    {
        int index = getIndex(holder);

        accounts.get(index).deposit(amount, password);

        /************************************
        *
        *  Consider adding a database refresh when an account is added
        *  removed or changed
        *
        *
        ************************************/

        setData();
    }

    public ArrayList<String> getLabels()
    {
        ArrayList<String> labels = new ArrayList<>();

        for(Account a : accounts)
        {
            labels.add(a.toString());
        }

        return labels;
    }

    public boolean isChecking(int index)
    {
        return accounts.get(index).getClass() == CheckingAccount.class;
    }

    public int remainingWithdrawals(int index)
    {
        if(accounts.get(index).getClass() != SavingsAccount.class)
            return 0;

        return ((SavingsAccount) accounts.get(index)).maxWithdrawals -
                ((SavingsAccount) accounts.get(index)).currentWithdrawals;
    }

    public int size()
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/banking/Bank.java

```
        {
            return accounts.size();
        }

        public void getData()
        {
            try {
                accounts = database.readFromDatabase(accounts);
            } catch (SQLException ex) {
                Logger.getLogger(Bank.class.getName()).log(Level.SEVERE, null,
                        ex);
            }
        }

        public void setData()
        {
            try {
                database.writeToDatabase(accounts);
            } catch (SQLException ex) {
                Logger.getLogger(Bank.class.getName()).log(Level.SEVERE, null,
                        ex);
                ex.printStackTrace();
            }
        }
}
```

# BankClient

E:/Newsbin Downloads/Code/SSE-554-Project-3/src/BankClient.java

```
1
2  import java.io.IOException;
3  import java.io.ObjectInputStream;
4  import java.io.ObjectOutputStream;
5  import java.net.Socket;
6  import java.util.ArrayList;
7  import Encryption.*;
8  import java.security.Key;
9
10 public class BankClient {
11
12     static int portNumber = 4444;
13     static String computerName = "localhost";
14     ObjectOutputStream outToServer;
15     ObjectInputStream inFromServer;
16     ClientEncryptor ce;
17     EncryptionUtility decrypter;
18
19     public BankClient()
20     {
21
22     }
23
24     public void connect() {
25         System.out.println("Attempting to connect to " + computerName + ":" + portNumber);
26         try {
27             Socket csocket = new Socket(computerName, portNumber);
28             System.out.println("Connection successful!");
29
30             outToServer = new ObjectOutputStream(csocket.getOutputStream());
31             inFromServer = new ObjectInputStream(csocket.getInputStream());
32
33             System.out.println((String)inFromServer.readObject());
34
35             ce = new ClientEncryptor((Key)inFromServer.readObject());
36             outToServer.writeObject(ce.getKey());
37
38             decrypter = new EncryptionUtility();
39             outToServer.writeObject(decrypter.getPublicKey());
40             decrypter.unwrapKey((byte[]) inFromServer.readObject());
41
42             // work with server
44         } catch (Exception e) {
45             e.printStackTrace();
46         }
47     }
48
49     public ArrayList<String> init()
50     {
51         try
52         {
54             return (ArrayList<String>)inFromServer.readObject();
55         } catch (IOException e) {e.printStackTrace();}
56         catch (ClassNotFoundException e) {e.printStackTrace();}
57         return null;
58     }
59
60     public void addAccount(String name, String password, double balance, Integer accountType)
61     {
62         try
63         {
64             Integer i = 0;
65             outToServer.writeObject(ce.encrypt(Integer.toString(i)));
66             outToServer.writeObject(ce.encrypt(name));
67             outToServer.writeObject(ce.encrypt(password));
68             outToServer.writeObject(ce.encrypt(Double.toString(balance)));
69             outToServer.writeObject(ce.encrypt(Integer.toString(accountType)));
70
71             boolean complete = Boolean.parseBoolean(
72                     decrypter.Decrypt( (byte []) inFromServer.readObject()));
73
74         } catch (IOException e) {e.printStackTrace();}
75         catch (ClassNotFoundException e) {e.printStackTrace();}
76     }
77
78     public void removeAccount(String name)
79     {
80         try
81         {
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/src/BankClient.java

```
80          Integer i = 1;
81          outToServer.writeObject(ce.encrypt(Integer.toString(i)));
82          outToServer.writeObject(ce.encrypt(name));
83          boolean complete = Boolean.parseBoolean(
84                  decrypter.Decrypt( (byte []) inFromServer.readObject()));
85      } catch (IOException e) {e.printStackTrace();}
86      catch (ClassNotFoundException e) {e.printStackTrace();}
87  }
88
89  public boolean authenticate(String name, String password)
90  {
91      try
92      {
93          Integer i = 2;
94          outToServer.writeObject(ce.encrypt(Integer.toString(i)));
95          outToServer.writeObject(ce.encrypt(name));
96          outToServer.writeObject(ce.encrypt(password));
97          return (boolean)Boolean.parseBoolean(
98                  decrypter.Decrypt( (byte []) inFromServer.readObject()));
99      } catch (IOException e) {e.printStackTrace();}
100     catch (ClassNotFoundException e) {e.printStackTrace();}
101
102     return false;
103 }
104
105 public boolean accountAction(String name, String password,double amount)
106 {
107     try
108     {
109         Integer i = 3;
110         outToServer.writeObject(ce.encrypt(Integer.toString(i)));
111         outToServer.writeObject(ce.encrypt(name));
112         outToServer.writeObject(ce.encrypt(Double.toString(amount)));
113         outToServer.writeObject(ce.encrypt(password));
114         boolean complete = Boolean.parseBoolean(
115                 decrypter.Decrypt( (byte []) inFromServer.readObject()));;
116         return complete;
117     } catch (IOException e) {e.printStackTrace();}
118     catch (ClassNotFoundException e) {e.printStackTrace();}
119     return false;
120 }
121
122 public ArrayList<String> interest()
123 {
124
125     try
126     {
127         Integer i = 4;
128         outToServer.writeObject(ce.encrypt(Integer.toString(i)));
129         ArrayList<byte[]> encryptedResults = (ArrayList<byte[]>)inFromServer.readObject();
130         ArrayList<String> results = new ArrayList<>();
131         for(byte[] encryptedName: encryptedResults)
132         {
133             results.add(decrypter.Decrypt(encryptedName));
134         }
135
136         boolean complete = Boolean.parseBoolean(
137                 decrypter.Decrypt( (byte []) inFromServer.readObject()));;
138
139         return results;
140     } catch (IOException e) {e.printStackTrace();}
141     catch (ClassNotFoundException e) {e.printStackTrace();}
142
143     return null;
144 }
145
146 public ArrayList<String> initAccounts()
147 {
148     try
149     {
150         Integer i = 5;
151         outToServer.writeObject(ce.encrypt(Integer.toString(i)));
152         ArrayList<byte[]> encryptedNames = (ArrayList<byte[]>)inFromServer.readObject();
153         ArrayList<String> accounts = new ArrayList<>();
154         for(byte[] encryptedName: encryptedNames)
155         {
156             accounts.add(decrypter.Decrypt(encryptedName));
157         }
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/src/BankClient.java

```
158             boolean complete = Boolean.parseBoolean(
                        decrypter.Decrypt( (byte []) inFromServer.readObject()));;
160             return accounts;
            } catch (IOException e) {e.printStackTrace();}
162         catch (ClassNotFoundException e) {e.printStackTrace();}
163
164         return null;
165     }
166
167     public Double getBalance(String name, String password)
168     {
169         Double balance = 0.0;
170
171         try
172         {
173             Integer i = 6;
174             outToServer.writeObject(ce.encrypt(Integer.toString(i)));
175             outToServer.writeObject(ce.encrypt(name));
176             outToServer.writeObject(ce.encrypt(password));
177             String temp = decrypter.Decrypt((byte[])inFromServer.readObject());
178             balance = Double.parseDouble(temp);
            } catch (IOException e) {e.printStackTrace();}
180         catch (ClassNotFoundException e) {e.printStackTrace();}
181
182         return balance;
183     }
184
185     public String withdrawCheck(String name)
186     {
187         String text = "";
188         try
189         {
190             Integer i = 7;
191             outToServer.writeObject(ce.encrypt(Integer.toString(i)));
192             outToServer.writeObject(ce.encrypt(name));
193             text = decrypter.Decrypt((byte[])inFromServer.readObject());
            } catch (IOException e) {e.printStackTrace();}
195         catch (ClassNotFoundException e) {e.printStackTrace();}
196
197         return text;
198     }
199 }
200
```

# BankGUI

E:/Newsbin Downloads/Code/SSE-554-Project-3/src/BankGUI.java

```
1  import java.awt.BorderLayout;
2  import java.awt.event.ActionEvent;
3  import java.awt.event.ActionListener;
4  import java.util.ArrayList;
5
6  import javax.swing.*;
7
8  import banking.*;
9  import banking.Account.CompoundResult;
10
11 public class BankGUI {
12
13         HomeGUI home;
14         AddGUI add;
15         AuthGUI auth;
16         UseGUI use;
17         CmpGUI cmp;
18         Bank bank;
19         BankClient socket;
20
21         public BankGUI(Bank bank, BankClient socket)
22         {
23                 home = new HomeGUI();
24                 add = new AddGUI();
25                 auth = new AuthGUI();
26                 use = new UseGUI();
27                 cmp = new CmpGUI();
28                 this.bank = bank;
29                 this.socket = socket;
30         }
31
32
33         public void showHome()
34         {
35                 home.show();
36         }
37
38         private class HomeGUI
39         {
40                 JFrame frame;
41                 JList<String> accountList;
42                 DefaultListModel<String> listModel;
43
44                 private void show()
45                 {
46                         frame = new JFrame();
47                         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
                        listModel = new DefaultListModel<String>();
49
50                         JButton addButton = new JButton("ADD");
51                         JButton removeButton = new JButton("REMOVE");
52                         JButton useButton = new JButton("USE");
53                         JButton cmpButton = new JButton("INTEREST");
54
55                         JPanel buttonPanel = new JPanel();
56                         JPanel centerPanel = new JPanel();
57
58                         frame.setTitle("Bank 1.0");
                        accountList = new JList<String>(listModel);
60
61                         ImageIcon addIcon = new ImageIcon(getClass().getResource("/images/client add.png'
62                         ImageIcon removeIcon = new ImageIcon(getClass().getResource("/images/client delet
63                         ImageIcon useIcon = new ImageIcon(getClass().getResource("/images/client edit.pn
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/src/BankGUI.java

```
64                      ImageIcon cmpIcon = new ImageIcon(getClass().getResource("/images/payment.png"));
65
66                      addButton.setIcon(addIcon);
67                      addButton.addActionListener(new AddListener());
68                      removeButton.setIcon(removeIcon);
69                      removeButton.addActionListener(new RemoveListener());
70                      useButton.setIcon(useIcon);
71                      useButton.addActionListener(new UseListener());
72                      cmpButton.setIcon(cmpIcon);
73                      cmpButton.addActionListener(new CmpListener());
74
75                      buttonPanel.add(addButton);
76                      buttonPanel.add(removeButton);
77                      buttonPanel.add(useButton);
78                      buttonPanel.add(cmpButton);
79
80                      JScrollPane scroller = new JScrollPane(accountList);
81                      scroller.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NE
82                      scroller.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_A
83
84                      centerPanel.add(scroller);
85                      frame.getContentPane().add(BorderLayout.NORTH, buttonPanel);
86                      frame.getContentPane().add(BorderLayout.CENTER, centerPanel);
87                      frame.setSize(500, 360);
88                      frame.setResizable(false);
89                      frame.setVisible(true);
90                      updateList();
91
92                  }
93                  public void updateList()
94                  {
95                      listModel.clear();
96                      // ArrayList<String> labels = bank.getLabels();
97
98                      ArrayList<String> accounts = socket.initAccounts();
99
100                     for(String s: accounts)
101                     {
102                         listModel.addElement(s.trim());
103                     }
104                 }
105
106                 class AddListener implements ActionListener {
107
108                     @Override
                        public void actionPerformed(ActionEvent arg0) {
110
111                         add.show();
112                     }
113                 }
114
115                 class RemoveListener implements ActionListener {
116
117                     @Override
                        public void actionPerformed(ActionEvent arg0) {
119
120                         String text = home.accountList.getSelectedValue();
121                         String holder = text.substring(0, text.indexOf('*')).trim();
122
123                         socket.removeAccount(holder);
124 //                          bank.removeAccount(holder);
125                         home.updateList();
126                     }
127
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/src/BankGUI.java

```
128             }
129             class UseListener implements ActionListener {
130
131                 @Override
                    public void actionPerformed(ActionEvent arg0) {
133
134                     auth.show();
135
136                 }
137
138             }
139             class CmpListener implements ActionListener {
140
141                 @Override
                    public void actionPerformed(ActionEvent arg0) {
143                     cmp.show();
144                 }
145             }
146     }
147
148     private class AuthGUI
149     {
150         JTextField nameField, passField;
151         JDialog frame;
152
153         private void show()
154         {
155             JPanel centerPanel = new JPanel();
156
157
158             frame = new JDialog();
                frame.setTitle("Login to Account");;
160
161             JPanel buttonPanel = new JPanel();
162
163             nameField = new JTextField(18);
164             passField = new JTextField(18);
165             JButton loginButton = new JButton("Login");
166             loginButton.addActionListener(new LoginListener());
167             JButton cancelButton = new JButton("Cancel");
168             cancelButton.addActionListener(new CancelListener());
169
170             centerPanel.add(new JLabel("Login: "));
171             centerPanel.add(nameField);
172             centerPanel.add(new JLabel("Password: "));
173             centerPanel.add(passField);
174             buttonPanel.add(loginButton);
175             buttonPanel.add(cancelButton);
176
177             frame.getContentPane().add(BorderLayout.CENTER, centerPanel);
178             frame.getContentPane().add(BorderLayout.SOUTH, buttonPanel);
179             frame.setModal(true);
180             frame.setSize(300, 130);
181             frame.setResizable(false);
182             frame.setVisible(true);
183         }
184
185         class LoginListener implements ActionListener {
186
187             @Override
                public void actionPerformed(ActionEvent arg0) {
189
190                 String name = nameField.getText();
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/src/BankGUI.java

```
191                     String pass = passField.getText();
192
193                     if(socket.authenticate(name, pass))
194                     {
195                         auth.frame.dispose();
196                         System.out.println("Name:" + name + ", Balance:"
197                             + socket.getBalance(name, pass));
198
199                         use.show(socket.getBalance(name, pass),
200                             name, pass);
201                     }
202                 }
203             }
204             class CancelListener implements ActionListener {
205
206                 @Override
                    public void actionPerformed(ActionEvent arg0) {
208                     auth.frame.dispose();
209                 }
210             }
211         }
212
213         private class UseGUI
214         {
215
216             JTextField deltaField, balField;
217             JButton witButton;
218             JDialog frame;
219             String password;
220             String holder;
221
222             private void show(double balance, String holder, String password)
223             {
224                 JPanel centerPanel = new JPanel();
225                 this.password = password;
226                 this.holder =  holder;
227
228                 frame = new JDialog();
229                 frame.setTitle("What would you like to do?");
230
231                 JPanel buttonPanel = new JPanel();
232
233                 deltaField = new JTextField(15);
234                 balField  = new JTextField(10);
235                 witButton = new JButton("Withdraw");
236                 witButton.addActionListener(new witListener());
237                 JButton depButton = new JButton("Deposit");
238                 depButton.addActionListener(new depListener());
239
240                 buttonPanel.add(witButton);
241                 buttonPanel.add(depButton);
242                 centerPanel.add(deltaField);
243                 centerPanel.add(new JLabel("Current Balance"));
244                 centerPanel.add(balField);
245
246                 balField.setText("" + balance);
247
248                 int index = bank.getIndex(holder);
249
250                 witButton.setText(socket.withdrawCheck(holder));
251
252                 frame.getContentPane().add(BorderLayout.CENTER, centerPanel);
253                 frame.getContentPane().add(BorderLayout.NORTH, buttonPanel);
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/src/BankGUI.java

```
254                     frame.setModal(true);
255                     frame.setSize(250, 130);
256                     frame.setResizable(false);
257                     frame.setVisible(true);
258                 }
259
260                 class witListener implements ActionListener {
261
262                     @Override
                    public void actionPerformed(ActionEvent arg0) {
264
265                         double amount = Double.parseDouble(deltaField.getText());
266                         amount = 0-amount;
267
268                         socket.accountAction(holder, password, amount);
269                         // bank.withdraw(amount, holder, password);
270                         balField.setText("" + socket.getBalance(holder,
271                                 password));
272
273                         witButton.setText(socket.withdrawCheck(holder));
274                     }
275                 }
276
277                 class depListener implements ActionListener {
278
279                     @Override
                    public void actionPerformed(ActionEvent arg0) {
281
282                         double amount = Double.parseDouble(deltaField.getText());
283
284                         socket.accountAction(holder, password ,amount);
285                         balField.setText("" + socket.getBalance(holder,
286                                 password));
287                     }
288                 }
289         }
290
291         private class CmpGUI
292         {
293             JDialog frame;
294             JList<String> compoundList;
295             DefaultListModel<String> listModel;
296
297             private void show()
298             {
299                 frame = new JDialog();
300                 frame.setTitle("Interest Report");
                    listModel = new DefaultListModel<String>();
302
303                 JPanel centerPanel = new JPanel();
304
                    compoundList = new JList<String>(listModel);
306
307                 JScrollPane scroller = new JScrollPane(compoundList);
308                 scroller.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NE
309                 scroller.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_A
310
311                 centerPanel.add(scroller);
312                 frame.getContentPane().add(BorderLayout.CENTER, centerPanel);
313                 frame.setSize(300, 200);
314                 frame.setResizable(false);
315                 frame.setVisible(true);
316
317                 ArrayList<String> results = socket.interest();
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/src/BankGUI.java

```
318                     for(int i =0; i < results.size(); i++)
320                     {
321                         listModel.addElement(results.get(i));
322                         //add a view of the compounding results
323                     }
324                 }
325             }
326
327             private class AddGUI
328             {
329
330                 JTextField nameField, passField, balField;
331                 JDialog frame;
332                 JRadioButton check;
333                 JRadioButton save;
334                 ButtonGroup group;
335
336                 private void show()
337                 {
338                     JPanel centerPanel = new JPanel();
339
340
341                     frame = new JDialog();
342                     frame.setTitle("Add Account");
343
344                     JPanel buttonPanel = new JPanel();
345
346                     nameField = new JTextField(18);
347                     passField = new JTextField(18);
348                     balField = new JTextField(18);
349                     JButton saveButton = new JButton("Save");
350                     saveButton.addActionListener(new SaveListener());
351                     JButton cancelButton = new JButton("Cancel");
352
353                     check = new JRadioButton();
354                     check.addActionListener(new RadioListener());
355                     save = new JRadioButton();
356                     save.addActionListener(new RadioListener());
357
358                     group = new ButtonGroup();
359
360                     group.add(check);
361                     group.add(save);
362
363                     check.setSelected(true);
364
365                     centerPanel.add(new JLabel("Name: "));
366                     centerPanel.add(nameField);
367                     centerPanel.add(new JLabel("Password: "));
368                     centerPanel.add(passField);
369                     centerPanel.add(new JLabel("Balance: "));
370                     centerPanel.add(balField);
371                     centerPanel.add(new JLabel("Checking: "));
372                     centerPanel.add(check);
373                     centerPanel.add(new JLabel("Savings: "));
374                     centerPanel.add(save);
375                     buttonPanel.add(saveButton);
376                     buttonPanel.add(cancelButton);
377
378                     cancelButton.addActionListener(new CancelListener());
379
380                     frame.getContentPane().add(BorderLayout.CENTER, centerPanel);
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/src/BankGUI.java

```
381                     frame.getContentPane().add(BorderLayout.SOUTH, buttonPanel);
382                     frame.setModal(true);
383                     frame.setSize(300, 180);
384                     frame.setResizable(false);
385                     frame.setVisible(true);
386                 }
387             }
388
389             class SaveListener implements ActionListener {
390
391                 @Override
                    public void actionPerformed(ActionEvent arg0) {
393
394                     String holder = add.nameField.getText();
395                     String pass = add.passField.getText();
396                     Double bal = Double.parseDouble(add.balField.getText());
397
398                     if(add.check.isSelected())
399                     {
400                         socket.addAccount(holder, pass, bal, 1);
401 //                          CheckingAccount acc = new CheckingAccount(bal, holder, pass);
402 //                          bank.addAccount(acc);
403                     }
404                     else
405                     {
406                         socket.addAccount(holder, pass, bal, 2);
407 //                          SavingsAccount acc = new SavingsAccount(bal, holder, pass);
408 //                          bank.addAccount(acc);
409                     }
410
411                     home.updateList();
412                     add.frame.dispose();
413                 }
414             }
415             class CancelListener implements ActionListener {
416
417                 @Override
                    public void actionPerformed(ActionEvent arg0) {
419                     add.frame.dispose();
420                 }
421
422             }
423
424             class RadioListener implements ActionListener {
425
426                 @Override
                    public void actionPerformed(ActionEvent arg0) {
428
429                 }
430             }
431 }
432
```

# Bank_Main

E:/Newsbin Downloads/Code/SSE-554-Project-3/src/Bank_Main.java

```
1  import banking.*;
2  import java.io.IOException;
3  import java.sql.SQLException;
4
5  public class Bank_Main {
6
7      public static void main(String[] args) {
8
9          Bank bank = new Bank("sse554");
10
11         BankClient socket = new BankClient();
12         socket.connect();
13
14         BankGUI main = new BankGUI(bank, socket);
15
16         main.showHome();
17     }
18
19 }
20
```

# BankServer

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/bankserver/BankServer.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package bankserver;

import java.io.BufferedWriter;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.OutputStreamWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Josh
 */
public class BankServer {

    static int portNumber = 4444;

    public static void main(String[] args) {
        ServerSocket serverSocket;
        try {
            serverSocket = new ServerSocket(portNumber);
            System.out.println("Starting the socket server:");

            getClients(serverSocket);
        }
        catch (IOException ex) {
            Logger.getLogger(BankServer.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public static void getClients(ServerSocket serverSocket)
    {
        System.out.println("Listening for clients...");

        while(true)
        {
            try
            {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Connection successful.");
                ClientThread ct = new ClientThread(clientSocket);
                ct.start();
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
        }
    }
}
```

# CheckingAccount

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/banking/CheckingAccount.java

```
1  package banking;
2
3  public class CheckingAccount extends Account {
4
5      protected double minimumBalance = 50;
6
7      public CheckingAccount(double balance, String holder, String password) {
8          super(balance, holder, password);
9
10         rate = .1;
11     }
12
13     @Override
14     protected void withdraw(double amount, String password)
15     {
16         if(!authenticate(password))
17             return;
18
19         if(amount > 0)
20             balance -= amount;
       }


       @Override
       protected CompoundResult compoundInterest()
       {
           if (balance == minimumBalance)
               return CompoundResult.NONE;

           if(balance > minimumBalance)
           {
               balance += balance * rate;
               return CompoundResult.RATE;
           }
           else
           {
               //apply 30 dollar fee
               balance -= 30;
               return CompoundResult.PENALTY;
           }
       }

       @Override
       public String toString()
       {
           return holder + " *Checking*";
       }
}
```

# ClientThread

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/bankserver/ClientThread.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package bankserver;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.net.SocketException;
import java.util.ArrayList;
import banking.*;
import banking.Account.CompoundResult;
import Encryption.*;
import java.security.Key;

/**
 *
 * @author Josh
 */
public class ClientThread extends Thread{

    Bank bank;
    Socket csocket;
    ObjectOutputStream outToClient;
    ObjectInputStream inFromClient;
    ServerDecryptor sd;
    EncryptionUtility encryptor;

    public ClientThread(Socket csocket)
    {
        this.csocket = csocket;
        bank = new Bank();
    }

    @Override
    public void run()
    {
        try {
            init();
        } catch (Exception e){
            e.printStackTrace();
        }
    }

    public void init() throws IOException, ClassNotFoundException
    {
        this.outToClient = new ObjectOutputStream(csocket.getOutputStream());
        this.inFromClient = new ObjectInputStream(csocket.getInputStream());
        this.sd = new ServerDecryptor();

        outToClient.writeObject("Hello and welcome to the Bank Socket Server!");
        outToClient.writeObject(sd.getPublic());

        sd.initCipher((byte[]) inFromClient.readObject());

        //create an encryptor and set the public key;
        encryptor = new EncryptionUtility((Key) inFromClient.readObject());
        outToClient.writeObject(encryptor.wrapSymmetricKey());
        begin();
    }

    public void begin() throws IOException, ClassNotFoundException
    {
        try {
            while (!csocket.isClosed())
            {
                Integer todo;
                todo = Integer.parseInt(sd.decrypt((byte[])inFromClient.readObject()));

                boolean complete;

                switch(todo)
                {
                    // Add a user
                    case 0: complete = add();
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/bankserver/ClientThread.java

```
                    outToClient.writeObject(encryptor.Encrypt(((Boolean)complete).toString()));
                    break;

                // Remove a user
                case 1: complete = remove();
                    outToClient.writeObject(encryptor.Encrypt(((Boolean)complete).toString()));
                    break;

                // Authenticate login
                case 2: complete = authenticate();
                    outToClient.writeObject(encryptor.Encrypt(((Boolean)complete).toString()));
                    break;

                // Withdraw or deposit
                case 3: complete = accountAction();
                    outToClient.writeObject(encryptor.Encrypt(((Boolean)complete).toString()));
                    break;

                // Interest
                case 4: complete = interest();
                    outToClient.writeObject(encryptor.Encrypt(((Boolean)complete).toString()));
                    break;

                // Reset account list
                case 5: complete = accountInit();
                    outToClient.writeObject(encryptor.Encrypt(((Boolean)complete).toString()));
                    break;

                // Get balance of account
                case 6: complete = getBalance();
                    break;

                // Get number of withdrawals left
                case 7: complete = withdrawals();
                    break;

                default: break;
            }
        }
    } catch (SocketException e)
    { System.out.println("Connection abandoned");}
}

public boolean add()
{
    try
    {
        System.out.println("Add account");
        String name = sd.decrypt((byte[])inFromClient.readObject());
        String password = sd.decrypt((byte[])inFromClient.readObject());
        String bal = sd.decrypt((byte[])inFromClient.readObject());
        Double balance = Double.parseDouble(bal);
        String acc = sd.decrypt((byte[])inFromClient.readObject());
        Integer accType = Integer.parseInt(acc);

        if(accType == 2)
            bank.addAccount(new SavingsAccount(balance, name, password));
        else
            bank.addAccount(new CheckingAccount(balance, name, password));

        System.out.println("Name: " + name + "\tPassword: " + password + "\tBalance: " + balance);
        if (accType == 2) System.out.println("Savings Account");
        else System.out.println("Checking Account");

        return true;
    } catch (IOException e) {
        System.out.println("IOException");
        return false;
    } catch (ClassNotFoundException e) {
        System.out.println("Classnotfoundexception");
        return false;
    }
}

public boolean remove()
{
    try
    {
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/bankserver/ClientThread.java

```
158             String name = sd.decrypt((byte[])inFromClient.readObject());
160             bank.removeAccount(name);
161             System.out.println("Account of " + name + " deleted.");
162
163             return true;
164         } catch (IOException e) {
165             System.out.println("IOException");
166             return false;
167         } catch (ClassNotFoundException e) {
168             System.out.println("Classnotfoundexception");
169             return false;
170         }
171     }
172
173     public boolean authenticate()
174     {
175         try
176         {
177             boolean validated = true;
178             String name = sd.decrypt((byte[])inFromClient.readObject());
179             String password = sd.decrypt((byte[])inFromClient.readObject());
180
181             bank.authenticateAccount(name, password);
182             System.out.println("Account authenticated");
183
184             return validated;
185         } catch (IOException e) {
186             System.out.println("IOException");
187             return false;
188         } catch (ClassNotFoundException e) {
189             System.out.println("Classnotfoundexception");
190             return false;
191         }
192     }
193
195     public boolean accountAction()
196     {
197         try
198         {
199             String name = sd.decrypt((byte[])inFromClient.readObject());
200             String am = sd.decrypt((byte[])inFromClient.readObject());
             Double amount = Double.parseDouble(am);
             String password = sd.decrypt((byte[])inFromClient.readObject());

             if(amount < 0)
             {
                 bank.withdraw(amount, name, password);
             }
             else
                 bank.deposit(amount, name, password);

             System.out.println("Account action performed on " + name + "'s account");

             return true;
         } catch (IOException e) {
             System.out.println("IOException");
         } catch (ClassNotFoundException e) {
             System.out.println("Classnotfoundexception");
         }
         return false;
     }

     public boolean interest()
     {
         try
         {
             System.out.println("Perform interest operation");

             ArrayList<byte[]> encryptedResults= new ArrayList<>();

             ArrayList<CompoundResult> cmp = bank.compoundAll();

             for(int i =0; i < bank.size(); i++)
             {
                 encryptedResults.add(encryptor.Encrypt(
                         bank.getHolder(i) + "  " + cmp.get(i).toString()));
             }
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/bankserver/ClientThread.java

```
            outToClient.writeObject(encryptedResults);
            return true;
        } catch (IOException e) {
            System.out.println("IOException");
        }
        return false;
    }

    public boolean accountInit() throws IOException, ClassNotFoundException
    {
        // Get the accounts from the database
        ArrayList<String> accounts = bank.getLabels();
        ArrayList<byte[]> encryptedNames = new ArrayList<>();

        for(String name: accounts)
        {
            encryptedNames.add(encryptor.Encrypt(name));
        }

        System.out.println("Reset account list.");

        outToClient.writeObject(encryptedNames);
        return true;
    }

    public boolean getBalance()
    {
        try
        {
            Double balance = 0.0;
            String name = sd.decrypt((byte[])inFromClient.readObject());
            String password = sd.decrypt((byte[])inFromClient.readObject());

            balance = bank.getBalance(name, password);
            System.out.println("Getting account balance of " + name);

            outToClient.writeObject(encryptor.Encrypt(balance.toString()));
            return true;
        } catch (IOException e) {
            System.out.println("IOException");
        } catch (ClassNotFoundException e) {
            System.out.println("Classnotfoundexception");
        }
        return false;
    }

    public boolean withdrawals()
    {
        try
        {
            // Should be in form of "Withdraw (#)" if checking or "Withdraw"
            String text = "Withdraw";
            String name = sd.decrypt((byte[])inFromClient.readObject());

            // Get balance
            int index = bank.getIndex(name);
            if (!bank.isChecking(index))
                text += " " + bank.remainingWithdrawals(index);
            System.out.println("Getting account withdrawals of " + name);

            outToClient.writeObject(encryptor.Encrypt(text));
            return true;
        } catch (IOException e) {
            System.out.println("IOException");
        } catch (ClassNotFoundException e) {
            System.out.println("Classnotfoundexception");
        }
        return false;
    }
}
```

# Database

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/banking/Database.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package banking;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

/**
 *
 * @author TSBob (Daryl Ebanks)
 */
public class Database {

    Connection link;

    public Database() throws SQLException, IOException
    {
        link = getConnection();
    }

    protected final Connection getConnection() throws SQLException, IOException
    {

       String url = "jdbc:mysql://localhost:3306/sse554";
       String username = "root";
       String password = "password";

       return DriverManager.getConnection(url, username, password);
    }

    public void writeToDatabase(ArrayList<Account> accounts) throws SQLException
    {
        /*
        Do some cool stuff here

        Remove old table
        Create new table with account's data
        */

        Statement stat = link.createStatement();
        stat.executeUpdate("DROP TABLE Bank");
        stat.executeUpdate("CREATE table Bank("
                           + "holder varchar(45),"
                           + "password varchar(45),"
                           + "balance float,"
                           + "accounttype varchar(10),"
                           + "rate float,"
                           + "withdrawals int,"
                           + "minbalance double);");

        for(Account a : accounts)
        {
            if(CheckingAccount.class == a.getClass())
            {
                stat.executeUpdate("INSERT INTO Bank ("
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/banking/Database.java

```
                    + "holder, password, balance, accounttype, rate,"
                    + "minbalance) "
                    + "VALUES ("
                    + "'" + a.holder + "'" + ","
                    + "'" + a.password + "'" + ","
                    + a.balance + ","
                    + "'Checking', "
                    + a.rate + ", "
                    + ((CheckingAccount) a).minimumBalance + ");" );
            }
            else
            {
                stat.executeUpdate("INSERT INTO Bank ("
                    + "holder, password, balance, accounttype, rate,"
                    + "withdrawals)"
                    + "VALUES ("
                    + "'" + a.holder + "'" + ","
                    + "'" + a.password + "'" + ","
                    + a.balance + ","
                    + "'Savings', "
                    + a.rate + ", "
                    + ((SavingsAccount) a).currentWithdrawals
                    + ");");
            }
        }
    }

    public ArrayList<Account> readFromDatabase(ArrayList<Account> accounts)
            throws SQLException
    {
        /*
        Do some cool stuff here too

        Select * from bank database
        Add data for each account as an insert command
        */
        Statement stat = link.createStatement();
        ResultSet res = stat.executeQuery("SELECT * from Bank");

        accounts.clear();
        while(res.next())
        {
            String holder = res.getString("holder");
            String password = res.getString("password");
            String type = res.getString("accounttype");
            Double balance = res.getDouble("balance");
            Double rate = res.getDouble("rate");
            if(type.equals("Checking"))
            {
                Double minbalance = res.getDouble("minbalance");

                CheckingAccount a = new CheckingAccount(balance, holder,
                        password);
                a.minimumBalance = minbalance;
                a.rate = rate;
                accounts.add(a);
                System.out.println(a.toString());
            }
            else
            {
                int withdrawals = res.getInt("minbalance");
                SavingsAccount a = new SavingsAccount(balance, holder,
                        password);
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/banking/Database.java

```
                password);
                a.currentWithdrawals = withdrawals;
                a.rate = rate;
                accounts.add(a);
                System.out.println(a.toString());
            }
        }
        return accounts;
    }
}
```

# SavingsAccount

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/banking/SavingsAccount.java

```
1  package banking;
2
3  public class SavingsAccount extends Account {
4
5      protected int maxWithdrawals = 5, currentWithdrawals = 0;
6
7      public SavingsAccount(double balance, String holder, String password) {
8          super(balance, holder, password);
9
10         rate = 0.05;
11     }
12
13
14     protected void withdraw(double amount, String password)
15     {
16
17         if(!authenticate(password))
18             return;
19
20         currentWithdrawals++;

           if(currentWithdrawals >= maxWithdrawals)
           {
               //apply ten dollar fee
               balance-=10;
               return;
           }

           if(amount > 0)
               balance -= amount;
       }

       protected CompoundResult compoundInterest()
       {

           if(balance >0)
           {
               balance += balance * rate;
               return CompoundResult.RATE;
           }

           if(balance < 0)
           {
               //apply 30 dollar fee
               balance -= 30;
               return CompoundResult.PENALTY;
           }

           this.currentWithdrawals = 0;

           return CompoundResult.NONE;
       }

       public String toString()
       {
           return holder + " *Savings*";
       }
   }
```

# Client Encryptor

E:/Newsbin Downloads/Code/SSE-554-Project-3/src/encryption/ClientEncryptor.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package Encryption;

import java.nio.charset.Charset;
import java.security.InvalidKeyException;
import java.security.Key;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;

/**
 *
 * @author Josh
 */
public class ClientEncryptor {

    private Cipher cipher;
    byte[] wrappedKey;
    SecretKey key;

    public ClientEncryptor(Key publicKey) {
        try {
            // Generate a random symmetric key
            System.out.println("Generate AES key");
            KeyGenerator keygen = KeyGenerator.getInstance("AES");
            SecureRandom random = new SecureRandom();
            keygen.init(random);
            key = keygen.generateKey();
            System.out.println("AES key generated");

            // Wrap the symmetric key in the public key and send
            // it back to the server
            Cipher publiccipher = Cipher.getInstance("RSA");
            publiccipher.init(Cipher.WRAP_MODE, publicKey);
            wrappedKey = publiccipher.wrap(key);
            System.out.println("Wrapped AES key");

            // Create the symmetric cipher for client side
            cipher = Cipher.getInstance("AES");
            cipher.init(Cipher.ENCRYPT_MODE, key);
            System.out.println("Client AES cipher created");
        } catch (InvalidKeyException ex) {
            Logger.getLogger(ClientEncryptor.class.getName()).log(Level.SEVERE, null, ex);
        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(ClientEncryptor.class.getName()).log(Level.SEVERE, null, ex);
        } catch (NoSuchPaddingException ex) {
            Logger.getLogger(ClientEncryptor.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IllegalBlockSizeException ex) {
            Logger.getLogger(ClientEncryptor.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public byte[] encrypt(String input)
    {
        try {
            byte[] in = input.getBytes(Charset.forName("UTF-8"));
            byte[] output = cipher.doFinal(in);

            return output;

        } catch (IllegalBlockSizeException ex) {
            Logger.getLogger(ClientEncryptor.class.getName()).log(Level.SEVERE, null, ex);
        } catch (BadPaddingException ex) {
            Logger.getLogger(ClientEncryptor.class.getName()).log(Level.SEVERE, null, ex);
        }
        return null;
    }
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/src/encryption/ClientEncryptor.java

```
80
81      public byte[] getKey()
82      {
83          return wrappedKey;
84      }
85
86      public Key getClientKey()
87      {
88          return key;
89      }
90  }
91
92
93
94
95
96
97
98
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
116
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
138
139
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
```

# Server Decrypter

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/Encryption/ServerDecryptor.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package Encryption;

import java.nio.charset.Charset;
import java.security.InvalidKeyException;
import java.security.Key;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;

/**
 *
 * @author Josh
 */
public class ServerDecryptor {

    private Cipher cipher;
    public static final int KEYSIZE = 512;
    private Key publicKey;
    private Key privateKey;
    private SecretKey key;

    public ServerDecryptor ()
    {
        try {
            // Create key pair and pull out public key
            System.out.println("Begin public key generation");
            KeyPairGenerator pairgen = KeyPairGenerator.getInstance("RSA");
            SecureRandom random = new SecureRandom();
            pairgen.initialize(KEYSIZE, random);
            KeyPair keyPair = pairgen.generateKeyPair();

            publicKey = keyPair.getPublic();
            privateKey = keyPair.getPrivate();
            System.out.println("Created public key!");
            // Send public key to client

            // Receive symmetric key from client and unwrap
            // byte[] a = new byte[8];
            // initCipher(a);

        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(ServerDecryptor.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public void initCipher(byte[] wrappedkey)
    {
        try {
            // Receive symmetric key from client and unwrap
            Cipher publiccipher = Cipher.getInstance("RSA");
            publiccipher.init(Cipher.UNWRAP_MODE, privateKey);
            key = (SecretKey) publiccipher.unwrap(wrappedkey, "AES", Cipher.SECRET_KEY);
            System.out.println("Received AES key!");

            // Create symmetric cipher from received key
            cipher = Cipher.getInstance("AES");
            cipher.init(Cipher.DECRYPT_MODE, key);
            System.out.println("Server AES cipher created");

        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(ServerDecryptor.class.getName()).log(Level.SEVERE, null, ex);
        } catch (NoSuchPaddingException ex) {
            Logger.getLogger(ServerDecryptor.class.getName()).log(Level.SEVERE, null, ex);
        } catch (InvalidKeyException ex) {
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/Encryption/ServerDecryptor.java

```
80      }
81  }
82
83  public String decrypt(byte[] input)
84  {
85      try {
86          byte[] decrypt = cipher.doFinal(input);
87          String output = new String(decrypt, Charset.forName("UTF-8"));
88
89          return output;
90      } catch (IllegalBlockSizeException ex) {
91          Logger.getLogger(ServerDecryptor.class.getName()).log(Level.SEVERE, null, ex);
92      } catch (BadPaddingException ex) {
93          Logger.getLogger(ServerDecryptor.class.getName()).log(Level.SEVERE, null, ex);
94      }
95      return null;
96  }
97
98
99  public Key getPublic()
100 {
101     return publicKey;
102 }
103
104 public Key getServerKey()
105 {
106     return key;
107 }
108 }
109
110
111
112
113
114
116
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
138
139
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
```

# Encryption Utility

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/Encryption/EncryptionUtility.java

```
1  *
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package Encryption;
7
8  import java.nio.charset.Charset;
9  import java.security.InvalidKeyException;
10 import java.security.Key;
11 import java.security.KeyPair;
12 import java.security.KeyPairGenerator;
13 import java.security.NoSuchAlgorithmException;
14 import java.security.SecureRandom;
15 import java.util.logging.Level;
16 import java.util.logging.Logger;
17 import javax.crypto.BadPaddingException;
18 import javax.crypto.Cipher;
19 import javax.crypto.IllegalBlockSizeException;
20 import javax.crypto.KeyGenerator;
21 import javax.crypto.NoSuchPaddingException;
22 import javax.crypto.SecretKey;
23
24 /**
25  *
26  * @author TSBob (Daryl Ebanks)
27  */
28 public class EncryptionUtility {
29
30     public static final int KEYSIZE = 512;
31     private Cipher cipher;
32     private SecretKey secretKey;
33     private Key publicKey;
34     private Key privateKey;
35     byte[] wrappedKey;
36
37     private Encryptor encrypt;
38     private Decrypter decrypt;
39
40
41     private class Encryptor
42     {
44         private Encryptor()
45         {
46             generateSymmetricKey();
47         }
48
49         private byte[] encrypt(String input)
50         {
51             try {
52                 cipher = Cipher.getInstance("AES");
                   cipher.init(Cipher.ENCRYPT_MODE, secretKey);
54             } catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException ex) {
55                 Logger.getLogger(EncryptionUtility.class.getName()).log(Level.SEVERE, null, ex);
56             }
57
58             byte[] output = null;
59
60             try {
61                 byte[] in = input.getBytes(Charset.forName("UTF-8"));
62                 output = cipher.doFinal(in);
63
64             } catch (IllegalBlockSizeException | BadPaddingException ex) {
65                 Logger.getLogger(EncryptionUtility.class.getName()).log(Level.SEVERE, null, ex);
66             }
67             return output;
68         }
69
70         private void generateSymmetricKey()
71         {
73             try {
74                 KeyGenerator keygen = KeyGenerator.getInstance("AES");
75                 SecureRandom random = new SecureRandom();
76                 keygen.init(random);
77                 secretKey = keygen.generateKey();
78             } catch (NoSuchAlgorithmException ex) {
                   Logger.getLogger(EncryptionUtility.class.getName()).log(Level.SEVERE, null, ex);
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/Encryption/EncryptionUtility.java

```
    }

    private byte[] wrapSymmetricKey()
    {
        try {
            Cipher pubCipher = Cipher.getInstance("RSA");
            pubCipher.init(Cipher.WRAP_MODE, publicKey);
            return pubCipher.wrap(secretKey);
        } catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException | IllegalBlockSizeException ex) {
            Logger.getLogger(EncryptionUtility.class.getName()).log(Level.SEVERE, null, ex);
        }

        return null;
    }
}

private class Decrypter
{
    private Decrypter(){
        createKeyPair();
    }

    private void createKeyPair()
    {
        try {
            KeyPairGenerator pairgen = KeyPairGenerator.getInstance("RSA");
            SecureRandom random = new SecureRandom();
            pairgen.initialize(KEYSIZE, random);
            KeyPair keyPair = pairgen.generateKeyPair();

            publicKey = keyPair.getPublic();
            privateKey = keyPair.getPrivate();
        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(EncryptionUtility.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void unwrapKey(byte [] key)
    {
        try {
            wrappedKey = key;

            Cipher publicCipher = Cipher.getInstance("RSA");
            publicCipher.init(Cipher.UNWRAP_MODE, privateKey);
            secretKey = (SecretKey) publicCipher.unwrap(wrappedKey, "AES", Cipher.SECRET_KEY);
        } catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException ex) {
            Logger.getLogger(EncryptionUtility.class.getName()).log(Level.SEVERE, null, ex);
        }

    }

    private String decrypt( byte[] input)
    {
        try {
            cipher = Cipher.getInstance("AES");
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
        } catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException ex) {
            Logger.getLogger(EncryptionUtility.class.getName()).log(Level.SEVERE, null, ex);
        }

        String output = null;

        try {
            byte[] decrypt = cipher.doFinal(input);
            output = new String(decrypt, Charset.forName("UTF-8"));

        } catch (IllegalBlockSizeException | BadPaddingException ex) {
            Logger.getLogger(EncryptionUtility.class.getName()).log(Level.SEVERE, null, ex);
        }
        return output;
    }
}

public EncryptionUtility()
{
    decrypt = new Decrypter();
}
```

E:/Newsbin Downloads/Code/SSE-554-Project-3/Server/BankServer/src/Encryption/EncryptionUtility.java

```
158     public EncryptionUtility(Key publicKey)
159     {
160         this.publicKey = publicKey;
161         encrypt = new Encryptor();
162     }
163
164     public byte[] Encrypt(String input)
165     {
166         return encrypt.encrypt(input);
167     }
168
169     public String Decrypt( byte[] input)
170     {
171         return decrypt.decrypt(input);
172     }
173
174     public byte[] wrapSymmetricKey()
175     {
176         return encrypt.wrapSymmetricKey();
177     }
178
179     public void unwrapKey(byte [] key)
180     {
181         decrypt.unwrapKey(key);
182     }
183
184     public Key getPublicKey()
185     {
186         return publicKey;
187     }
188 }
189
190
191
192
193
194
195
196
197
198
199
200
```