HT5

Тесты до исправления:

package root.account;  
  
import org.junit.Test;  
import root.account.Account;  
  
import static org.junit.Assert.\*;  
  
public class AccountTest {  
 @Test  
 public void depositTest() {  
 Account ac = new Account();  
 ac.deposit(500);  
 *assertEquals*(ac.getBalance(), 500);  
 }  
  
 @Test  
 public void withdrawTest() {  
 Account ac = new Account();  
 ac.withdraw(500);  
 *assertEquals*(ac.getBalance(), -500);  
 }  
  
 @Test  
 public void generalTest(){  
 Account ac = new Account();  
 ac.deposit(500);  
 *assertTrue*(ac.withdraw(500));  
 ac.deposit(600);  
 *assertTrue*(ac.withdraw(200));  
 *assertTrue*(ac.withdraw(100));  
 *assertEquals*(ac.getBalance(),300);  
 }  
 @Test  
 public void changeCreditMax(){  
 Account ac = new Account();  
 *assertFalse*(ac.isBlocked());  
 *assertFalse*(ac.setMaxCredit(0));  
 *assertFalse*(ac.setMaxCredit(-1000));  
 *assertFalse*(ac.setMaxCredit(1000));  
 }  
 @Test  
 public void creditMaxBound(){  
 Account ac = new Account();  
 *assertFalse*(ac.setMaxCredit(1000001));  
 *assertFalse*(ac.setMaxCredit(-1000001));  
 *assertEquals*(ac.getMaxCredit(),1000);  
 }  
 @Test  
 public void setCreditMax(){  
 Account ac = new Account();  
 ac.block();  
 *assertTrue*(ac.isBlocked());  
  
 *assertTrue*(ac.setMaxCredit(100));  
 *assertEquals*(ac.getMaxCredit(),100);  
  
 *assertTrue*(ac.setMaxCredit(-100));  
 *assertEquals*(ac.getMaxCredit(),-100);  
  
 }  
 @Test  
 public void testBlock(){  
 Account ac = new Account();  
 ac.deposit(100);  
 ac.withdraw(1500);  
 ac.block();  
 *assertTrue*(ac.isBlocked());  
 *assertTrue*(ac.unblock());  
 *assertFalse*(ac.isBlocked());  
 }  
 @Test  
 public void testBoundTransaction(){  
 Account ac = new Account();  
 *assertFalse*(ac.deposit(-100));  
 *assertFalse*(ac.withdraw(-100));  
 *assertFalse*(ac.deposit(1000001));  
 *assertFalse*(ac.withdraw(1000001));  
 *assertEquals*(ac.getBalance(),0);  
 }  
 @Test  
 public void testBlockedTransaction(){  
 Account ac = new Account();  
 ac.block();  
 *assertFalse*(ac.deposit(100));  
 *assertFalse*(ac.withdraw(100));  
 *assertFalse*(ac.deposit(10000));  
 *assertFalse*(ac.withdraw(500));  
 *assertEquals*(ac.getBalance(),0);  
 }  
 @Test  
 public void breakBound(){  
 Account ac = new Account();  
 ac.deposit(1000001-10);  
 *assertFalse*(ac.deposit(10));  
 *assertEquals*(ac.getBalance(),1000001-10);  
 }  
  
 @Test  
 public void setMaxCreditUnblockedTest() {  
 Account ac = new Account();  
 int tmp = ac.getMaxCredit();  
 *assertFalse*(ac.setMaxCredit(500));  
 *assertEquals*(ac.getMaxCredit(), tmp);  
 ac.withdraw(1000);  
 ac.block();  
 ac.setMaxCredit(100);  
 *assertFalse*(ac.unblock());  
 }  
  
 @Test  
 public void setMaxCreditBoundTest() {  
 Account ac = new Account();  
 ac.block();  
 int tmp = ac.getMaxCredit();  
 *assertFalse*(ac.setMaxCredit(1000000 + 5));  
 *assertEquals*(ac.getMaxCredit(), tmp);  
 }  
  
  
}

Тесты после исправления:

package root.account;  
  
import org.junit.Test;  
import root.account.Account;  
  
import static org.junit.Assert.\*;  
  
public class AccountTestNew {  
 @Test  
 public void depositTest() {  
 Account ac = new Account();  
 ac.deposit(500);  
 *assertEquals*(ac.getBalance(), 500);  
 *assertTrue*(ac.deposit(100));  
 *assertTrue*(ac.deposit(0));  
 *assertTrue*(ac.deposit(999400));  
 }  
  
 @Test  
 public void withdrawTest() {  
 Account ac = new Account();  
 ac.withdraw(500);  
 *assertEquals*(ac.getBalance(), -500);  
 *assertFalse*(ac.withdraw(1000));  
 *assertTrue*(ac.withdraw(0));  
 }  
  
 @Test  
 public void generalTest(){  
 Account ac = new Account();  
 ac.deposit(500);  
 *assertTrue*(ac.withdraw(500));  
 ac.deposit(600);  
 *assertTrue*(ac.withdraw(200));  
 *assertTrue*(ac.withdraw(100));  
 *assertEquals*(ac.getBalance(),300);  
 }  
 @Test  
 public void changeCreditMax(){  
 Account ac = new Account();  
 *assertFalse*(ac.isBlocked());  
 *assertFalse*(ac.setMaxCredit(0));  
 *assertFalse*(ac.setMaxCredit(-1000));  
 *assertFalse*(ac.setMaxCredit(1000));  
 }  
 @Test  
 public void creditMaxBound(){  
 Account ac = new Account();  
 *assertFalse*(ac.setMaxCredit(1000001));  
 *assertFalse*(ac.setMaxCredit(-1000001));  
 *assertEquals*(ac.getMaxCredit(),1000);  
 }  
 @Test  
 public void setCreditMax(){  
 Account ac = new Account();  
 ac.block();  
 *assertTrue*(ac.isBlocked());  
  
 *assertTrue*(ac.setMaxCredit(100));  
 *assertEquals*(ac.getMaxCredit(),100);  
  
 *assertTrue*(ac.setMaxCredit(-100));  
 *assertEquals*(ac.getMaxCredit(),-100);  
  
 }  
 @Test  
 public void testBlock(){  
 Account ac = new Account();  
 ac.deposit(100);  
 ac.withdraw(1500);  
 ac.block();  
 *assertTrue*(ac.isBlocked());  
 *assertTrue*(ac.unblock());  
 *assertFalse*(ac.isBlocked());  
 }  
 @Test  
 public void testBoundTransaction(){  
 Account ac = new Account();  
 *assertFalse*(ac.deposit(-100));  
 *assertFalse*(ac.withdraw(-100));  
 *assertFalse*(ac.deposit(1000001));  
 *assertFalse*(ac.withdraw(1000001));  
 *assertEquals*(ac.getBalance(),0);  
 }  
 @Test  
 public void testBlockedTransaction(){  
 Account ac = new Account();  
 ac.block();  
 *assertFalse*(ac.deposit(100));  
 *assertFalse*(ac.withdraw(100));  
 *assertFalse*(ac.deposit(10000));  
 *assertFalse*(ac.withdraw(500));  
 *assertEquals*(ac.getBalance(),0);  
 }  
 @Test  
 public void breakBound(){  
 Account ac = new Account();  
 ac.deposit(1000001-10);  
 *assertFalse*(ac.deposit(10));  
 *assertEquals*(ac.getBalance(),1000001-10);  
 }  
  
 @Test  
 public void setMaxCreditUnblockedTest() {  
 Account ac = new Account();  
 int tmp = ac.getMaxCredit();  
 *assertFalse*(ac.setMaxCredit(500));  
 *assertEquals*(ac.getMaxCredit(), tmp);  
 ac.withdraw(1000);  
 ac.block();  
 ac.setMaxCredit(100);  
 *assertFalse*(ac.unblock());  
 }  
  
 @Test  
 public void setMaxCreditBoundTest() {  
 Account ac = new Account();  
 ac.block();  
 int tmp = ac.getMaxCredit();  
 *assertFalse*(ac.setMaxCredit(1000000 + 5));  
 *assertEquals*(ac.getMaxCredit(), tmp);  
 *assertTrue*(ac.setMaxCredit(1000000));  
 *assertEquals*(ac.getMaxCredit(),1000000);  
 *assertTrue*(ac.setMaxCredit(-1000000));  
 *assertEquals*(ac.getMaxCredit(),-1000000);  
 }  
 @Test  
 public void boundDepositWithdrawTest() {  
 Account ac = new Account();  
 *assertTrue*(ac.deposit(1000000));  
 *assertTrue*(ac.withdraw(1000000));  
 }  
 @Test  
 public void blockRestrictionTest() {  
 Account ac = new Account();  
 *assertTrue*(ac.withdraw(1000));  
 ac.block();  
 *assertTrue*(ac.isBlocked());  
 *assertTrue*(ac.unblock());  
 *assertFalse*(ac.isBlocked());  
 }  
  
  
}

Мутанты по методам:

Метод deposit:

public boolean deposit(int sum)  
{  
 if(blocked)  
 return false;  
 else if(sum <= 0 || sum +balance> bound)  
 return false;  
 else  
 {  
 balance += sum;  
 return true;  
 }  
}

Мой мутант-эквивалент:

public boolean deposit(int sum)  
{  
 if(blocked)  
 return false;  
 else if(sum < 0 ||sum==0 || sum +balance> bound)  
 return false;  
 else  
 {  
 balance += sum;  
 return true;  
 }  
}

Метод withdraw:

public boolean withdraw(int sum)  
{  
 if(blocked)  
 return false;  
 else if(sum < 0 || sum > bound)  
 return false;  
 else if(balance <= maxCredit + sum)  
 return false;  
 else  
 {  
 balance -= sum;  
 return true;  
 }  
}

Мой мутант-эквивалент:

public boolean withdraw(int sum)  
{  
 if(blocked)  
 return false;  
 else if(sum < 0 || sum > bound)  
 return false;  
 else if(balance <= sum + maxCredit)  
 return false;  
 else  
 {  
 balance -= sum;  
 return true;  
 }  
}

Метод getBalance:

public int getBalance()  
{  
 return 0;  
}

Метод getMaxCredit:

public int getMaxCredit()  
{  
 return maxCredit;  
}

Метод IsBlocked:

public boolean isBlocked()  
{  
 return true;  
}

Метод block:

public void block()  
{  
 blocked = false;  
}

Метод unblock:

public boolean unblock()  
{  
 if(balance <= maxCredit)  
 return false;  
 else  
 blocked = false;  
  
 return true;  
}

Метод setMaxCredit:

public boolean setMaxCredit(int mc)  
{  
 if(mc < bound || mc > bound || !blocked)  
 return false;  
 else  
 maxCredit = -mc;  
  
 return true;  
}

Выжившие мутанты по методам:

1)

public boolean deposit(int sum)  
{  
 if(blocked)  
 return false;  
 else if(sum <= 0 || sum +balance> bound)  
 return false;  
 else  
 {  
 balance += sum;  
 return true;  
 }  
}

2)

public boolean deposit(int sum)  
{  
 if(blocked)  
 return false;  
 else if(sum < 0 || sum +balance>= bound)  
 return false;  
 else  
 {  
 balance += sum;  
 return true;  
 }  
}

3)

public boolean deposit(int sum)  
{  
 if(blocked)  
 return false;  
 else if(sum < 0 || sum +balance> bound)  
 return false;  
 else  
 {  
 balance += sum;  
 return false;  
 }  
}

4)

public boolean withdraw(int sum)  
{  
 if(blocked)  
 return false;  
 else if(sum <= 0 || sum > bound)  
 return false;  
 else if(balance < maxCredit + sum)  
 return false;  
 else  
 {  
 balance -= sum;  
 return true;  
 }  
}

5)

public boolean withdraw(int sum)  
{  
 if(blocked)  
 return false;  
 else if(sum < 0 || sum >= bound)  
 return false;  
 else if(balance < maxCredit + sum)  
 return false;  
 else  
 {  
 balance -= sum;  
 return true;  
 }  
}

6)

public boolean withdraw(int sum)  
{  
 if(blocked)  
 return false;  
 else if(sum < 0 || sum > bound)  
 return false;  
 else if(balance < maxCredit + sum)  
 return true;  
 else  
 {  
 balance -= sum;  
 return true;  
 }  
}

7)

public boolean unblock()  
{  
 if(balance <= maxCredit)  
 return false;  
 else  
 blocked = false;  
  
 return true;  
}

8)

public boolean setMaxCredit(int mc)  
{  
 if(mc <= -bound || mc > bound || !blocked)  
 return false;  
 else  
 maxCredit = -mc;  
  
 return true;  
}

9)

public boolean setMaxCredit(int mc)  
{  
 if(mc < -bound || mc >= bound || !blocked)  
 return false;  
 else  
 maxCredit = -mc;  
  
 return true;  
}

Всего 9 выживших мутантов.

Целый класс эквивалентного мутанта (изменён метод deposit):

*/\*\*  
 \* Copyright (c) 2009 ISP RAS.  
 \* 109004, A. Solzhenitsina, 25, Moscow, Russia.  
 \* All rights reserved.  
 \*  
 \* $Id$  
 \* Created on 08.10.2009  
 \*  
 \*/*package root.account;  
  
*/\*\*  
 \** ***@author*** *Victor Kuliamin  
 \*  
 \*/*public class Account  
{  
 boolean blocked = false;  
 int bound = 1000000;  
 int balance = 0;  
 int maxCredit = -1000;  
  
 public boolean deposit(int sum)  
 {  
 if(blocked)  
 return false;  
 else if(sum < 0 || sum==0 || sum +balance> bound)  
 return false;  
 else  
 {  
 balance += sum;  
 return true;  
 }  
 }  
  
 public boolean withdraw(int sum)  
 {  
 if(blocked)  
 return false;  
 else if(sum < 0 || sum > bound)  
 return false;  
 else if(balance < maxCredit + sum)  
 return false;  
 else  
 {  
 balance -= sum;  
 return true;  
 }  
 }  
  
 public int getBalance()  
 {  
 return balance;  
 }  
  
 public int getMaxCredit()  
 {  
 return -maxCredit;  
 }  
  
 public boolean isBlocked()  
 {  
 return blocked;  
 }  
  
 public void block()  
 {  
 blocked = true;  
 }  
  
 public boolean unblock()  
 {  
 if(balance < maxCredit)  
 return false;  
 else  
 blocked = false;  
  
 return true;  
 }  
  
 public boolean setMaxCredit(int mc)  
 {  
 if(mc < -bound || mc > bound || !blocked)  
 return false;  
 else  
 maxCredit = -mc;  
  
 return true;  
 }  
}

P. S. Автоматические отчёты инструмента тестирования можно найти в папках Before и After.