**Список обнаруженных ошибок и исправлений**

**Выполнил Игумнов Никита БПИ191**

**1.**

1.

public int getCoins2() {  
 if (mode == Mode.*OPERATION*)  
 return coins1;  
 else  
 return coins2;  
}

2. Нарушение требования “Возвращает количество внесенных монет 2 вида. Вне режима отладки возвращает 0.”

3. При получении количества монет из режима OPERATION возвращается ненулевое значение.

4.

public int getCoins2() {  
 if (mode == Mode.*OPERATION*)  
 return 0;  
 else  
 return coins2;  
}

**2.**

1.

public Response fillProduct1() {  
 num1 = max2;  
 return Response.*OK*;  
}

2. Нарушение требования “Заполняет отделение с 1 продуктом до максимального количества 1 продукта. Функционирует только в режиме отладки. При запуске в некорректном режиме возвращает ILLEGAL\_OPERATION”,

3. При заполнении аппарата товарами в режиме OPERATION никаких ошибок не возникает и аппарат заполняется товарами, при заполнении автомата товарами типа 1, количество товаров автомата становится равно максимальному количеству товаров типа 2

4.

public Response fillProduct1() {  
 if (mode == Mode.*OPERATION*) return Response.*ILLEGAL\_OPERATION*;  
  
 num1 = max1;  
 return Response.*OK*;  
}

**3.**

1.

public Response fillProduct2() {  
 num2 = max2;  
 return Response.*OK*;  
}

2. Нарушение требования “Заполняет отделение с 2 продуктом до максимального количества 2 продукта. Функционирует только в режиме отладки. При запуске в некорректном режиме возвращает ILLEGAL\_OPERATION”,

3. При заполнении аппарата товарами в режиме OPERATION никаких ошибок не возникает и аппарат заполняется товарами, при заполнении автомата товарами типа 2

4.

public Response fillProduct2() {  
 if (mode == Mode.*OPERATION*) return Response.*ILLEGAL\_OPERATION*;  
  
 num2 = max2;  
 return Response.*OK*;  
}

**4.**

1.

public Response enterAdminMode(long code) {  
 if (balance != 0) return Response.*UNSUITABLE\_CHANGE*;  
 if (code != id) return Response.*INVALID\_PARAM*;   
 mode = Mode.*ADMINISTERING*;  
 return Response.*OK*;  
}

2. Нарушение требования “Переводит автомат в режим администрирования. В качестве параметра принимает секретный код. При несовпадении кода с эталоном возвращает INVALID\_PARAM. При наличии внесенных покупателем средств перехода в режим отладки не происходит и возвращается CANNOT\_PERFORM”,

3. При проверке – вначале проверяется balance, а только после - code, при входе в режим отладки с ненулевым балансом возвращается ошибка UNSUITABLE\_CHANGE.

4.

public Response enterAdminMode(long code) {  
 if (code != id) return Response.*INVALID\_PARAM*;  
 if (balance != 0) return Response.*CANNOT\_PERFORM*;  
 mode = Mode.*ADMINISTERING*;  
 return Response.*OK*;  
}

**5.**

1.

public Response setPrice1(int p) {  
 if (mode == Mode.*OPERATION*) return Response.*ILLEGAL\_OPERATION*;   
  
 price1 = p;  
 return Response.*OK*;  
}

2. Нарушение требования “ При попытке установки значения цены p меньше или равной 0 возвращает INVALID\_PARAM”,

3. Возможна установка значения равного или меньшего нулю.

4.

public Response setPrice1(int p) {  
 if (mode == Mode.*OPERATION*) return Response.*ILLEGAL\_OPERATION*;  
 if (p <= 0) return Response.*INVALID\_PARAM*;  
  
 price1 = p;  
 return Response.*OK*;  
}

**6.**

1.

public Response putCoin1() {  
 if (mode == Mode.*ADMINISTERING*) return Response.*ILLEGAL\_OPERATION*;  
 if (coins2 == maxc2) return Response.*CANNOT\_PERFORM*;  
  
 balance += *coinval2*;  
 coins2++;  
  
 return Response.*OK*;   
}  
  
public Response putCoin2() {  
 if (mode == Mode.*ADMINISTERING*) return Response.*ILLEGAL\_OPERATION*;  
 if (coins1 == maxc1) return Response.*CANNOT\_PERFORM*;  
  
 balance += *coinval1*;  
 coins1++;  
  
 return Response.*OK*;  
}

2. Нарушение требования “Добавляет монету 1 вида на счет пользователя… Добавляет монету 2 вида на счет пользователя…”,

3. При выполнении putCoin1 все операции происходят с coins2, coinval2, maxc2, для putCoins – coins1, coinval1, maxc1.

4.

public Response putCoin1() {  
 if (mode == Mode.*ADMINISTERING*) return Response.*ILLEGAL\_OPERATION*;  
 if (coins1 == maxc1) return Response.*CANNOT\_PERFORM*;  
  
 balance += *coinval1*;  
 coins1++;  
  
 return Response.*OK*;  
}  
  
public Response putCoin2() {  
 if (mode == Mode.*ADMINISTERING*) return Response.*ILLEGAL\_OPERATION*;  
 if (coins2 == maxc2) return Response.*CANNOT\_PERFORM*;  
  
 balance += *coinval2*;  
 coins2++;  
  
 return Response.*OK*;  
}

**7.**

1. (returnMoney())

else {  
 // using coinval1 == 1  
 coins1 -= (res / *coinval2*);  
 coins2--;  
 balance = 0;   
  
 return Response.*OK*;  
}

2. Нарушение требования “Во всех иных случаях выдается баланс/2 монет 2 вида и 1 монета 1 вида”,

3. При возвращении денег торговый автомат дает монеты первого типа вместо второго и второго типа вместо первого.

4.

else {  
 // using coinval1 == 1  
 coins2 -= (res / *coinval2*);  
 coins1--;  
 balance = 0;   
  
 return Response.*OK*;  
}

**8.**

1. (giveProduct2(int number))

else if (res > coins1 \* *coinval1* + coins2 \* *coinval2*) {  
 return Response.*INSUFFICIENT\_MONEY*;  
}

2. Нарушение требования “Если после выполнения операции в автомате недостаточно сдачи, то возвращается TOO\_BIG\_CHANGE”,

3. При выдаче второго продукта, если после выполнения операции недостаточно сдачи, то возвращается INSUFFICIENT\_MONEY.

4.

else if (res > coins1 \* *coinval1* + coins2 \* *coinval2*) {  
 return Response.*TOO\_BIG\_CHANGE*;  
}

**9.**

1. (giveProduct2(int number))

else {  
 // using coinval1 == 1  
 coins1 -= (res / *coinval2*);  
 coins2--;  
 balance = 0;  
 num2 -= number;  
  
 return Response.*OK*;  
}

2. Нарушение требования “В остальных случаях сдача выдается монетами 2 вида, когда это возможно, затем – монетами 1 вида”,

3. При выдаче второго продукта торговый автомат дает монеты первого типа вместо второго и второго типа вместо первого.

4.

else {  
 // using coinval1 == 1  
 coins2 -= (res / *coinval2*);  
 coins1--;  
 balance = 0;  
 num2 -= number;  
  
 return Response.*OK*;  
}