Сообщение об ошибках

I. Дублирование кода в методе *Response giveProduct(int number)*

1. Код до исправления:

```
public Response giveProduct(int number) {
    if (mode == Mode.ADMINISTERING) return Response.ILLEGAL_OPERATION;

    if (number <= 0 || number > max) return Response.INVALID_PARAM;
    if (number > num) return Response.INSUFFICIENT_PRODUCT;

    int res = balance - number * price;

    if (res < 0) return Response.INSUFFICIENT_MONEY;
    else if (res > coins1 * coinval1 + coins2 * coinval2) {
        return Response.INSUFFICIENT_MONEY;
    } else if (res > coins2 * coinval2) {
        // using coinval1 == 1
        coins1 -= (res - coins2 * coinval2);
        coins2 = 0;
        balance = 0;
        num -= number;


        return Response.OK;
    } else if (res % coinval2 == 0) {
        coins2 -= (res / coinval2);
        balance = 0;
        num -= number;


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


        return Response.OK;
    }

}
```

2. Данные, на которых наблюдается некорректное поведение: —
3. Полученное и ожидаемое значения: —
4. Код после исправления:

```
public Response giveProduct(int number) {
    if (mode == Mode.ADMINISTERING) return Response.ILLEGAL_OPERATION;

    if (number <= 0 || number > max) return Response.INVALID_PARAM;
    if (number > num) return Response.INSUFFICIENT_PRODUCT;
```

```
    if (balance < number * price) return Response.INSUFFICIENT_MONEY;
    else {
        balance -= number * price;
        Response response = returnMoney();
        if (response != Response.OK) balance += number * price;
        return response;
    }
}
```

II. Перепутаны названия методов *Response putCoin1()* и *Response putCoin2()*

1. Код до исправления:

```
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. Данные, на которых наблюдается некорректное поведение:

   При вызове метода *Response putCoin1()* увеличивается количество монет второго типа, аналогично при вызове метода *Response putCoin2()* увеличивается количество монет первого типа.

3. Полученное и ожидаемое значения:

   При вызове *Response putCoin1()* ожидается c1 = 1 и c2 = 0, получаем c1 = 0 и c2 = 1.

При вызове *Response putCoin2()* ожидается c1 = 0 и c2 = 1, получаем c1 = 1 и c2 = 0.

4. Код после исправления:

```
public Response putCoin2() {
    if (mode == Mode.ADMINISTERING) return Response.ILLEGAL_OPERATION;
    if (coins2 == maxc2) return Response.CANNOT_PERFORM;

    balance += coinval2;
    coins2++;

    return Response.OK;
}

public Response putCoin1() {
    if (mode == Mode.ADMINISTERING) return Response.ILLEGAL_OPERATION;
    if (coins1 == maxc1) return Response.CANNOT_PERFORM;

    balance += coinval1;
    coins1++;

    return Response.OK;
}
```

III. Несоответствие метода *int getCoins2()* документации (неверная работа в режиме операций)

1. Код до исправления:

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

2. Данные, на которых наблюдается некорректное поведение: c1 = 50 и c2 = 1
3. Полученное и ожидаемое значения: получено 50, ожидаемое 0
4. Код после исправления:

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

IV. Несоответствие метода *Response fillProducts()* документации (неверная обработка в режиме операций)

1. Код до исправления:

```
public Response fillProducts() {
    num = max;
    return Response.OK;
}
```

2. Данные, на которых наблюдается некорректное поведение: режим операции
3. Полученное и ожидаемое значения: получено *OK*, ожидаемое *ILLEGAL_OPERATION*
4. Код после исправления:

```
public Response fillProducts() {
    if (mode == Mode.OPERATION) return Response.ILLEGAL_OPERATION;
    num = max;
    return Response.OK;
}
```

V. Несоответствие метода *Response fillCoins(int c1, int c2)* документации (не обрабатывается случай, где монет первого типа больше допустимого значения)
   1. Код до исправления:

```
public Response fillCoins(int c1, int c2) {
    if (mode == Mode.OPERATION) return Response.ILLEGAL_OPERATION;
    if (c1 <= 0 || c2 > maxc1) return Response.INVALID_PARAM;
    if (c2 <= 0 || c2 > maxc2) return Response.INVALID_PARAM;
    coins1 = c1;
    coins2 = c2;
    return Response.OK;
}
```

   2. Данные, на которых наблюдается некорректное поведение: c1 = 51 и c2 = 5
   3. Полученное и ожидаемое значения: получено *OK*, ожидаемое *INVALID_PARAM*
   4. Код после исправления:

```
public Response fillCoins(int c1, int c2) {
    if (mode == Mode.OPERATION) return Response.ILLEGAL_OPERATION;
    if (c1 <= 0 || c1 > maxc1) return Response.INVALID_PARAM;
    if (c2 <= 0 || c2 > maxc2) return Response.INVALID_PARAM;
    coins1 = c1;
    coins2 = c2;
    return Response.OK;
}
```

VI. Несоответствие метода *Response enterAdminMode(long code)* документации (неверно обрабатывается случай с ненулевым балансом)
   1. Код до исправления:

```
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. Данные, на которых наблюдается некорректное поведение: code = *117345294655382* и balance = 1
3. Полученное и ожидаемое значения: получено *UNSUITABLE_CHANGE*, ожидаемое *CANNOT_PERFORM*
4. Код после исправления:

```
public Response enterAdminMode(long code) {
    if (balance != 0) return Response.CANNOT_PERFORM;
    if (code != id) return Response.INVALID_PARAM;
    mode = Mode.ADMINISTERING;
    return Response.OK;
}
```

VII. Несоответствие метода *Response setPrices(int p)* документации (не обрабатывается случай с отрицательной ценой)

1. Код до исправления:

```
public Response setPrices(int p) {
    if (mode == Mode.OPERATION) return Response.ILLEGAL_OPERATION;
    if (price <= 0) return Response.INVALID_PARAM;
    price = p;
    return Response.OK;
}
```

2. Данные, на которых наблюдается некорректное поведение: p = -3
3. Полученное и ожидаемое значения: получено *OK*, ожидаемое *INVALID_PARAM*
4. Код после исправления:

```
public Response setPrices(int p) {
    if (mode == Mode.OPERATION) return Response.ILLEGAL_OPERATION;
    if (p <= 0) return Response.INVALID_PARAM;
    price = p;
    return Response.OK;
}
```

VIII. Несоответствие метода *Response returnMoney()* документации (неверно обрабатывается случай, когда выдается нечетная сдача)

1. Код до исправления:

```
else {
   // using coinval1 == 1
   coins1 -= (balance / coinval2);
   coins2--;
   balance = 0;

   return Response.OK;
}
```

2. Данные, на которых наблюдается некорректное поведение: c1 = 4, c2 = 4 и balance = 7
3. Полученное и ожидаемое значения: получено c1 = 1 и c2 = 3, ожидаемое *c1 = 3 и c2 = 0*
4. Код после исправления:

```
else {
   // using coinval1 == 1
   coins2 -= (balance / coinval2);
   coins1--;
   balance = 0;

   return Response.OK;
}
```