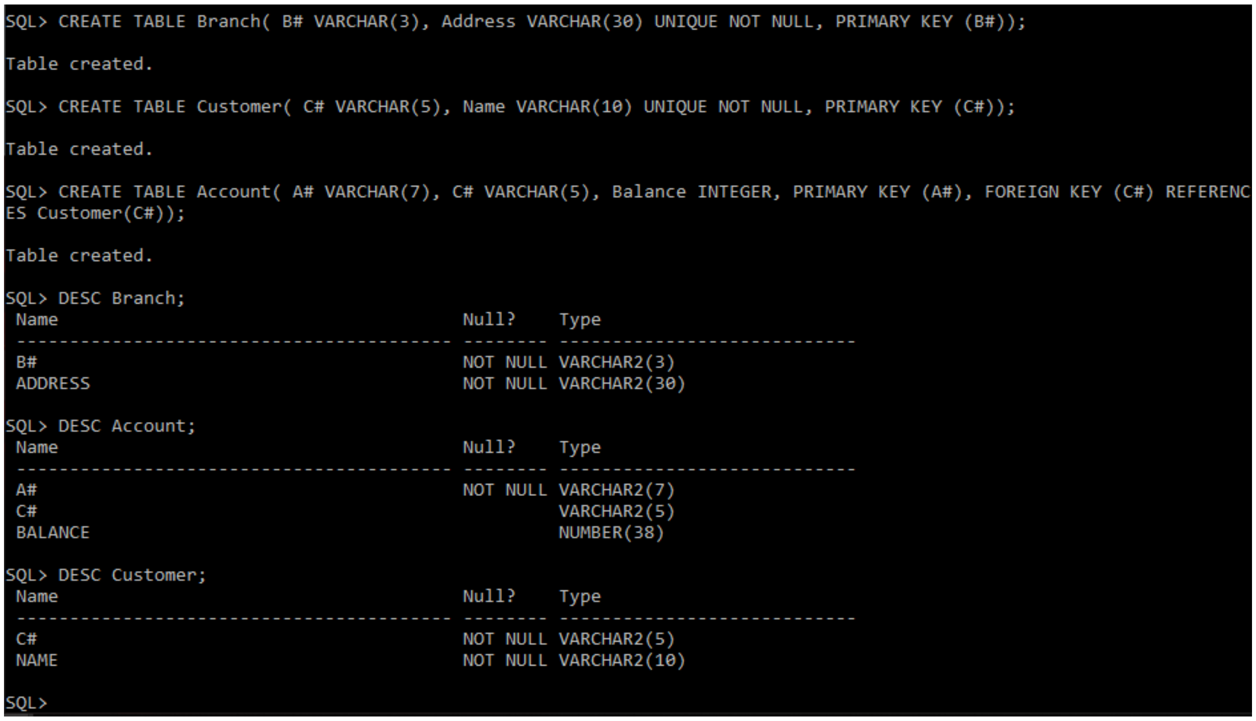
Part 1 SQL DDL

CREATE TABLE Branch(B# VARCHAR(3), Address VARCHAR(30) UNIQUE NOT NULL, PRIMARY KEY (B#));
CREATE TABLE Customer(C# VARCHAR(5), Name VARCHAR(10) UNIQUE NOT NULL, PRIMARY KEY (C#));
CREATE TABLE Account(A# VARCHAR(7), C# VARCHAR(5), Balance INTEGER, PRIMARY KEY (A#),
FOREIGN KEY (C#) REFERENCES Customer(C#));



```
SQL> CREATE TABLE Branch( B# VARCHAR(3), Address VARCHAR(30) UNIQUE NOT NULL, PRIMARY KEY (B#));

Table created.

SQL> CREATE TABLE Customer( C# VARCHAR(5), Name VARCHAR(10) UNIQUE NOT NULL, PRIMARY KEY (C#));

Table created.

SQL> CREATE TABLE Account( A# VARCHAR(7), C# VARCHAR(5), Balance INTEGER, PRIMARY KEY (A#), FOREIGN KEY (C#) REFERENC
ES Customer(C#));

Table created.

SQL> DESC Branch;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 B#                                        NOT NULL VARCHAR2(3)
 ADDRESS                                   NOT NULL VARCHAR2(30)

SQL> DESC Account;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 A#                                        NOT NULL VARCHAR2(7)
 C#                                                 VARCHAR2(5)
 BALANCE                                            NUMBER(38)

SQL> DESC Customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 C#                                        NOT NULL VARCHAR2(5)
 NAME                                      NOT NULL VARCHAR2(10)

SQL>
```

Part 2 PL/SQL

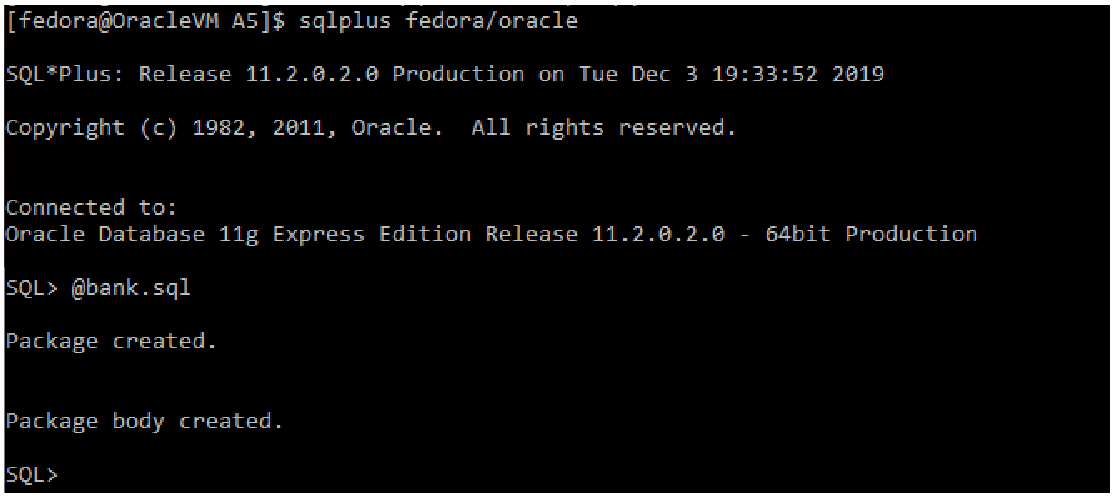
Bank.sql

```
CREATE OR REPLACE PACKAGE bank IS
 PROCEDURE get_branch (in_add IN VARCHAR2, out_Bno OUT VARCHAR2);
 PROCEDURE get_account (in_name IN VARCHAR2, in_Bno IN VARCHAR2, out_Ano OUT VARCHAR2);
 PROCEDURE open_branch (in_add IN VARCHAR2, out_Bno OUT VARCHAR2);
 PROCEDURE close_branch (in_Bno IN VARCHAR2, out_chk OUT NUMBER);
 PROCEDURE create_customer (in_name IN VARCHAR2, out_Cno OUT VARCHAR2);
 PROCEDURE remove_customer (in_name IN VARCHAR2, out_chk OUT NUMBER);
 PROCEDURE open_account (in_cust IN VARCHAR2, in_Bno IN VARCHAR2, in_amount IN INTEGER,
out_chk OUT INTEGER);
 PROCEDURE close_account (in_Ano IN VARCHAR2, out_chk OUT INTEGER);
 PROCEDURE withdraw (in_Ano IN VARCHAR2, in_amount IN NUMBER, out_chk OUT INTEGER);
 PROCEDURE deposit (in_Ano IN VARCHAR2, in_amount IN NUMBER);
 PROCEDURE show_branch (in_Bno IN VARCHAR2, out_add OUT VARCHAR2, out_cursor OUT
SYS_REFCURSOR, out_total OUT NUMBER);
 PROCEDURE show_all_branches (out_cursor OUT SYS_REFCURSOR);
 PROCEDURE show_customer (in_name IN VARCHAR2, out_chk OUT INTEGER, out_Cno OUT VARCHAR2,
out_cursor OUT SYS_REFCURSOR, out_total OUT NUMBER);
END bank;
/


CREATE OR REPLACE PACKAGE BODY bank IS
 PROCEDURE get_branch (in_add IN VARCHAR2, out_Bno OUT VARCHAR2) IS
  v_Bno VARCHAR2(5);
 BEGIN
  SELECT B# INTO v_Bno FROM Branch WHERE Address = in_add OR B# = in_add;
  out_Bno := v_Bno;
  EXCEPTION WHEN NO_DATA_FOUND THEN
   out_Bno := NULL;
 END get_branch;


 PROCEDURE get_account (in_name IN VARCHAR2, in_Bno IN VARCHAR2, out_Ano OUT VARCHAR2) IS
  v_Ano VARCHAR2(10);
 BEGIN
  SELECT a.A# INTO v_Ano FROM Account a, Customer c WHERE a.A# LIKE CONCAT(in_Bno, '%') AND
a.C# = c.C# AND Name = in_name;
  out_Ano := v_Ano;
  EXCEPTION WHEN NO_DATA_FOUND THEN
   out_Ano := NULL;
 END get_account;


 PROCEDURE open_branch (in_add IN VARCHAR2, out_Bno OUT VARCHAR2) IS
  num NUMBER := 0;
  v_count NUMBER;
  v_Bno VARCHAR2(5);
  cursor c1 is SELECT * FROM Branch ORDER BY B#;
 BEGIN
  SELECT COUNT(*) INTO v_count FROM Branch WHERE Address = in_add;
  IF v_count > 0 THEN
   out_Bno := NULL;
   RETURN;
  END IF;
  FOR item IN c1
  LOOP
```

```
   EXIT WHEN TO_NUMBER(item.B#, '000') != num;
   DBMS_OUTPUT.PUT_LINE(TO_NUMBER(item.B#, '000'));
   DBMS_OUTPUT.PUT_LINE(item.Address);
   num := num + 1;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE(num);
  v_Bno := lpad(TO_CHAR(num),3, '0');
  INSERT INTO Branch VALUES(v_Bno, in_add);
  out_Bno := v_Bno;
END open_branch;



 PROCEDURE close_branch (in_Bno IN VARCHAR2, out_chk OUT NUMBER) IS
  v_count NUMBER;
  v_Ano VARCHAR2(5);
 BEGIN
  v_Ano := CONCAT(in_Bno, '%');
  SELECT COUNT(*) INTO v_count FROM Account WHERE A# LIKE v_Ano;
  IF v_count > 0 THEN
   out_chk := 0;
   RETURN;
  END IF;
  DELETE FROM Branch WHERE B# = in_Bno;
  out_chk := 1;
 END close_branch;



 PROCEDURE create_customer (in_name IN VARCHAR2, out_Cno OUT VARCHAR2) IS
  num NUMBER := 0;
  v_Cno VARCHAR2(5);
  v_count NUMBER;
  v_max VARCHAR2(5);
 BEGIN
  SELECT COUNT(*) INTO v_count FROM Customer WHERE Name = in_name;
  IF v_count > 0 THEN
   out_Cno := NULL;
   RETURN;
  END IF;
  SELECT COUNT(*) INTO v_count FROM Customer ;
  IF v_count < 1 THEN
   out_Cno := '00000';
   INSERT INTO Customer VALUES(out_Cno, in_name);
   RETURN;
  END IF;
  SELECT MAX(C#) into v_max FROM Customer;
  v_Cno := lpad(TO_CHAR(TO_NUMBER(v_max, '00000') + 1),5, '0');
  INSERT INTO Customer VALUES(v_Cno, in_name);
  out_Cno := v_Cno;
 END create_customer;



 PROCEDURE remove_customer (in_name IN VARCHAR2, out_chk OUT NUMBER) IS
  v_count_C NUMBER;
  v_count_A NUMBER;
  v_Cno VARCHAR2(5);
 BEGIN
  SELECT COUNT(*) INTO v_count_C FROM Customer WHERE Name = in_name;
  IF v_count_C < 1 THEN
   out_chk := 0;
   DBMS_OUTPUT.PUT_LINE('No customer exists with this name');
   RETURN;
  END IF;
  SELECT C# INTO v_Cno FROM Customer WHERE Name = in_name;
  DBMS_OUTPUT.PUT_LINE(in_name);
  DBMS_OUTPUT.PUT_LINE(v_Cno);
  SELECT COUNT(*) INTO v_count_A FROM Account WHERE C# = v_Cno;
  IF v_count_A > 0 THEN
   DBMS_OUTPUT.PUT_LINE('Account exists for this customer');
   out_chk := 1;
   RETURN;
  END IF;
  DELETE FROM Customer WHERE C# = v_Cno;
  out_chk := 2;
 END remove_customer;



 PROCEDURE open_account (in_cust IN VARCHAR2, in_Bno IN VARCHAR2, in_amount IN INTEGER,
out_chk OUT INTEGER) IS
  num NUMBER;
  v_count_C NUMBER;
  v_count_A NUMBER;
  v_Cno VARCHAR2(5);
  v_Ano VARCHAR2(10);
  cursor c1 is SELECT * FROM Account WHERE A# LIKE CONCAT(in_Bno, '%') ORDER BY A#;
```

```
BEGIN
 SELECT COUNT(*) INTO v_count_C FROM Customer WHERE Name = in_cust;
 IF v_count_C < 1 THEN
  out_chk := 0;
  DBMS_OUTPUT.PUT_LINE('No customer exists with this name');
  RETURN;
 END IF;
 SELECT C# INTO v_Cno FROM Customer WHERE Name = in_cust;
 DBMS_OUTPUT.PUT_LINE(v_Cno);
 v_Ano := CONCAT(in_Bno, '%');
 SELECT COUNT(*) INTO v_count_A FROM Account WHERE C# = v_Cno AND A# LIKE v_Ano;
 IF v_count_A > 0 THEN
  out_chk := 1;
  DBMS_OUTPUT.PUT_LINE('Account at this branch and for this customer already exists.');
  RETURN;
 END IF;
 num := 0;
 FOR item IN c1
 LOOP
  EXIT WHEN TO_NUMBER(SUBSTR(item.A#, 4,4), '0000') != num;
  DBMS_OUTPUT.PUT_LINE(item.A#);
  DBMS_OUTPUT.PUT_LINE(TO_NUMBER(SUBSTR(item.A#, 4,4), '0000'));
  DBMS_OUTPUT.PUT_LINE(item.C#);
  num := num + 1;
 END LOOP;
 DBMS_OUTPUT.PUT_LINE(num);
 v_Ano := CONCAT(in_Bno, lpad(TO_CHAR(num),4,'0'));
 DBMS_OUTPUT.PUT_LINE(v_Ano);
 INSERT INTO Account VALUES(v_Ano, v_Cno, in_amount);
 out_chk := 2;
END open_account;

PROCEDURE close_account (in_Ano IN VARCHAR2, out_chk OUT INTEGER) IS
 v_count NUMBER;
BEGIN
 SELECT COUNT(*) INTO v_count FROM Account WHERE A# = in_Ano;
 IF v_count < 1 THEN
  out_chk := 0;
  DBMS_OUTPUT.PUT_LINE('Account does not exist');
  RETURN;
 END IF;
 SELECT COUNT(*) INTO v_count FROM Account WHERE A# = in_Ano AND Balance = 0;
 IF v_count < 1 THEN
  out_chk := 1;
  DBMS_OUTPUT.PUT_LINE('Account balance not 0');
  RETURN;
 END IF;
 DELETE FROM Account WHERE A# = in_Ano;
 out_chk := 2;
END close_account;


PROCEDURE withdraw (in_Ano IN VARCHAR2, in_amount IN NUMBER, out_chk OUT INTEGER) IS
 v_count NUMBER;
BEGIN
 SELECT COUNT(*) INTO v_count FROM Account WHERE A# = in_Ano AND Balance >= in_amount;
 IF v_count < 1 THEN
  out_chk := 0;
  DBMS_OUTPUT.PUT_LINE('Account does not have enough money.');
  RETURN;
 END IF;
 UPDATE Account SET Balance = ((SELECT Balance FROM Account WHERE A# = in_Ano) - in_amount)
WHERE A# = in_Ano;
 out_chk := 1;
END withdraw;



PROCEDURE deposit (in_Ano IN VARCHAR2, in_amount IN NUMBER) IS
BEGIN
 UPDATE Account SET Balance = ((SELECT Balance FROM Account WHERE A# = in_Ano) + in_amount)
WHERE A# = in_Ano;
END deposit;


PROCEDURE show_branch (in_Bno IN VARCHAR2, out_add OUT VARCHAR2, out_cursor OUT
SYS_REFCURSOR, out_total OUT NUMBER) IS
 v_add VARCHAR2(30);
 v_total NUMBER;
BEGIN
 SELECT Address INTO v_add FROM Branch WHERE B# = in_Bno;
 SELECT SUM(Balance) INTO v_total FROM Account WHERE A# LIKE CONCAT(in_Bno, '%');
 OPEN out_cursor FOR SELECT a.A#, a.C#, c.Name, a.Balance FROM Account a, Customer c where a.A#
LIKE CONCAT(in_Bno, '%') AND a.C# = c.C# ORDER BY a.A#;
 out_add := v_add;
 out_total := v_total;
END show_branch;
```

```
 PROCEDURE show_all_branches (out_cursor OUT SYS_REFCURSOR) IS
 BEGIN
  OPEN out_cursor FOR SELECT B# FROM Branch ORDER BY B#;
 END show_all_branches;




 PROCEDURE show_customer (in_name IN VARCHAR2, out_chk OUT INTEGER, out_Cno OUT VARCHAR2,
out_cursor OUT SYS_REFCURSOR, out_total OUT NUMBER) IS
  v_Cno VARCHAR2(5);
  v_count NUMBER;
  v_total NUMBER;
 BEGIN
  SELECT COUNT(*) INTO v_count FROM Customer WHERE Name = in_name;
  IF v_count < 1 THEN
   out_chk := 0;
   DBMS_OUTPUT.PUT_LINE('Customer doesnt exist');
   RETURN;
  END IF;
  SELECT C# INTO v_Cno FROM Customer WHERE Name = in_name;
  OPEN out_cursor FOR SELECT b.Address, a.A#, a.Balance FROM Account a, Branch b WHERE b.B# =
SUBSTR(a.A#, 1,3) AND C# = v_Cno;
  SELECT SUM(Balance) INTO v_total FROM Account WHERE C# = v_Cno;
  out_chk := 1;
  out_Cno := v_Cno;
  out_total := v_total;
 END show_customer;

END;
/
```



```
[fedora@OracleVM A5]$ sqlplus fedora/oracle

SQL*Plus: Release 11.2.0.2.0 Production on Tue Dec 3 19:33:52 2019

Copyright (c) 1982, 2011, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL> @bank.sql

Package created.


Package body created.

SQL>
```

Part 3 JDBC Programming

JDBCbank.java

```
import java.sql.*;
import java.io.*;
import oracle.jdbc.*;
import oracle.sql.*;
import java.util.*;
import java.util.Scanner;

public class JDBCbank
{

            public static String get_branch(String add){
                        String Bno = null;
                        try{
                    DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
                    Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","fedora",
"oracle");
                    CallableStatement cs = conn.prepareCall("{call bank.get_branch(?,?)}");
                    cs.setString(1, add);
                    cs.registerOutParameter(2, Types.VARCHAR);
                                cs.executeUpdate();
                    Bno = cs.getString(2);
                                if (Bno != null) {
                                            System.out.println("Branch number: "+Bno);
                                }
                                else{
                                            System.out.println("Error: branch with address "+add+" does not
exist.");
                                }
                                cs.close();
                    conn.close();
                        }
             catch(Exception e){
                    System.out.println("SQL exception: ");
                    e.printStackTrace();
                    System.exit(-1);
             }
                        return Bno;
      }
```

```
            public static String get_account(String name, String branch){
             String Ano = null;
             try{
                    DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
                    Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","fedora",
"oracle");
                   String Bno = get_branch(branch);
                    if (Bno == null) {
                           return null;
                    }
                              CallableStatement cs = conn.prepareCall("{call bank.get_account(?,?,?)}");
                    cs.setString(1, name);
                              cs.setString(2, Bno);
                    cs.registerOutParameter(3, Types.VARCHAR);
                    cs.executeUpdate();
                    Ano = cs.getString(3);
                    if (Ano != null) {
                           System.out.println("Account number: "+Ano);
                    }
                    else{
                           System.out.println("Error: customer or account does not exist.");
                    }
                    cs.close();
                    conn.close();
             }
             catch(Exception e){
                    System.out.println("SQL exception: ");
                    e.printStackTrace();
                    System.exit(-1);
             }
             return Ano;
      }

      public static void open_branch(String add){
             try{
                    DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
                    Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","fedora","oracle");
                              CallableStatement cs = conn.prepareCall("{call bank.open_branch(?,?)}");
                              cs.setString(1, add);
                              cs.registerOutParameter(2, Types.VARCHAR);
                              cs.executeUpdate();
                              String Bno = cs.getString(2);
```

```
                        if (Bno != null) {
                System.out.println("Created new branch with B#: "+Bno+" and address: "+add);
            }
            else{
                System.out.println("Error branch with address "+add+" already exists.");
            }
            cs.close();
            conn.close();
        }
        catch(Exception e){
            System.out.println("SQL exception: ");
            e.printStackTrace();
            System.exit(-1);
        }
    }

       public static void close_branch(String branch){
        try{
                        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
            Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","fedora",
"oracle");
            String Bno = get_branch(branch);
                        if (Bno == null) {
                                return;
                        }
                        CallableStatement cs = conn.prepareCall("{call bank.close_branch(?,?)}");
                        cs.setString(1, Bno);
                        cs.registerOutParameter(2, Types.INTEGER);
            cs.executeUpdate();
                        int chk = cs.getInt(2);
                        if (chk == 0){
                                System.out.println("Error: Branch still has open accounts.");
                        }
                        else{
                                System.out.println("Closed branch.");
                        }
            cs.close();
            conn.close();
        }
        catch(Exception e){
            System.out.println("SQL exception: ");
            e.printStackTrace();
            System.exit(-1);
            }
        }

            public static void create_customer(String name){
             try{
                    DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
                    Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","fedora",
"oracle");
                    CallableStatement cs = conn.prepareCall("{call bank.create_customer(?,?)}");
                    cs.setString(1, name);
                    cs.registerOutParameter(2, Types.VARCHAR);
                    cs.executeUpdate();
                    String Cno = cs.getString(2);
                    if (Cno != null) {
                          System.out.println("Created new customer with C#: "+Cno+" and name: "+name);
                    }
                    else{
                          System.out.println("Error customer with name "+name+" already exists.");
                    }
                    cs.close();
                    conn.close();
             }
             catch(Exception e){
                    System.out.println("SQL exception: ");
                    e.printStackTrace();
                    System.exit(-1);
             }
        }

            public static void remove_customer(String name){
             try{
                    DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
                    Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","fedora",
"oracle");
                    CallableStatement cs = conn.prepareCall("{call bank.remove_customer(?,?)}");
                    cs.setString(1, name);
                    cs.registerOutParameter(2, Types.INTEGER);
                    cs.executeUpdate();
                    int chk = cs.getInt(2);
                    if (chk == 1){
                          System.out.println("Error: Customer still has open accounts.");
                    }
```

```
                        else if (chk == 0){
                                System.out.println("Error: Customer does not exist.");
                }
                        else{
                    System.out.println("Removed customer.");
                }
                cs.close();
                conn.close();
            }
            catch(Exception e){
                System.out.println("SQL exception: ");
                e.printStackTrace();
                System.exit(-1);
            }
        }

            public static void open_account(String name, String branch, int amount){
                    try{
                DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
                Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","fedora",
"oracle");
                String Bno = get_branch(branch);
                        if (amount < 0) {
                                System.out.println("Error: Balance below 0.");
                                return;
                        }
                        if (Bno == null) {
                    return;
                }
                        CallableStatement cs = conn.prepareCall("{call bank.open_account(?,?,?,?)}");
                        cs.setString(1, name);
                        cs.setString(2, Bno);
                        cs.setInt(3, amount);
                        cs.registerOutParameter(4, Types.INTEGER);
                        cs.executeUpdate();
                        int chk = cs.getInt(4);
                        if (chk == 0){
                    System.out.println("Error: No customer with that name exists.");
                }
                        else if (chk == 1){
                    System.out.println("Error: Customer already has an account at that branch.");
                }
                        else{
```

```
                System.out.println("Opened a new account for "+name+".");
            }
            cs.close();
            conn.close();
        }
            catch(Exception e){
            System.out.println("SQL exception: ");
            e.printStackTrace();
            System.exit(-1);
        }
    }

        public static void close_account(String name, String branch){
            try{
            DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
            Connection conn = 
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","fedora", 
"oracle");
                String Ano = get_account(name, branch);
                if (Ano == null){
                    return;
                }
                CallableStatement cs = conn.prepareCall("{call bank.close_account(?,?)}");
                cs.setString(1,Ano);
                cs.registerOutParameter(2, Types.INTEGER);
                cs.executeUpdate();
                int chk = cs.getInt(2);
                if (chk == 0){
                System.out.println("Error: Account does not exist.");
            }
                else if (chk == 1){
                System.out.println("Error: Account does not have a balance of 0.");
            }
                else{
                System.out.println("Closed "+name+"'s account at branch "+branch+".");
            }
            cs.close();
            conn.close();
        }
        catch(Exception e){
            System.out.println("SQL exception: ");
            e.printStackTrace();
            System.exit(-1);
        }
    }


        public static boolean withdraw(String branch, String name, int amount){
                boolean result = false;
         try{
              DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
              Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","fedora",
"oracle");
                        String Bno = get_branch(branch);
              if (Bno == null) {
                   return result;
              }
                        String Ano = get_account(name, branch);
              if (Ano == null){
                   return result;
              }
              CallableStatement cs = conn.prepareCall("{call bank.withdraw(?,?,?)}");
              cs.setString(1,Ano);
              cs.setInt(2,amount);
              cs.registerOutParameter(3, Types.INTEGER);
              cs.executeUpdate();
              int chk = cs.getInt(3);
              if (chk == 0){
                                System.out.println("Error: Not enough money in account.");
                        }
              else{
                                result = true;
                   System.out.println("Withdrew $"+Integer.toString(amount)+" from "+name+"'s
"+branch+" account.");
              }
              cs.close();
              conn.close();
         }
         catch(Exception e){
              System.out.println("SQL exception: ");
              e.printStackTrace();
              System.exit(-1);
         }
                return result;
    }
```

```
        public static boolean deposit(String branch, String name, int amount){
                boolean result = false;
                if (amount < 0){
                        System.out.println("Error: amount to deposit is less than 0.");
                        return result;
                }
                try{

              DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
              Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","fedora","oracle");
              String Bno = get_branch(branch);
              if (Bno == null) {
                  return result;
              }
                        String Ano = get_account(name, branch);
              if (Ano == null){
                  return result;
              }
                        CallableStatement cs = conn.prepareCall("{call bank.deposit(?,?)}");
              cs.setString(1,Ano);
                        cs.setInt(2,amount);
                        cs.executeUpdate();
                        result = true;
              System.out.println("Deposited $"+Integer.toString(amount)+" from "+name+"'s
"+branch+" account.");
                        cs.close();
              conn.close();
          }
          catch(Exception e){
              System.out.println("SQL exception: ");
              e.printStackTrace();
              System.exit(-1);
          }
                return result;
      }


        public static void transfer(String branchW, String nameW, String branchD, String nameD, int
amount){
                if (amount < 0){
              System.out.println("Error: amount to deposit is less than 0.");
              return;
          }
```

```
             if(withdraw(branchW, nameW, amount)){
                                  if(deposit(branchD, nameD, amount)){
                                            System.out.println("Transfer from "+nameW+"'s "+branchW+" account
to "+nameD+"'s "+branchD+" account successful.");
                                  }
                                  else{
                                            deposit(branchW, nameW, amount);
                                  }
                       }
            }


            public static void show_branch(String branch){
             try{
                                  System.out.println("");
                                  String Bno = get_branch(branch);
                   if (Bno == null) {
                        return;
                   }
                   DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
                   Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","fedora","oracle");
                   CallableStatement cs = conn.prepareCall("{call bank.show_branch(?,?,?,?)}");
                                  cs.setString(1,Bno);
                                  cs.registerOutParameter(2, Types.VARCHAR);
                   cs.registerOutParameter(3, OracleTypes.CURSOR);
                                  cs.registerOutParameter(4, Types.INTEGER);
                   cs.executeUpdate();
                                  String add;
                                  add = cs.getString(2);
                                  System.out.println("with Address: "+add);
                                  System.out.println("  A#   |  C#  | Name |Balance");
                                  System.out.println("------- ------ ------ -------");
                   ResultSet rs = (ResultSet) cs.getObject(3);
                                  while (rs.next()) {
                                            System.out.print(rs.getString("A#")+" ");
                                            System.out.print(rs.getString("C#")+"  ");
                                            System.out.print(rs.getString("Name")+"   ");
                                            System.out.print("$"+rs.getString("Balance")+"\n");
                                  }
                                  String total;
                                  total = Integer.toString(cs.getInt(4));
                                  System.out.println("The branch total is $"+total);
                                  System.out.println("");
                        rs.close();
                cs.close();
                conn.close();
            }
            catch(Exception e){
                System.out.println("SQL exception: ");
                e.printStackTrace();
                System.exit(-1);
            }
        }

            public static void show_all_branches(){
                  try{
                DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
                Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","fedora","oracle");
                CallableStatement cs = conn.prepareCall("{call bank.show_all_branches(?)}");
                        cs.registerOutParameter(1, OracleTypes.CURSOR);
                        cs.executeUpdate();
                        ResultSet rs = (ResultSet) cs.getObject(1);
                        while (rs.next()) {
                              show_branch(rs.getString("B#"));
                        }
                        rs.close();
                cs.close();
                conn.close();
            }
            catch(Exception e){
                System.out.println("SQL exception: ");
                e.printStackTrace();
                System.exit(-1);
            }
        }

            public static void show_customer(String name){
                  try{
                DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
                Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","fedora",
"oracle");
                CallableStatement cs = conn.prepareCall("{call bank.show_customer(?,?,?,?,?)}");
                        cs.setString(1,name);
                cs.registerOutParameter(2, Types.INTEGER);
                        cs.registerOutParameter(3, Types.VARCHAR);
                cs.registerOutParameter(4, OracleTypes.CURSOR);
                        cs.registerOutParameter(5, Types.INTEGER);
                cs.executeUpdate();
                        int chk = cs.getInt(2);
                if (chk == 0){
                    System.out.println("Error: customer does not exist");
                              return;
                }
                String Cno;
                Cno = cs.getString(3);
                        System.out.print("Customer "+name);
                System.out.println(" with C#: "+Cno);
                System.out.println("Addres      |  A#   | Balance");
                System.out.println("------------ ------- --------");
                ResultSet rs = (ResultSet) cs.getObject(4);
                while (rs.next()) {
                    System.out.print(rs.getString("Address")+"       ");
                    System.out.print(rs.getString("A#")+" ");
                    System.out.print("$"+rs.getString("Balance")+"\n");
                }
                String total;
                total = Integer.toString(cs.getInt(5));
                System.out.println("The branch total is $"+total);
                System.out.println("");
                rs.close();
                cs.close();
                conn.close();
            }
            catch(Exception e){
                System.out.println("SQL exception: ");
                e.printStackTrace();
                System.exit(-1);
            }
        }


            public static void main(String[] args){
            int choice = -1;
            String line = "";
            Scanner scan = new Scanner(System.in);
            while(choice != 0){
                System.out.print("Please choose a number from the following options:\n");
                System.out.print("0) Exit\n");
                        System.out.print("1) Get branch B#\n");
```

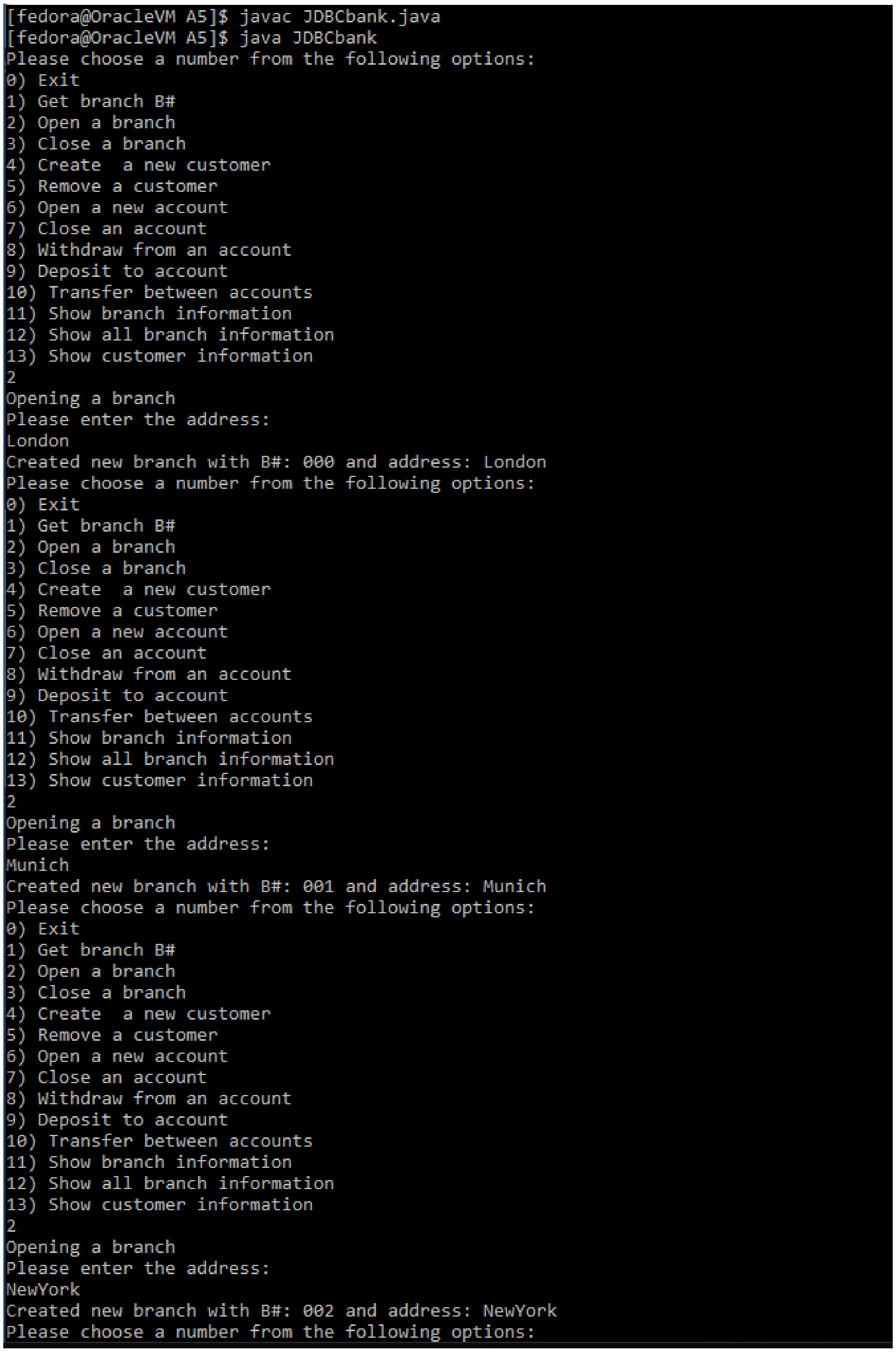
```
System.out.print("2) Open a branch\n");
System.out.print("3) Close a branch\n");
            System.out.print("4) Create  a new customer\n");
System.out.print("5) Remove a customer\n");
System.out.print("6) Open a new account\n");
            System.out.print("7) Close an account\n");
            System.out.print("8) Withdraw from an account\n");
System.out.print("9) Deposit to account\n");
System.out.print("10) Transfer between accounts\n");
System.out.print("11) Show branch information\n");
System.out.print("12) Show all branch information\n");
System.out.print("13) Show customer information\n");
choice = scan.nextInt();
if(choice == 0){
     System.out.println("Thank you. Goodbye.");
}
else if(choice == 1){
                    System.out.print("Getting a branch B#\n");
     System.out.print("Please enter the address:\n");
     String add = scan.next();
     scan.nextLine();
     get_branch(add);
}
            else if(choice == 2){
                    System.out.print("Opening a branch\n");
            System.out.print("Please enter the address:\n");
            String add = scan.next();
            scan.nextLine();
     open_branch(add);
}
            else if(choice == 3){
     System.out.print("Closing a branch\n");
     System.out.print("Please enter the address or branch number:\n");
     String branch = scan.next();
     scan.nextLine();
     close_branch(branch);
}
            else if(choice == 4){
     System.out.print("Creating a new customer\n");
     System.out.print("Please enter the new customer name:\n");
     String name = scan.next();
     scan.nextLine();
     create_customer(name);
}
```

```
          else if(choice == 5){
    System.out.print("Removing a customer\n");
    System.out.print("Please enter the customer name:\n");
    String name = scan.next();
    scan.nextLine();
    remove_customer(name);
}
          else if(choice == 6){
    System.out.print("Opening a new account\n");
    System.out.print("Please enter the customer name:\n");
    String name = scan.next();
    scan.nextLine();
                  System.out.print("Please enter branch number or address:\n");
                  String branch = scan.next();
    scan.nextLine();
                  System.out.print("Please enter starting balance\n");
                  String amount = scan.next();
    scan.nextLine();
    open_account(name, branch, Integer.parseInt(amount));
}
          else if(choice == 7){
    System.out.print("Closing an account\n");
    System.out.print("Please enter the customer name:\n");
    String name = scan.next();
    scan.nextLine();
                  System.out.print("Please enter the branch number or address:\n");
    String branch = scan.next();
    scan.nextLine();
    close_account(name, branch);
}
          else if(choice == 8){
    System.out.print("Withdrawing money.\n");
    System.out.print("Please enter the customer name:\n");
    String name = scan.next();
    scan.nextLine();
                  System.out.print("Please enter the branch number or address:\n");
    String branch = scan.next();
    scan.nextLine();
    System.out.print("Please enter the amount to withdraw\n");
    String amount = scan.next();
    scan.nextLine();
    withdraw(branch, name, Integer.parseInt(amount));
}
          else if(choice == 9){
```

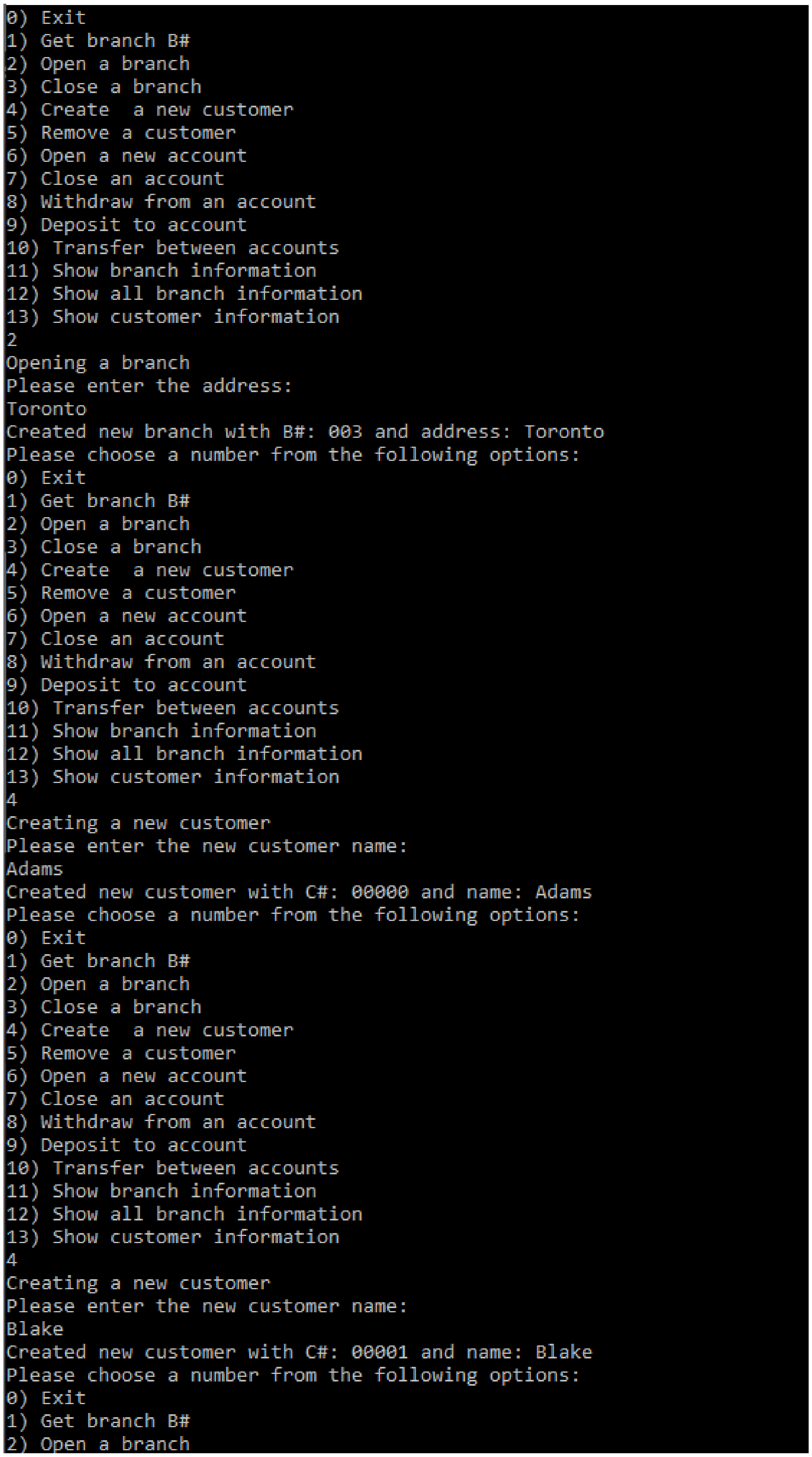
```
        System.out.print("Depositing money.\n");
                    System.out.print("Please enter the customer name:\n");
        String name = scan.next();
        scan.nextLine();
        System.out.print("Please enter the branch number or address:\n");
        String branch = scan.next();
        scan.nextLine();
                    System.out.print("Please enter the amount to deposit\n");
                    String amount = scan.next();
        scan.nextLine();
                    deposit(branch, name, Integer.parseInt(amount));
                }
                else if(choice == 10){
        System.out.print("Transferring money.\n");
        System.out.print("Please enter the customer name to withdraw from:\n");
        String nameW = scan.next();
        scan.nextLine();
        System.out.print("Please enter the branch number or address to withdraw from:\n");
        String branchW = scan.next();
        scan.nextLine();
                    System.out.print("Please enter the customer name to deposit to:\n");
        String nameD = scan.next();
        scan.nextLine();
        System.out.print("Please enter the branch number or address to deposit to:\n");
        String branchD = scan.next();
        scan.nextLine();
        System.out.print("Please enter the amount to transfer\n");
        String amount = scan.next();
        scan.nextLine();
        transfer(branchW, nameW, branchD, nameD, Integer.parseInt(amount));
}
                else if(choice == 11){
                    System.out.print("Please enter a branch number or address:\n");
                    String branch = scan.next();
        scan.nextLine();
                    show_branch(branch);
                }
                else if(choice == 12){
                    show_all_branches();
                }
                else if(choice == 13){
                    System.out.print("Please enter the customer name:\n");
                    String name = scan.next();
        scan.nextLine();
```

```
                     show_customer(name);
                             }
                    }
                    scan.close();
          }
}
```

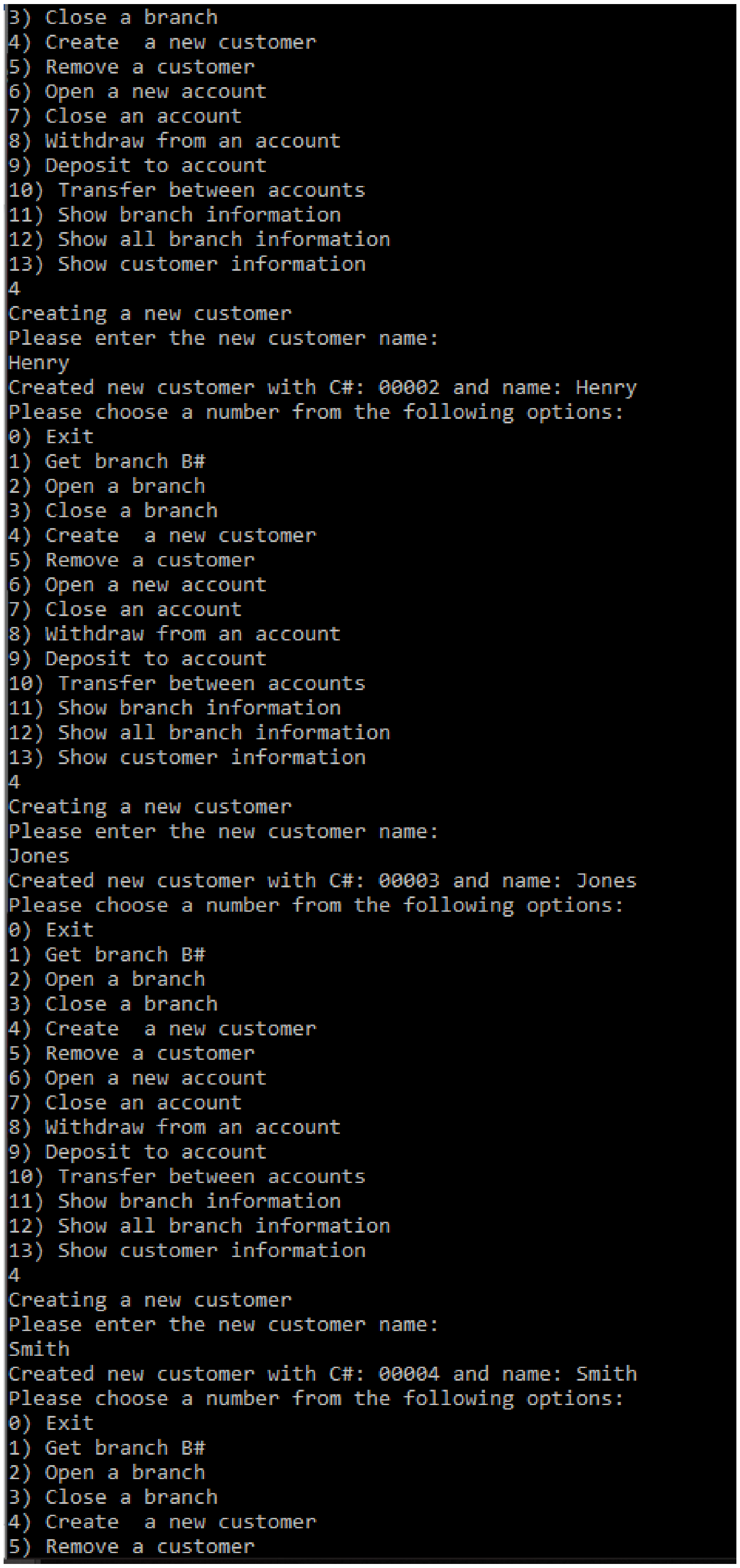
Part 4 JDBC Database Populating and Testing



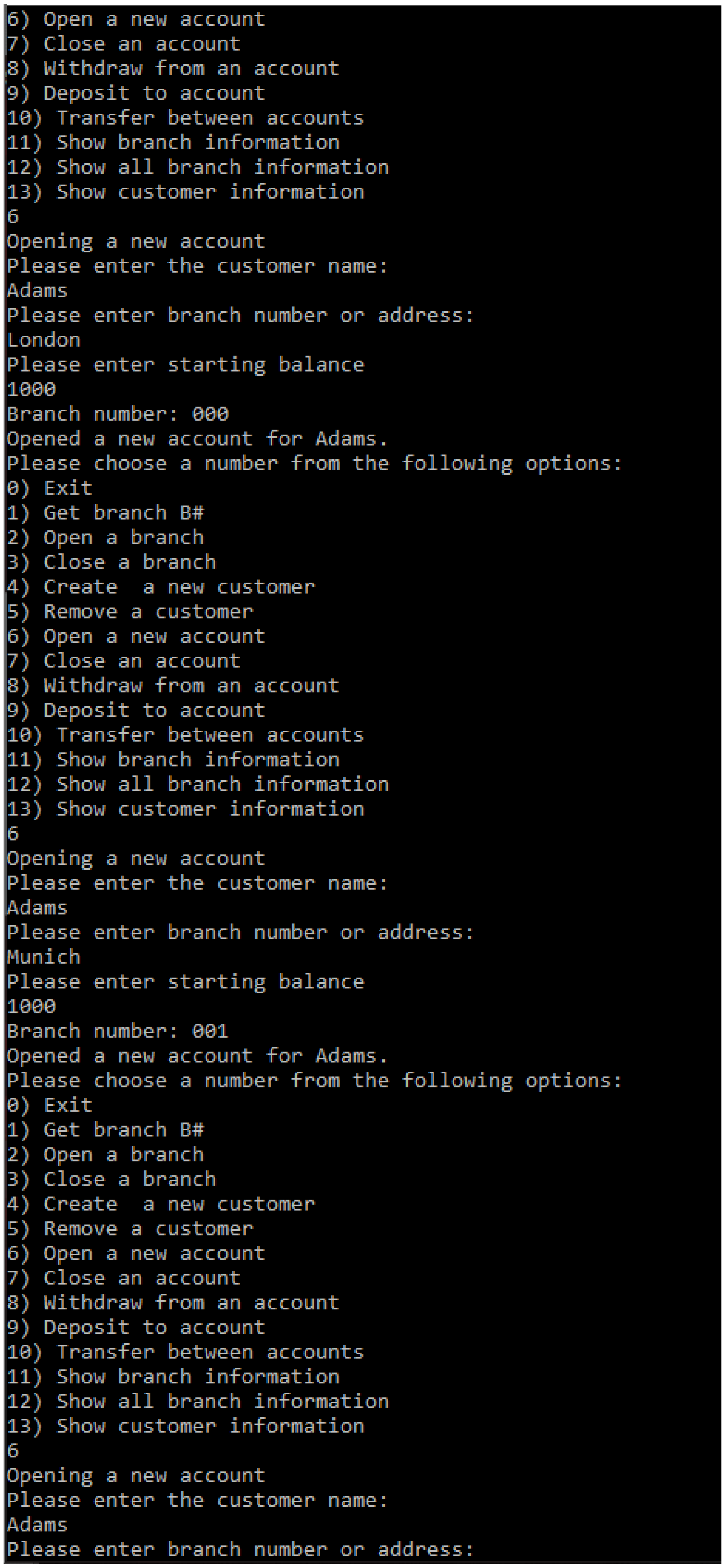
```
[fedora@OracleVM A5]$ javac JDBCbank.java
[fedora@OracleVM A5]$ java JDBCbank
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
2
Opening a branch
Please enter the address:
London
Created new branch with B#: 000 and address: London
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
2
Opening a branch
Please enter the address:
Munich
Created new branch with B#: 001 and address: Munich
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
2
Opening a branch
Please enter the address:
NewYork
Created new branch with B#: 002 and address: NewYork
Please choose a number from the following options:
```



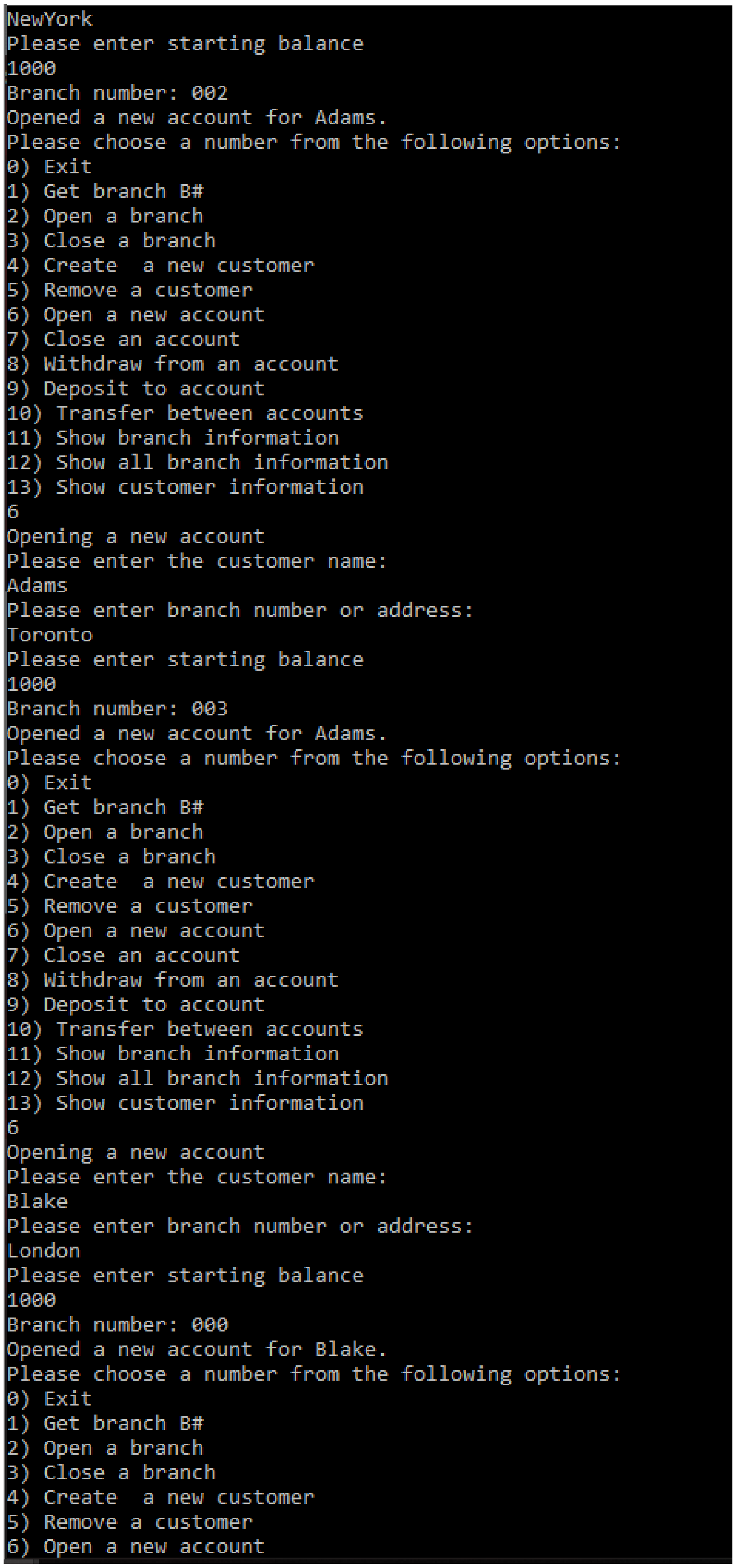
```
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
2
Opening a branch
Please enter the address:
Toronto
Created new branch with B#: 003 and address: Toronto
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
4
Creating a new customer
Please enter the new customer name:
Adams
Created new customer with C#: 00000 and name: Adams
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
4
Creating a new customer
Please enter the new customer name:
Blake
Created new customer with C#: 00001 and name: Blake
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
```

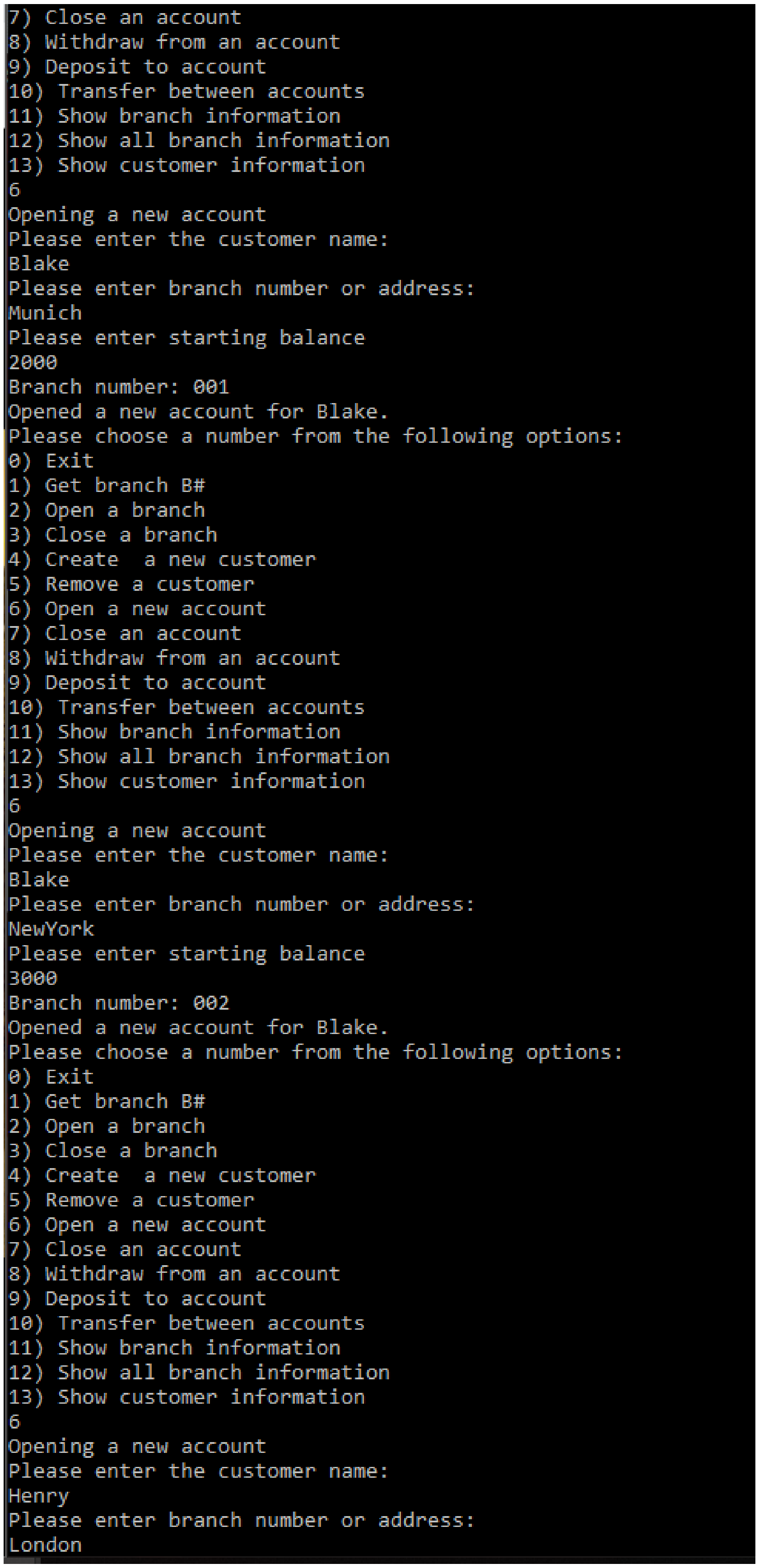


```
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
4
Creating a new customer
Please enter the new customer name:
Henry
Created new customer with C#: 00002 and name: Henry
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
4
Creating a new customer
Please enter the new customer name:
Jones
Created new customer with C#: 00003 and name: Jones
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
4
Creating a new customer
Please enter the new customer name:
Smith
Created new customer with C#: 00004 and name: Smith
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
```

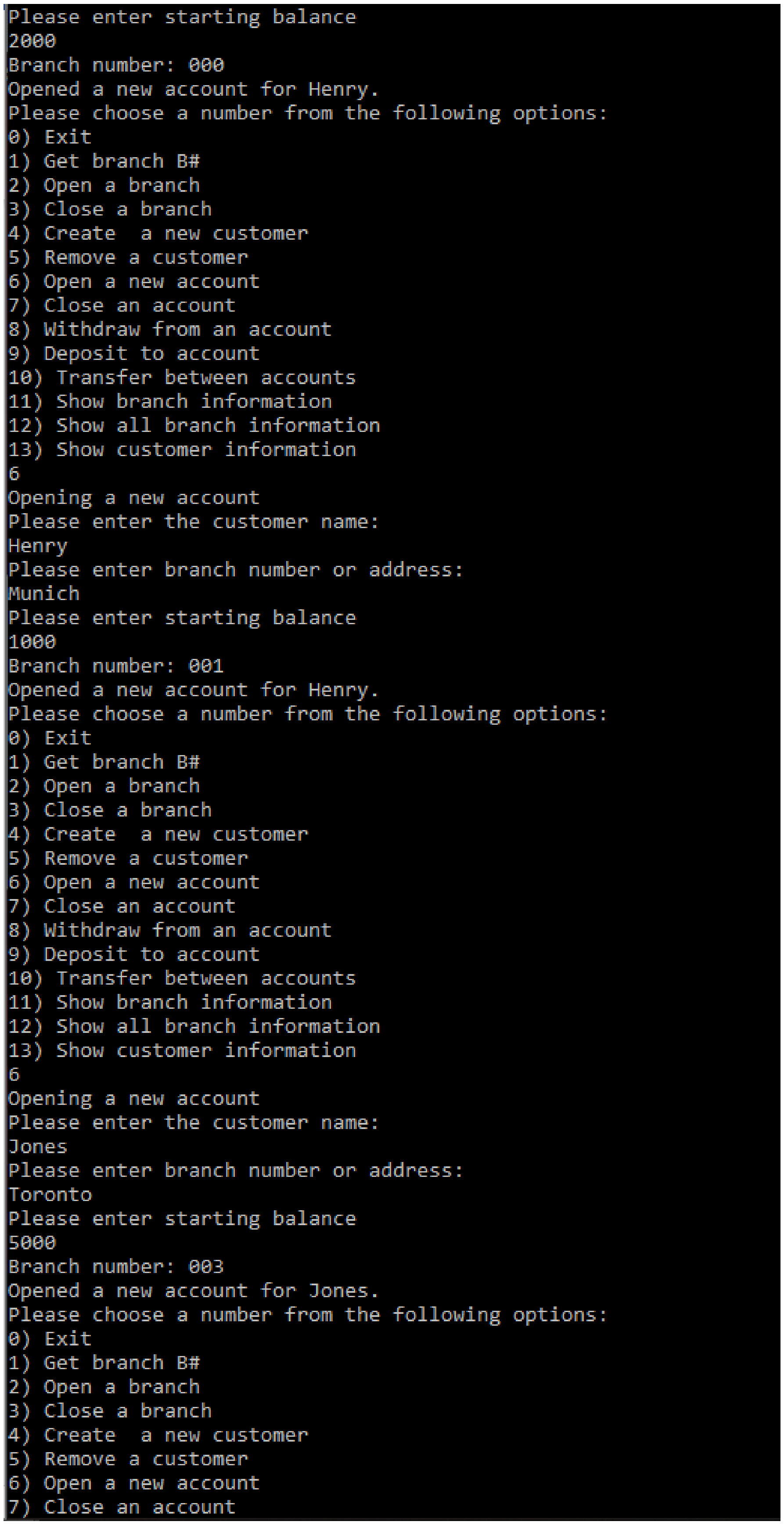


```
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
6
Opening a new account
Please enter the customer name:
Adams
Please enter branch number or address:
London
Please enter starting balance
1000
Branch number: 000
Opened a new account for Adams.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
6
Opening a new account
Please enter the customer name:
Adams
Please enter branch number or address:
Munich
Please enter starting balance
1000
Branch number: 001
Opened a new account for Adams.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
6
Opening a new account
Please enter the customer name:
Adams
Please enter branch number or address:
```

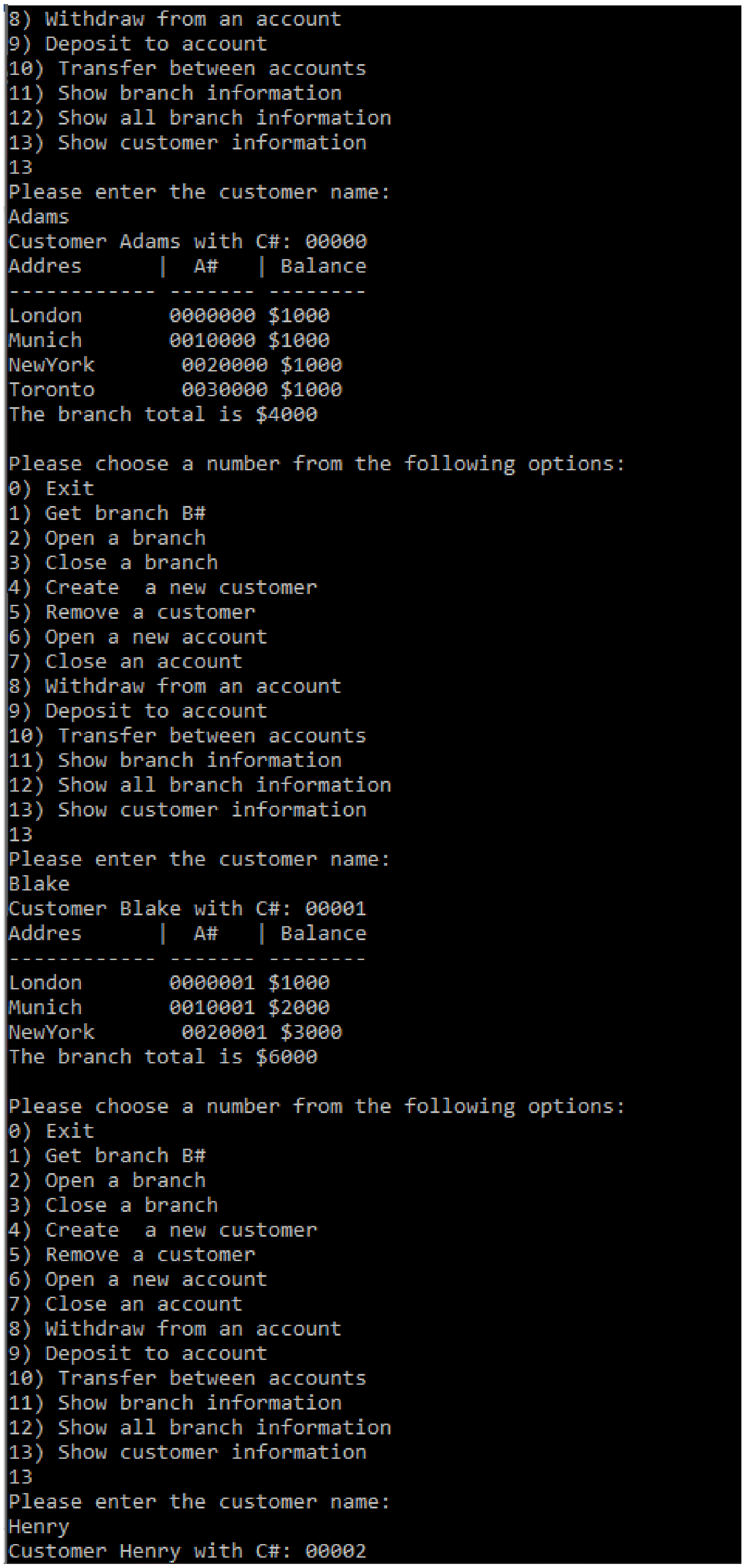

```
NewYork
Please enter starting balance
1000
Branch number: 002
Opened a new account for Adams.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
6
Opening a new account
Please enter the customer name:
Adams
Please enter branch number or address:
Toronto
Please enter starting balance
1000
Branch number: 003
Opened a new account for Adams.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
6
Opening a new account
Please enter the customer name:
Blake
Please enter branch number or address:
London
Please enter starting balance
1000
Branch number: 000
Opened a new account for Blake.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
```



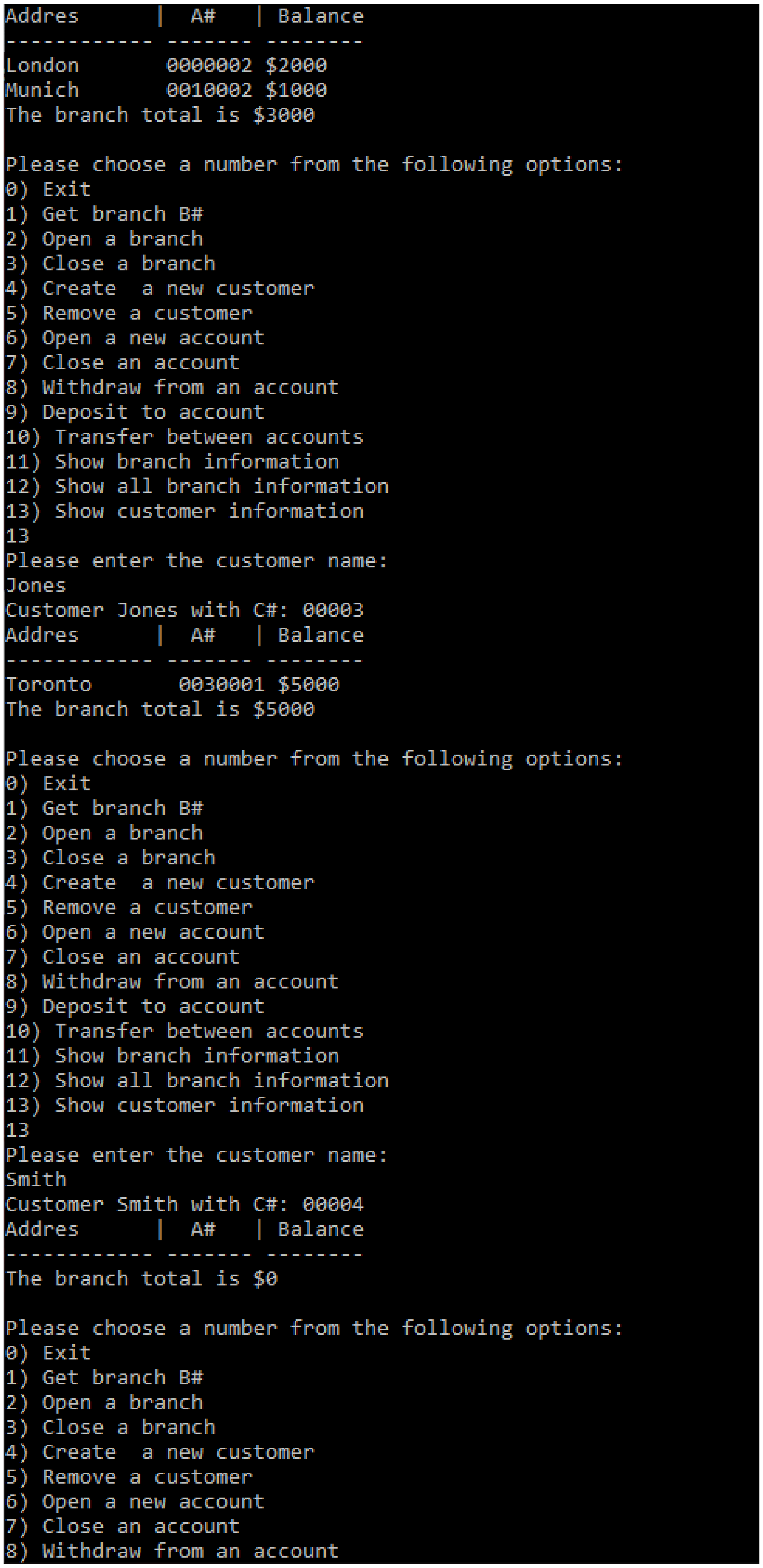
```
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
6
Opening a new account
Please enter the customer name:
Blake
Please enter branch number or address:
Munich
Please enter starting balance
2000
Branch number: 001
Opened a new account for Blake.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
6
Opening a new account
Please enter the customer name:
Blake
Please enter branch number or address:
NewYork
Please enter starting balance
3000
Branch number: 002
Opened a new account for Blake.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
6
Opening a new account
Please enter the customer name:
Henry
Please enter branch number or address:
London
```

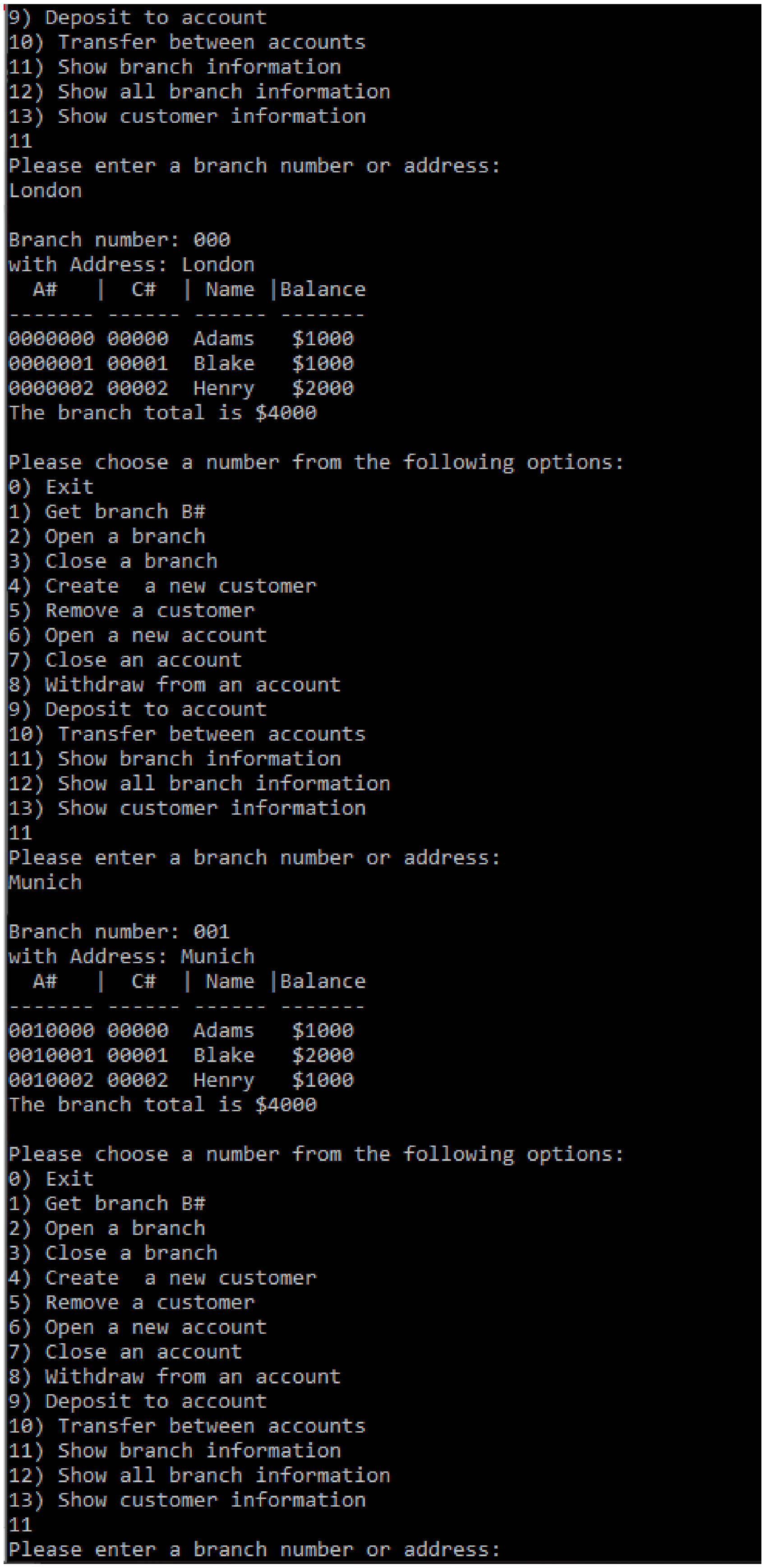


```
Please enter starting balance
2000
Branch number: 000
Opened a new account for Henry.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
6
Opening a new account
Please enter the customer name:
Henry
Please enter branch number or address:
Munich
Please enter starting balance
1000
Branch number: 001
Opened a new account for Henry.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
6
Opening a new account
Please enter the customer name:
Jones
Please enter branch number or address:
Toronto
Please enter starting balance
5000
Branch number: 003
Opened a new account for Jones.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
```

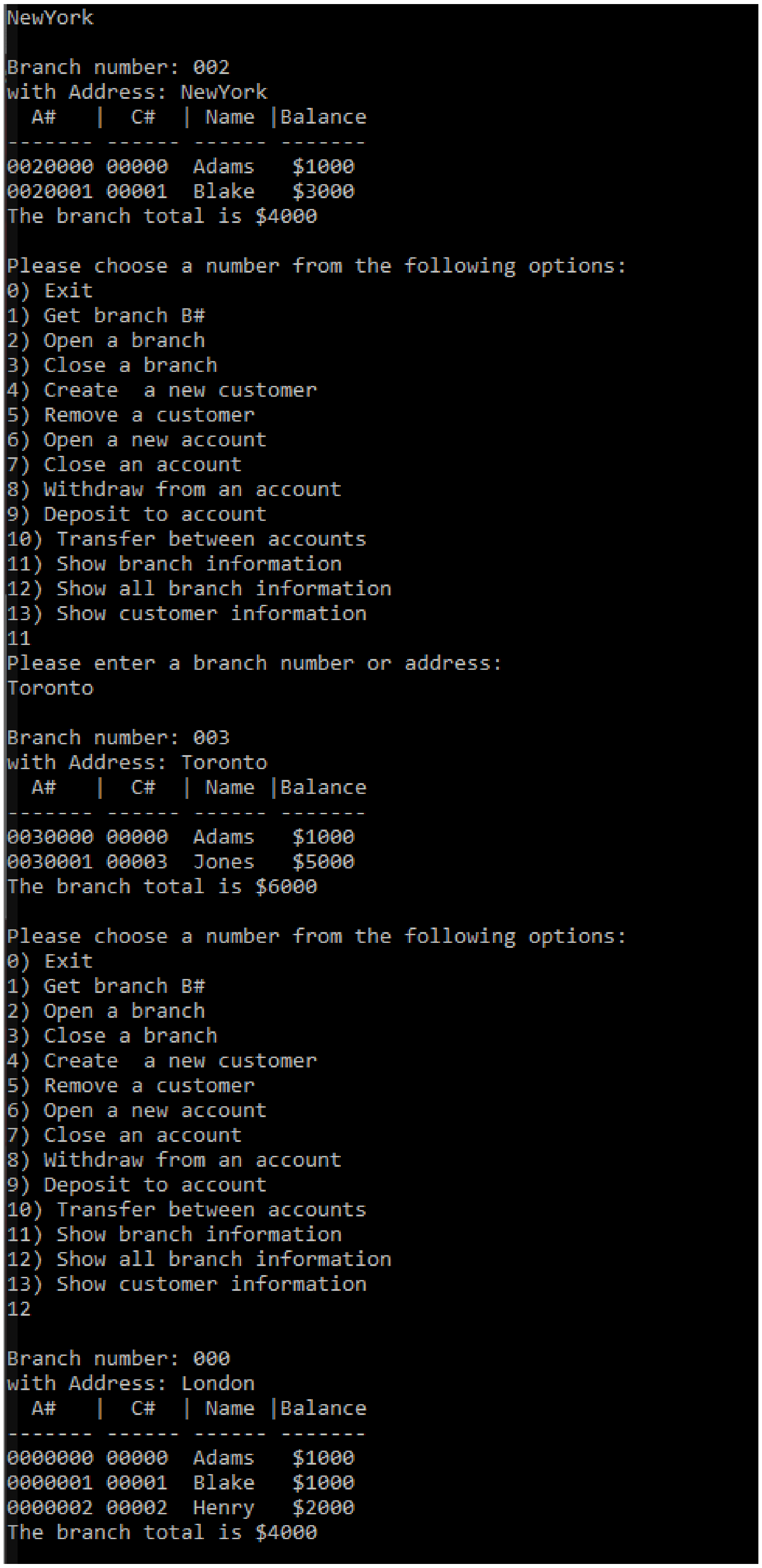


```
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
13
Please enter the customer name:
Adams
Customer Adams with C#: 00000
Addres      |  A#   | Balance
------------ ------- --------
London       0000000 $1000
Munich       0010000 $1000
NewYork       0020000 $1000
Toronto       0030000 $1000
The branch total is $4000

Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
13
Please enter the customer name:
Blake
Customer Blake with C#: 00001
Addres      |  A#   | Balance
------------ ------- --------
London       0000001 $1000
Munich       0010001 $2000
NewYork       0020001 $3000
The branch total is $6000

Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
13
Please enter the customer name:
Henry
Customer Henry with C#: 00002
```

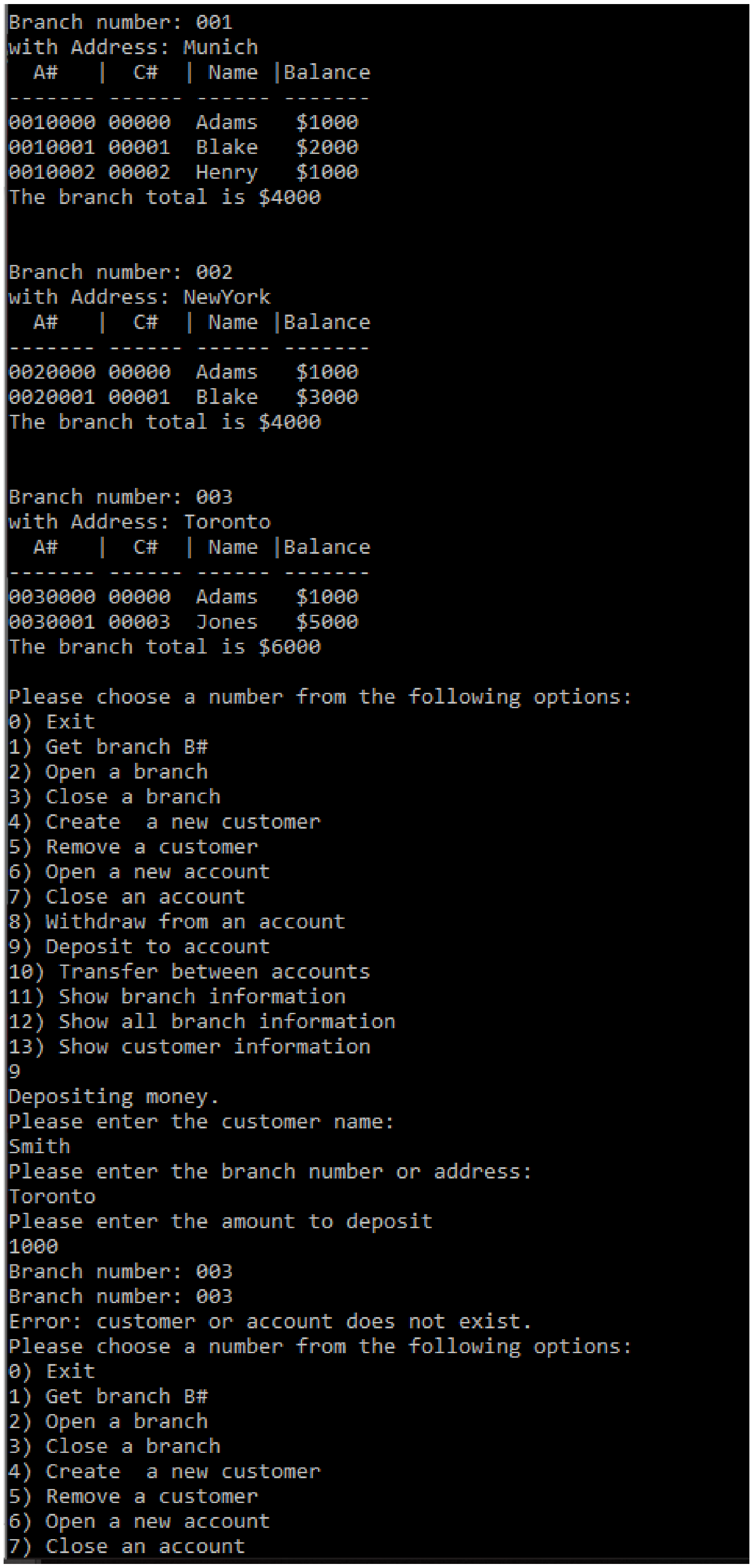

```
Addres       |  A#    | Balance
------------ ------- --------
London       0000002 $2000
Munich       0010002 $1000
The branch total is $3000

Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
13
Please enter the customer name:
Jones
Customer Jones with C#: 00003
Addres       |  A#    | Balance
------------ ------- --------
Toronto       0030001 $5000
The branch total is $5000

Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
13
Please enter the customer name:
Smith
Customer Smith with C#: 00004
Addres       |  A#    | Balance
------------ ------- --------
The branch total is $0

Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
```

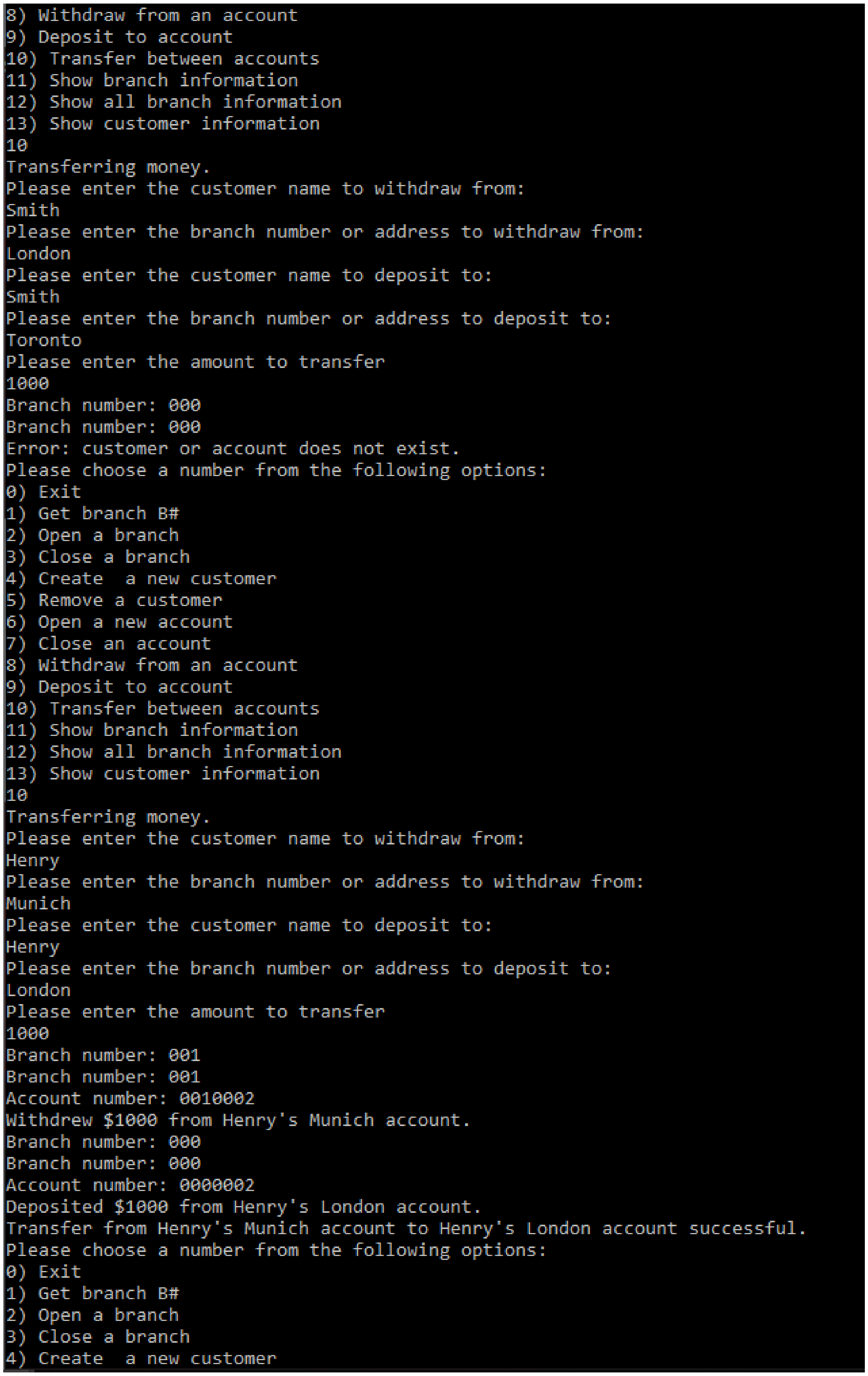


```
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
11
Please enter a branch number or address:
London

Branch number: 000
with Address: London
  A#   |  C#  | Name |Balance
------- ------ ------ -------
0000000 00000  Adams   $1000
0000001 00001  Blake   $1000
0000002 00002  Henry   $2000
The branch total is $4000

Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
11
Please enter a branch number or address:
Munich

Branch number: 001
with Address: Munich
  A#   |  C#  | Name |Balance
------- ------ ------ -------
0010000 00000  Adams   $1000
0010001 00001  Blake   $2000
0010002 00002  Henry   $1000
The branch total is $4000

Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
11
Please enter a branch number or address:
```

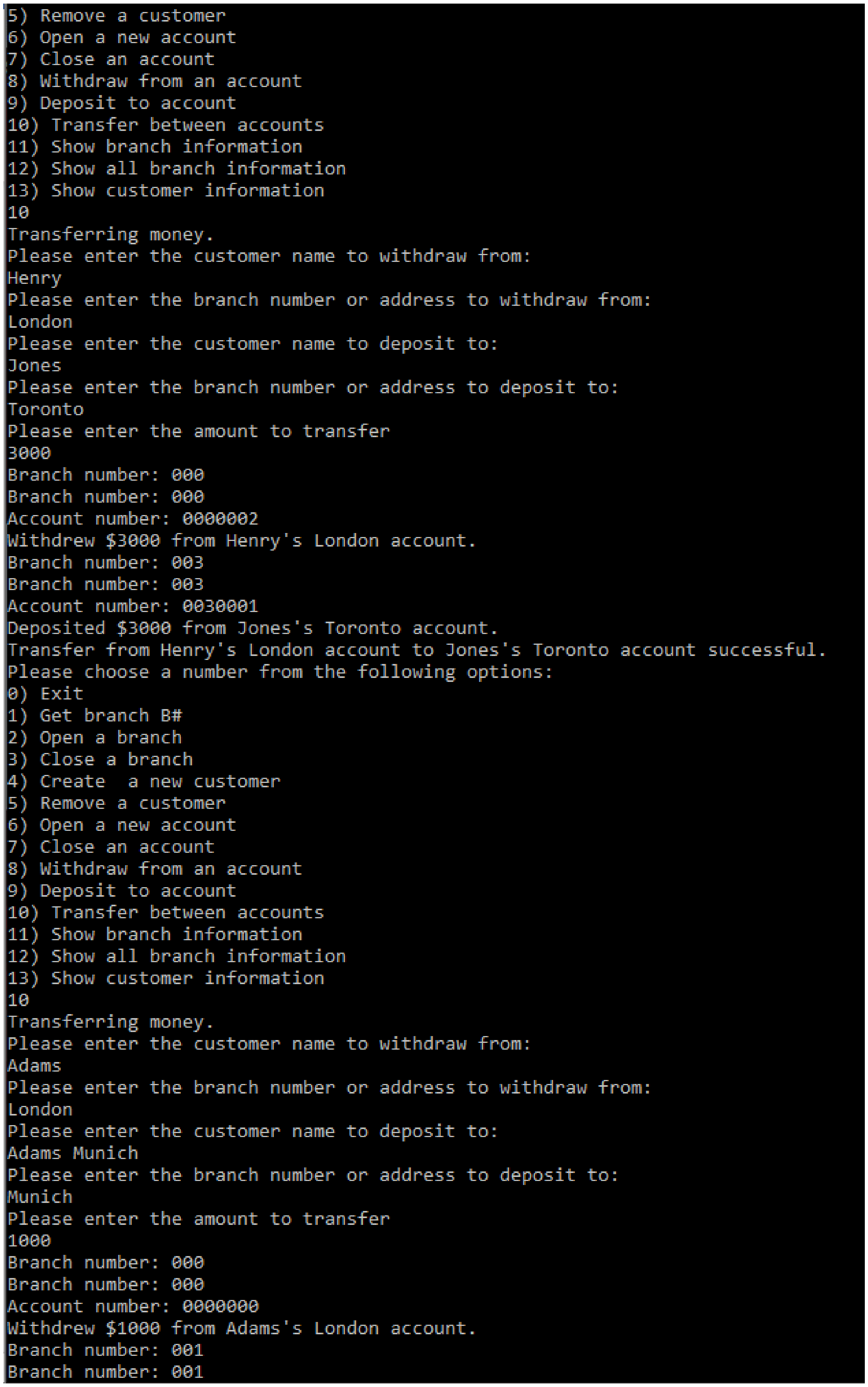


```
NewYork

Branch number: 002
with Address: NewYork
  A#   |  C#  | Name |Balance
------- ------ ------ -------
0020000 00000  Adams   $1000
0020001 00001  Blake   $3000
The branch total is $4000

Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
11
Please enter a branch number or address:
Toronto

Branch number: 003
with Address: Toronto
  A#   |  C#  | Name |Balance
------- ------ ------ -------
0030000 00000  Adams   $1000
0030001 00003  Jones   $5000
The branch total is $6000

Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
12

Branch number: 000
with Address: London
  A#   |  C#  | Name |Balance
------- ------ ------ -------
0000000 00000  Adams   $1000
0000001 00001  Blake   $1000
0000002 00002  Henry   $2000
The branch total is $4000
```

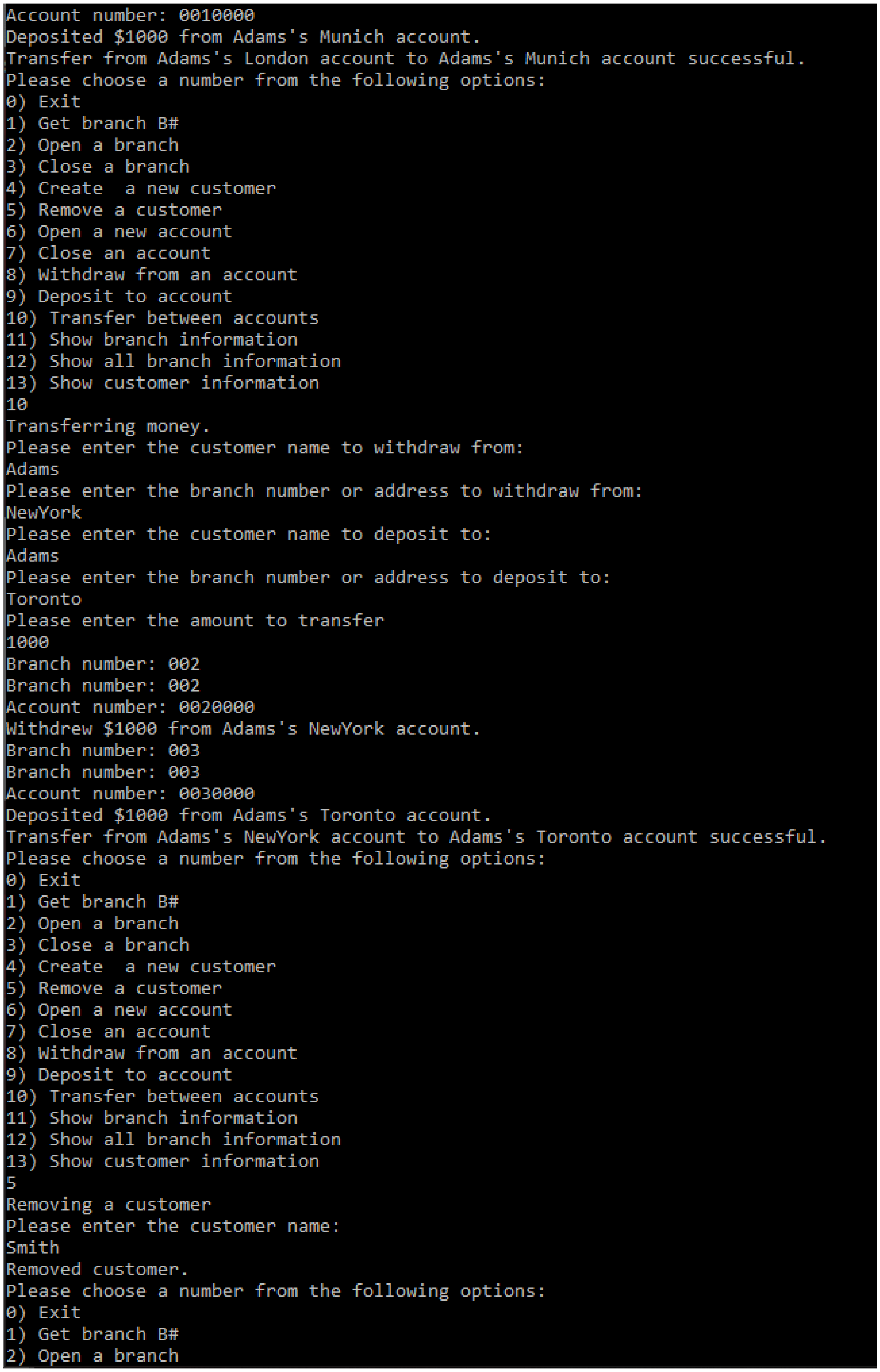

```
Branch number: 001
with Address: Munich
  A#   |  C#   | Name |Balance
------- ------ ------ -------
0010000 00000  Adams   $1000
0010001 00001  Blake   $2000
0010002 00002  Henry   $1000
The branch total is $4000


Branch number: 002
with Address: NewYork
  A#   |  C#   | Name |Balance
------- ------ ------ -------
0020000 00000  Adams   $1000
0020001 00001  Blake   $3000
The branch total is $4000


Branch number: 003
with Address: Toronto
  A#   |  C#   | Name |Balance
------- ------ ------ -------
0030000 00000  Adams   $1000
0030001 00003  Jones   $5000
The branch total is $6000

Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
9
Depositing money.
Please enter the customer name:
Smith
Please enter the branch number or address:
Toronto
Please enter the amount to deposit
1000
Branch number: 003
Branch number: 003
Error: customer or account does not exist.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
```



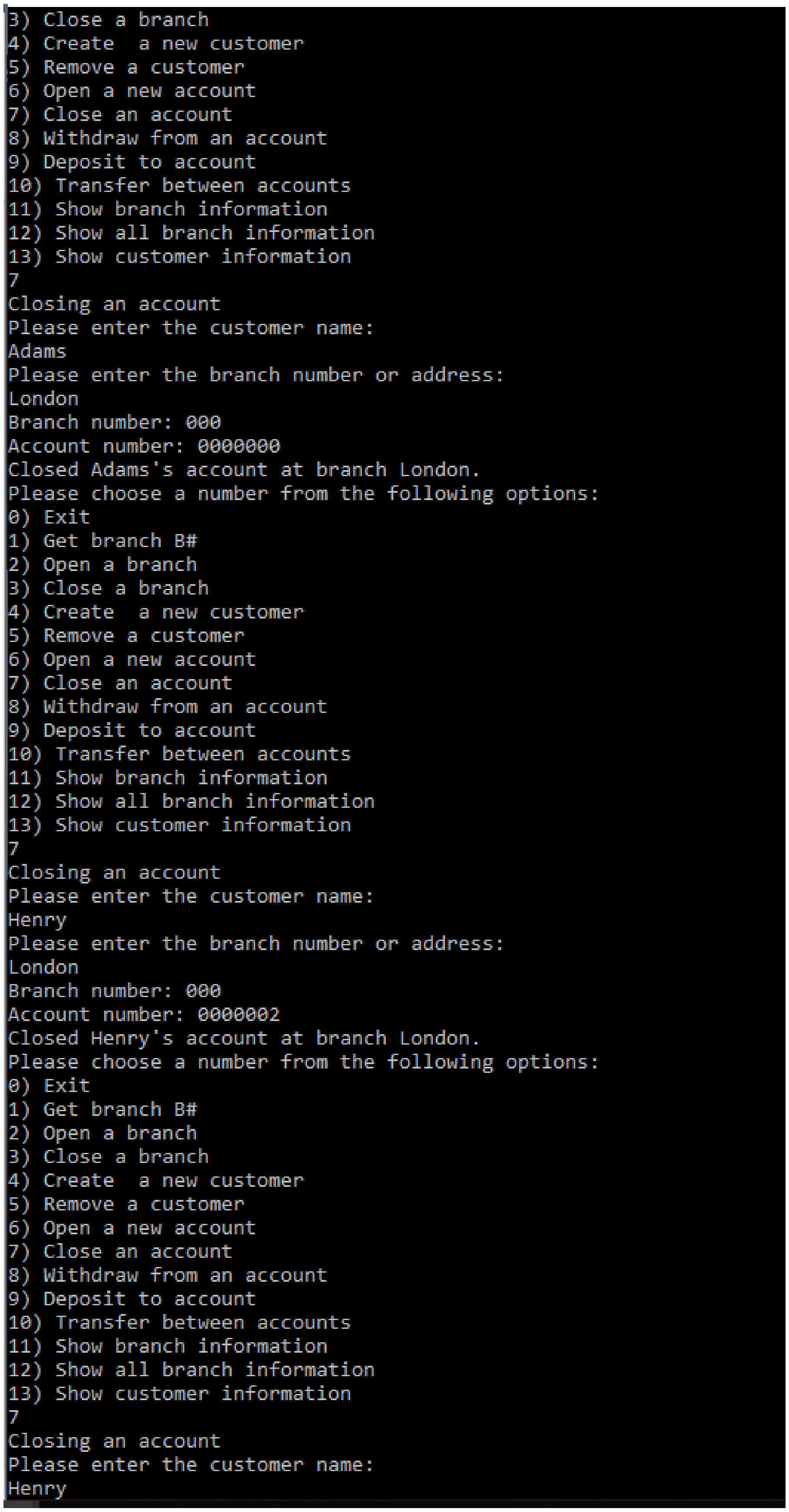
```
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
10
Transferring money.
Please enter the customer name to withdraw from:
Smith
Please enter the branch number or address to withdraw from:
London
Please enter the customer name to deposit to:
Smith
Please enter the branch number or address to deposit to:
Toronto
Please enter the amount to transfer
1000
Branch number: 000
Branch number: 000
Error: customer or account does not exist.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
10
Transferring money.
Please enter the customer name to withdraw from:
Henry
Please enter the branch number or address to withdraw from:
Munich
Please enter the customer name to deposit to:
Henry
Please enter the branch number or address to deposit to:
London
Please enter the amount to transfer
1000
Branch number: 001
Branch number: 001
Account number: 0010002
Withdrew $1000 from Henry's Munich account.
Branch number: 000
Branch number: 000
Account number: 0000002
Deposited $1000 from Henry's London account.
Transfer from Henry's Munich account to Henry's London account successful.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
```



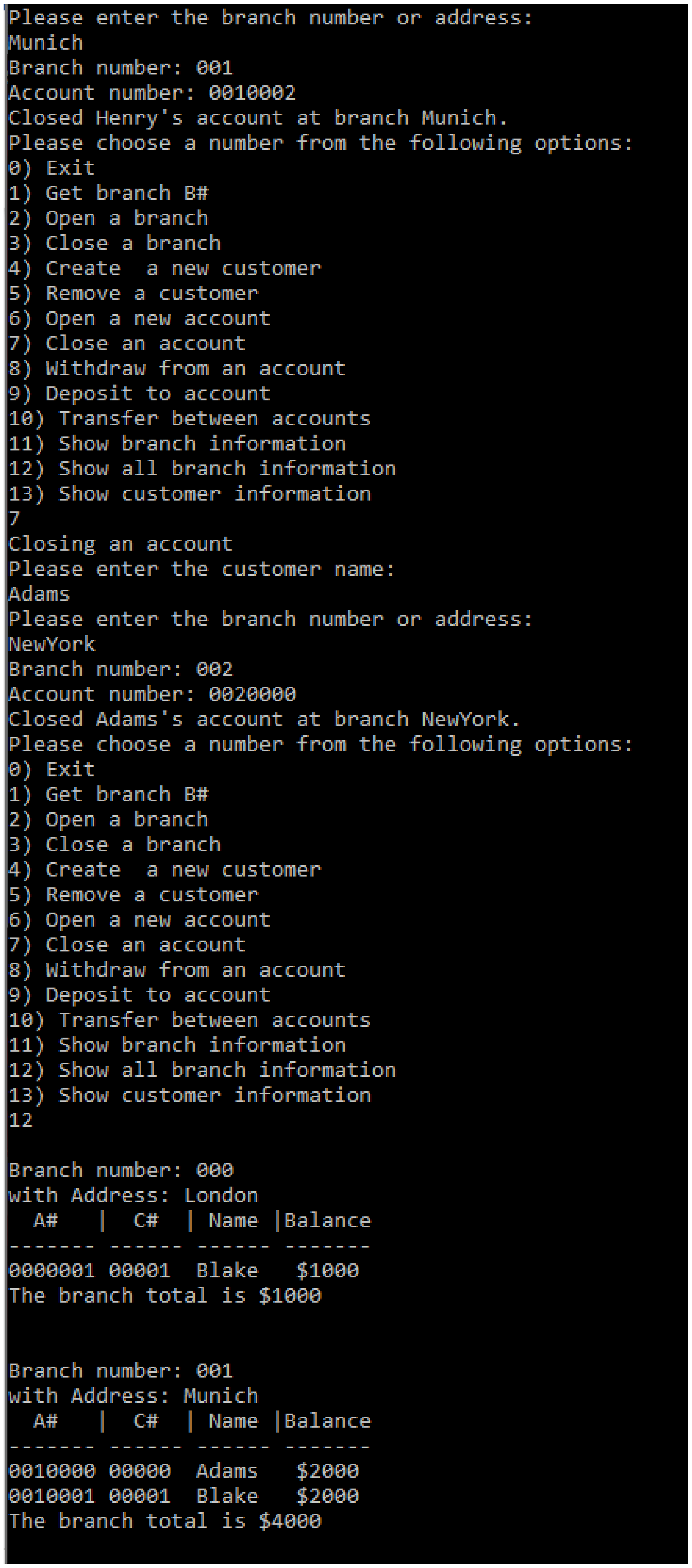
```
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
10
Transferring money.
Please enter the customer name to withdraw from:
Henry
Please enter the branch number or address to withdraw from:
London
Please enter the customer name to deposit to:
Jones
Please enter the branch number or address to deposit to:
Toronto
Please enter the amount to transfer
3000
Branch number: 000
Branch number: 000
Account number: 0000002
Withdrew $3000 from Henry's London account.
Branch number: 003
Branch number: 003
Account number: 0030001
Deposited $3000 from Jones's Toronto account.
Transfer from Henry's London account to Jones's Toronto account successful.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
10
Transferring money.
Please enter the customer name to withdraw from:
Adams
Please enter the branch number or address to withdraw from:
London
Please enter the customer name to deposit to:
Adams Munich
Please enter the branch number or address to deposit to:
Munich
Please enter the amount to transfer
1000
Branch number: 000
Branch number: 000
Account number: 0000000
Withdrew $1000 from Adams's London account.
Branch number: 001
Branch number: 001
```



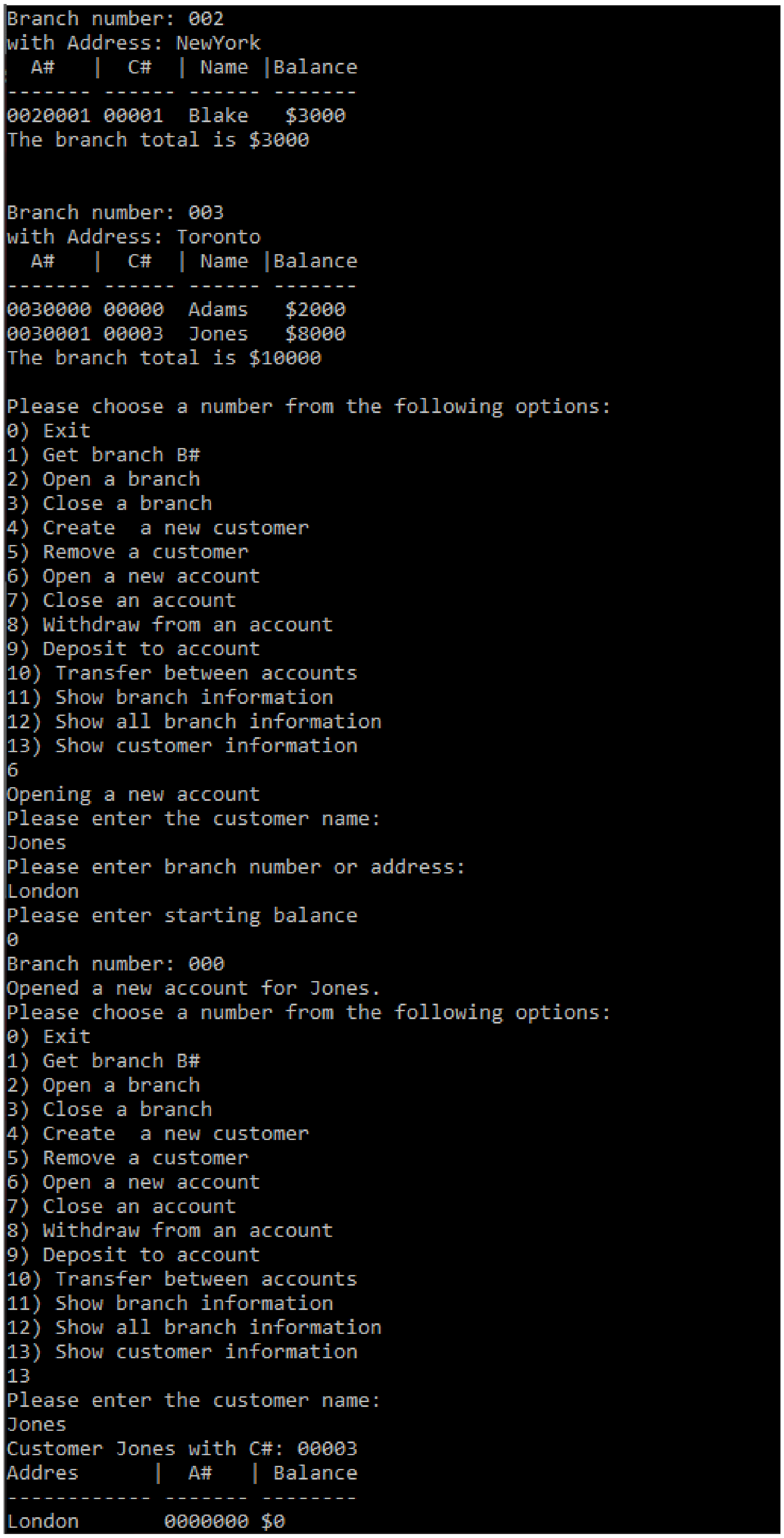
```
Account number: 0010000
Deposited $1000 from Adams's Munich account.
Transfer from Adams's London account to Adams's Munich account successful.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
10
Transferring money.
Please enter the customer name to withdraw from:
Adams
Please enter the branch number or address to withdraw from:
NewYork
Please enter the customer name to deposit to:
Adams
Please enter the branch number or address to deposit to:
Toronto
Please enter the amount to transfer
1000
Branch number: 002
Branch number: 002
Account number: 0020000
Withdrew $1000 from Adams's NewYork account.
Branch number: 003
Branch number: 003
Account number: 0030000
Deposited $1000 from Adams's Toronto account.
Transfer from Adams's NewYork account to Adams's Toronto account successful.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
5
Removing a customer
Please enter the customer name:
Smith
Removed customer.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
```



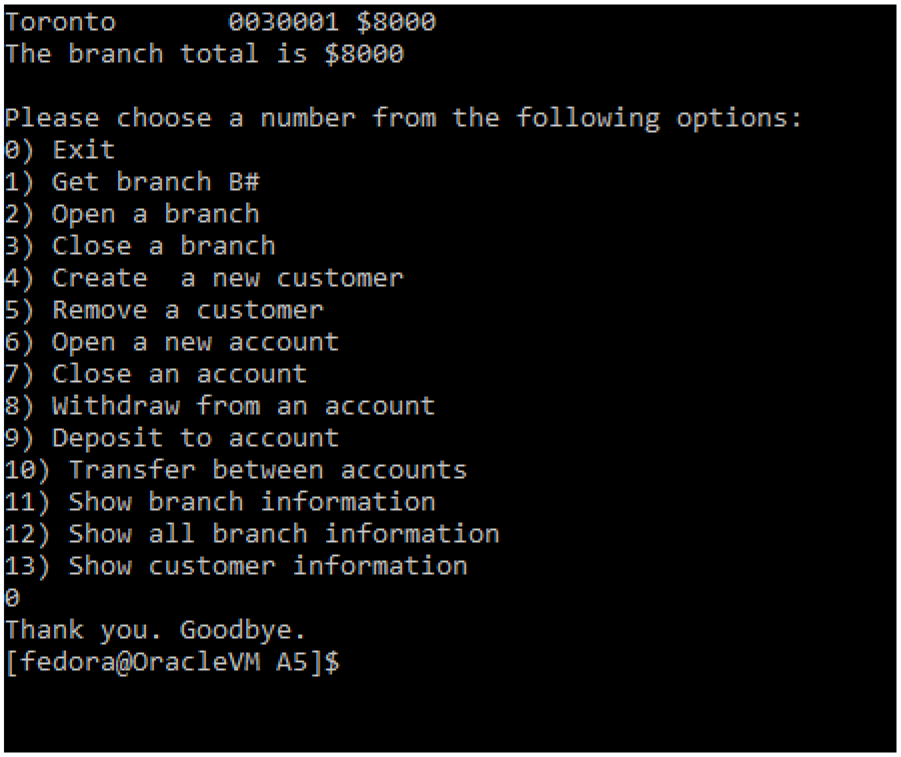
```
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
7
Closing an account
Please enter the customer name:
Adams
Please enter the branch number or address:
London
Branch number: 000
Account number: 0000000
Closed Adams's account at branch London.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
7
Closing an account
Please enter the customer name:
Henry
Please enter the branch number or address:
London
Branch number: 000
Account number: 0000002
Closed Henry's account at branch London.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
7
Closing an account
Please enter the customer name:
Henry
```



```
Please enter the branch number or address:
Munich
Branch number: 001
Account number: 0010002
Closed Henry's account at branch Munich.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
7
Closing an account
Please enter the customer name:
Adams
Please enter the branch number or address:
NewYork
Branch number: 002
Account number: 0020000
Closed Adams's account at branch NewYork.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
12

Branch number: 000
with Address: London
  A#   |  C#  | Name |Balance
------- ------ ------ -------
0000001 00001  Blake   $1000
The branch total is $1000


Branch number: 001
with Address: Munich
  A#   |  C#  | Name |Balance
------- ------ ------ -------
0010000 00000  Adams   $2000
0010001 00001  Blake   $2000
The branch total is $4000
```



```
Branch number: 002
with Address: NewYork
  A#    |  C#   | Name |Balance
------- ------ ------ -------
0020001 00001  Blake   $3000
The branch total is $3000


Branch number: 003
with Address: Toronto
  A#    |  C#   | Name |Balance
------- ------ ------ -------
0030000 00000  Adams   $2000
0030001 00003  Jones   $8000
The branch total is $10000

Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
6
Opening a new account
Please enter the customer name:
Jones
Please enter branch number or address:
London
Please enter starting balance
0
Branch number: 000
Opened a new account for Jones.
Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
13
Please enter the customer name:
Jones
Customer Jones with C#: 00003
Addres      |  A#    | Balance
------------ ------- --------
London       0000000 $0
```



```
Toronto       0030001 $8000
The branch total is $8000

Please choose a number from the following options:
0) Exit
1) Get branch B#
2) Open a branch
3) Close a branch
4) Create  a new customer
5) Remove a customer
6) Open a new account
7) Close an account
8) Withdraw from an account
9) Deposit to account
10) Transfer between accounts
11) Show branch information
12) Show all branch information
13) Show customer information
0
Thank you. Goodbye.
[fedora@OracleVM A5]$
```