# Ошибка 1

## Код до исправления

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;  
 }  
}

## Данные, на которых наблюдается некорректное поведение

@Test  
void withdrawMax() {  
 Account account = new Account();  
 *assertTrue*(account.withdraw(account.getMaxCredit()));  
 *assertEquals*(account.maxCredit, account.getBalance());  
}

## Полученное значение, ожидаемое значение

Ожидается, что, в соответствии с требованиями 1g и 2 функция должна возвращать true при значении аргумента равном maxCredit, и false при большем значении. Однако, тест выше не проходит, так как функция возвращает false.

## Код после исправления

Проблему можно решить, если изменить код проверки аргументов с

else if(balance <= maxCredit + sum)

на

else if(balance < maxCredit + sum)

# Ошибка 2

## Код до исправления

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

## Данные, на которых наблюдается некорректное поведение

@Test  
void depositMax() {  
 Account account = new Account();  
 account.deposit(1000);  
 *assertEquals*(1000, account.getBalance());  
  
 *assertFalse*(account.deposit(999001));  
 *assertEquals*(1000, account.getBalance());  
}

## Полученное значение, ожидаемое значение

Ожидается, что, в соответствии с требованием 8, при попытке положить на счет сумму, которая сделает баланс больше, чем BOUND (= 1000000), функция вернет false, и значение баланса не изменится. Однако этого не происходит, функция срабатывает (о чем свидетельствует непрохождение теста выше), баланс меняется и превосходит BOUND.

## Код после исправления

Чтобы исправить ситуацию достаточно лишь добавить проверку на результат баланса, похожую на проверку в функции withdraw, в код функции:

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

else if(balance > BOUND - sum)  
 return false;  
 else  
 {  
 balance += sum;  
 return true;  
 }  
}

# Ошибка 3

## Код до исправления

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

## Данные, на которых наблюдается некорректное поведение

@Test  
void setMaxCreditUnblocked() {  
 Account account = new Account();  
 *assertFalse*(account.setMaxCredit(100));  
 *assertEquals*(1000, account.getMaxCredit());  
}  
  
@Test  
void setMaxCreditUnblockedValueChanged() {  
 Account account = new Account();  
 account.setMaxCredit(100);  
 *assertEquals*(1000, account.getMaxCredit());  
}

## Полученное значение, ожидаемое значение

Ожидается, что, в соответствии с требованием 3 функция должна возвращать false при попытке изменить максимальный кредит, если аккаунт не заблокирован. Также значение максимального кредита не должно меняться, однако, как можно видеть по тестам выше, это не происходит. Даже в разблокированном аккаунте функция меняет максимальный кредит и возвращает true.

## Код после исправления

Для обработки этой ситуации стоит добавить в код функции проверку на блокировку аккаунта (как, например, в функции withdraw, только наоборот):

public boolean setMaxCredit(int mc)  
{

if(!blocked)  
 return false;  
  
 if(mc < -bound || mc > bound)  
 return false;  
 else  
 maxCredit = -mc;  
  
 return true;  
}

добавление по поводу setMaxCredit и требований: вероятно нам хотелось бы, чтобы невозможно было задать отрицательные значения максимального кредита, но в требованиях нет никаких ограничений на положительность параметров, в реализации функции тоже, поэтому нарушения нет, но по логике работы приложения это наверное не то, как оно должно быть.