VALIDATION REPORT

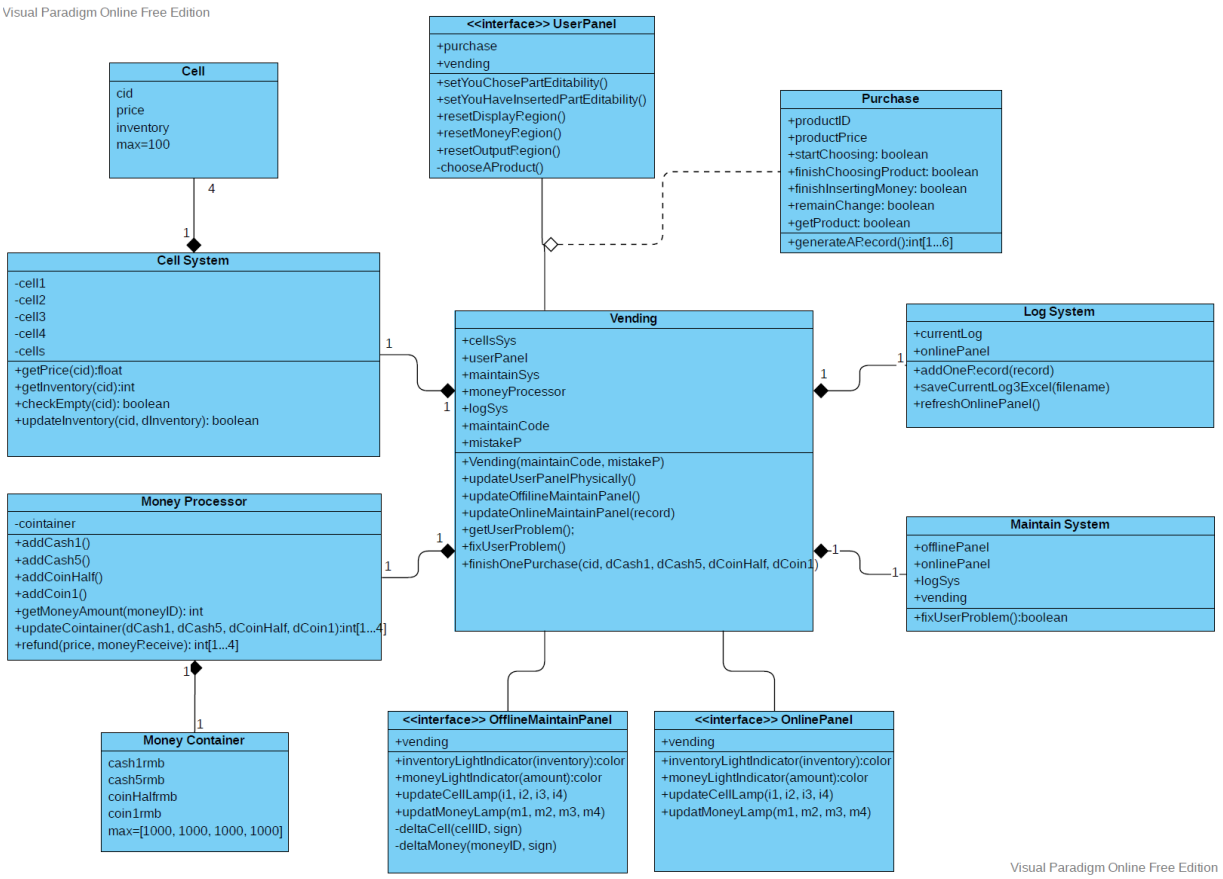
Vending

Group 1
Author: Tiansu Chen

Table of Contents

System Architecture

The system architecture is shown below:



ALL private functions are changed to public functions when testing.

T1: Unit Test

Declaration: since the physical containers(money containers and cells act as a data structures that do not have any function besides the constructor in the code, they are not tested separately without the corresponding system.

T1.1 Money Processor + Money Container Unit Test

T1.1.1 Test refund(obj, price, moneyReceive)

```
% the program for deciding how to refund
function v = refund(obj, price, moneyReceive)
    change = moneyReceive - price;
    mistake = 0;
    remainChange = 0;
    if obj.container.coin1rmb >= floor(change) && obj.container.coinHalfrmb >= (change - floor(change))/0.5
        % Brance T Cover 1.1.1.1
        % cool in both cases
        coin1 = floor(change);
        coinHalf = (change - floor(change))/0.5;
    elseif obj.container.coin1rmb < floor(change) && obj.container.coinHalfrmb >= (change- obj.container.coin1rmb)/0.5
        % Brance T Cover 1.1.1.2
        % coin1 is not enough but coin 0.5 is enough to cover
        coin1 = obj.container.coin1rmb;
        coinHalf = (change-coin1)/0.5;
    else
        % Brance T Cover 1.1.1.3
        % coin 1 and coin 0.5 in total is not enough to cover
        coin1 =  obj.container.coin1rmb;
        coinHalf = obj.container.coinHalfrmb;
        remainChange = change - (coin1 + coinHalf*0.5);
        % this is for the failcase dealing:
        % return [-1, remainChange, coinHalf, coin1].
        mistake = 1;
        % call maintainer and display sorry&how2callmaintainer info
    end
    if mistake ~= 1
        % Brance T Cover 1.1.1.4
        v = [coinHalf, coin1];
    else
        % Brance T Cover 1.1.1.5
        v = [-1, remainChange, coinHalf, coin1];
    end
    obj.container.coin1rmb = obj.container.coin1rmb - coin1;
    obj.container.coinHalfrmb = obj.container.coinHalfrmb - coinHalf;
end
```

- Coverage Criteria: Branch coverage
- Test Case

	Test Case T1.1.1.1	Test Case T1.1.1.2	Test Case T1.1.1.3
Coverage Item	T cover 1.1.1.1, T cover 1.1.1.4	T cover 1.1.1.2, T cover 1.1.1.4	T cover 1.1.1.3, T cover 1.1.1.5
Input	Price=2.5, Input=5	Price=2.5, Input=5	Price=2.5, Input=5
State	Obj=MoneyProcessor Obj.container.coin1rmb=100 Obj.container.coinHalfrmb=100	Obj=MoneyProcessor Obj.container.coin1rmb=0 Obj.container.coinHalfrmb=5	Obj=MoneyProcessor Obj.container.coin1rmb=0 Obj.container.coinHalfrmb=0
Expected Output	[1,2]	[5,0]	[-1, 2.5, 0, 0]

- Test Coverage: 5/5
- Test Result: 3 passed

T1.1.2 Test updateContainer()

```
function status = updateContainer(obj, dCash1, dCash5, dCoinHalf, dCoin1)
    if obj.container.cash1rmb + dCash1 < 0 || obj.container.cash1rmb + dCash1 > obj.container.max(1)
        % Branch T cover 1.1.2.1
        status1 = 0;
    else
        % Branch T cover 1.1.2.2
        status1 = 1;
        obj.container.cash1rmb = obj.container.cash1rmb + dCash1;
    end

    if obj.container.cash5rmb + dCash5 < 0 || obj.container.cash5rmb + dCash5 > obj.container.max(2)
        % Branch T cover 1.1.2.3
        status2 = 0;
    else
        % Branch T cover 1.1.2.4
        status2 = 1;
        obj.container.cash5rmb = obj.container.cash5rmb + dCash5;
    end

    if obj.container.coinHalfrmb + dCoinHalf < 0 || obj.container.coinHalfrmb + dCoinHalf > obj.container.max(3)
        % Branch T cover 1.1.2.5
        status3 = 0;
    else
        % Branch T cover 1.1.2.6
        status3 = 1;
        obj.container.coinHalfrmb = obj.container.coinHalfrmb + dCoinHalf;
    end

    if obj.container.coin1rmb + dCoin1 < 0 || obj.container.coin1rmb + dCoin1 > obj.container.max(4)
        % Branch T cover 1.1.2.7
        status4 = 0;
    else
        % Branch T cover 1.1.2.8
        status4 = 1;
        obj.container.coin1rmb = obj.container.coin1rmb + dCoin1;
    end
    status = [status1, status2, status3, status4];
end
```

- Coverage Criteria: Branch coverage
- Test Case

	Test Case T1.1.2.1	Test Case T1.1.2.2
Coverage Item	T cover 1.1.2.1, T cover 1.1.2.3, T cover 1.1.2.5, T cover 1.2.2.7	T cover 1.1.2.2, T cover 1.1.2.4, T cover 1.1.2.6, T cover 1.2.2.8
Input	dCash1=1000,dCash5=1000, dCoinHalf=-1000, dCoin1=1000	dCash1=0, dCash5=0, dCoinHalf=0, dCoin1=0
State	Obj=MoneyProcessor Obj.container.coin1rmb=0 Obj.container.coinHalfrmb=5 Obj.container.cash1rmb=0 Obj.container.cash5rmb=5	Obj=MoneyProcessor Obj.container.coin1rmb=0 Obj.container.coinHalfrmb=5 Obj.container.cash1rmb=0 Obj.container.cash5rmb=5
Expected Output	[0,0,0,0]	[1,1,1,1]

- Test Coverage: 8/8
- Test Result: 2 passed

T1.1.3 Test getMoneyAmount()

```
function amount = getMoneyAmount(obj, moneyID)
    % 1 - cash 1r; 2 - cash 5r; 3 - coin 0.5r; 4 - coin 1r.
    if ismember([moneyID], [1 2 3 4])
        % Branch T cover 1.1.3.1
        switch(moneyID)
            case 1
                % Branch T cover 1.1.3.2
                amount = obj.container.cash1rmb;
            case 2
                % Branch T cover 1.1.3.3
                amount = obj.container.cash5rmb;
            case 3
                % Branch T cover 1.1.3.4
                amount = obj.container.coinHalfrmb;
            case 4
                % Branch T cover 1.1.3.5
                amount = obj.container.coin1rmb;
        end
    else
        % Branch T cover 1.1.3.6
        amount = "wrong";
    end
end
```

- Coverage Criteria: Branch coverage
- Test Case

	Test Case T1.1.3.1	Test Case T1.1.3.2	Test Case T1.1.3.3	Test Case T1.1.3.4	Test Case T1.1.3.5
Coverage Item	T cover 1.1.3.1, T cover 1.1.3.2,	T cover 1.1.3.1, T cover 1.1.3.3	T cover 1.1.3.1, T cover 1.1.3.4	T cover 1.1.3.1, T cover 1.1.3.5	T cover 1.1.3.6
Input	MoneyID=1	MoneyID=2	MoneyID=3	MoneyID=4	MoneyID=5
State	Obj.container. coin1rmb=12 Obj.container. coinHalfrmb= 34 Obj.container. cash1rmb=56 Obj.container. cash5rmb=78	Obj.container. coin1rmb=12 Obj.container. coinHalfrmb= 34 Obj.container. cash1rmb=56 Obj.container. cash5rmb=78	Obj.container. coin1rmb=12 Obj.container. coinHalfrmb= 34 Obj.container. cash1rmb=56 Obj.container. cash5rmb=78	Obj.container. coin1rmb=12 Obj.container. coinHalfrmb= 34 Obj.container. cash1rmb=56 Obj.container. cash5rmb=78	Obj.container. coin1rmb=12 Obj.container. coinHalfrmb= 34 Obj.container. cash1rmb=56 Obj.container. cash5rmb=78
Expected Output	12	34	56	78	“wrong”

- Test Coverage: 6/6
- Test Result: 5 passed

T1.1.4. Test addCoin1()

```
function addCoin1(obj)
    % Statement T cover  1.1.4.1
   obj.container.coin1rmb = obj.container.coin1rmb + 1;
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.1.4.1
Coverage Item	T cover 1.1.4.1
Input	--
State	Obj=MoneyProcessor Obj.container.coin1rmb=0 Obj.container.coinHalfrmb=5 Obj.container.cash1rmb=0 Obj.container.cash5rmb=5
Expected Output	---

- Test Coverage: 1/1
- Test Result: 1 passed

T1.1.5. Test addCash1()

```
function addCash1(obj)
    % Statement T cover  1.1.5.1
   obj.container.cash1rmb = obj.container.cash1rmb + 1;
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.1.5.1
Coverage Item	T cover 1.1.5.1
Input	--
State	Obj=MoneyProcessor Obj.container.coin1rmb=0 Obj.container.coinHalfrmb=5 Obj.container.cash1rmb=0 Obj.container.cash5rmb=5
Expected Output	---

- Test Coverage: 1/1
- Test Result: 1 passed

T1.1.6 Test addCash5()

```
function addCash5(obj)
    % Statement T cover  1.1.6.1
   obj.container.cash5rmb = obj.container.cash5rmb + 1;
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.1.6.1
Coverage Item	T cover 1.1.6.1
Input	--
State	Obj=MoneyProcessor Obj.container.coin1rmb=0 Obj.container.coinHalfrmb=5 Obj.container.cash1rmb=0 Obj.container.cash5rmb=5
Expected Output	---

- Test Coverage: 1/1
- Test Result: 1 passed

T1.1.7 Test addCoinHalf()

```
function addCoinHalf(obj)
    % Statement T cover  1.1.7.1
   obj.container.coinHalfrmb = obj.container.coinHalfrmb + 1;
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.1.7.1
Coverage Item	T cover 1.1.7.1
Input	--
State	Obj=MoneyProcessor Obj.container.coin1rmb=0 Obj.container.coinHalfrmb=5 Obj.container.cash1rmb=0 Obj.container.cash5rmb=5
Expected Output	---

- Test Coverage: 1/1
- Test Result: 1 passed

T1.2 Cell System + Cells Unit Test

T1.2.1 Test getPrice()

```
function price=getPrice(obj, cid)
    % Statement T cover 1.2.1.1
    price = obj.cells(cid).price;
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.2.1.1
Coverage Item	T cover 1.2.1.1
Input	--
State	Obj=CellsSystem Obj.cells=[obj.cell1, obj,cell2, obj.cell3, obj.cell4] Obj.cell1= cell(1, 23, 56)
Expected Output	23

- Test Coverage: 1/1
- Test Result: 1 passed

T1.2.2 Test getInventory()

```
function inventory=getInventory(obj, cid)
    % Statement T cover 1.2.2.1
    inventory = obj.cells(cid).inventory;
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.2.2.1
Coverage Item	T cover 1.2.2.1
Input	--
State	Obj=CellsSystem Obj.cells=[obj.cell1, obj,cell2, obj.cell3, obj.cell4] Obj.cell1= cell(1, 23, 56)
Expected Output	56

- Test Coverage: 1/1
- Test Result: 1 passed

T1.2.3 Test checkEmpty()

```
function isEmpty=checkEmpty(obj, cid)
    % Statement T cover 1.2.3.1
    isEmpty = obj.cells(cid).inventory==0;
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.2.3.1
Coverage Item	T cover 1.2.3.1
Input	--
State	Obj=CellsSystem Obj.cells=[obj.cell1, obj,cell2, obj.cell3, obj.cell4] Obj.cell1= cell(1, 23, 56)
Expected Output	0

- Test Coverage: 1/1
- Test Result: 1 passed

T1.2.4 Test updateInventory()

```
function status = updateInventory(obj, cid, dInventory)
    if obj.cells(cid).inventory + dInventory < 0 || obj.cells(cid).inventory + dInventory > obj.cells(cid).max
                % Branch T cover 1.2.4.1
        status = 0;
    else
                % branch T cover 1.2.4.2
        obj.cells(cid).inventory = obj.cells(cid).inventory + dInventory;
        status = 1;
    end
end
```

- Coverage Criteria: Branch coverage
- Test Case

	Test Case T1.2.4.1	Test Case T1.2.4.2
Coverage Item	T cover 1.2.4.1	T cover 1.2.4.2
Input	cid=1, dInventory=0	cid=1, dInventory=-100
State	Obj=CellsSystem Obj.cells=[obj.cell1, obj,cell2, obj.cell3, obj.cell4] Obj.cell1= cell(1, 23, 56)	Obj=CellsSystem Obj.cells=[obj.cell1, obj,cell2, obj.cell3, obj.cell4] Obj.cell1= cell(1, 23, 56)
Expected Output	0	1

- Test Coverage: 2/2
- Test Result: 2 passed

T1.3 Log System Unit Test

T1.3.1 Test addOneRecord()

```
function addOneRecord(obj, record)
    if obj.currentLog == -1
        % branch T cover 1.3.1.1
        obj.currentLog = record;
    else
        % branch T cover 1.3.1.2
        obj.currentLog = [obj.currentLog;record];
    end
end
```

- Coverage Criteria: Branch coverage
- Test Case

	Test Case T1.3.1.1	Test Case T1.3.1.2
Coverage Item	T cover 1.3.1.1	T cover 1.3.1.2
Input	record=[618121212, 1, 2.5, 5, 0, 1]	Record=[618121212, 1, 2.5, 5, 0, 1]
State	Obj=logSys Obj.currentLog=-1	Obj=logSys Obj.currentLog=[618121211, 1, 2.5, 5, 0, 1]
Expected Output	Obj.currentlog = [618121212, 1, 2.5, 5, 0, 1]	Obj.currentLog=[618121211, 1, 2.5, 5, 0, 1; 618121212, 1, 2.5, 5, 0, 1]

- Test Coverage: 2/2
- Test Result: 2 passed

T1.3.2 Test saveCurrentLog2Excel()

```
function saveCurrentLog2Excel(obj, filename)
    % statement T cover 1.3.2.1
    title = ["Time Stamp" "Product ID" "Product Price" "Input Money" "Remain Change" "Get Product"];
    data = obj.currentLog;
    xlswrite(filename, [title; data]);
    obj.currentLog = -1;
    obj.onlinePanel.UITable.Data = [];
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.3.2.1
Coverage Item	T cover 1.3.2.1
Input	Filename=”hehe”
State	Obj=logSys Obj.currentLog=[618121211, 1, 2.5, 5, 0, 1]
Expected Output	A file named hehe.xlsx is saved; obj.onlinePanel.UITable.Data=[]

- Test Coverage: 1/1
- Test Result: 1 passed

T1.3.3 Test refreshOnlinePanel()

```
function refreshOnlinePanel(obj)
    % statement T cover 1.3.1.3
    obj.onlinePanel.UITable.Data = obj.currentLog;
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.3.3.1
Coverage Item	T cover 1.3.3.1
Input	--
State	Obj=logSys Obj.currentLog=[618121211, 1, 2.5, 5, 0, 1]
Expected Output	obj.onlinePanel.UITable.Data=[618121211, 1, 2.5, 5, 0, 1]

- Test Coverage: 1/1
- Test Result: 1 passed

T1.4 Purchase Unit Test

T1.4.1 Test GenrateOneRecord()

```
function record = generateOneRecord(obj, inputMoney)
    % Statement T cover 1.4.1.1
    % ["Time Stamp" "Product ID" "Product Price" "Input Money" "Remain Change" "Get Product"]
    a = clock;
    timeStamp = a(2)*100000000 + a(3)*1000000 + a(4)*10000 +a(5)*100 + int64(a(6));
    record = [timeStamp, obj.productID, obj.productPrice, inputMoney, obj.remainChange, obj.getProduct];
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.4.1.1
Coverage Item	T cover 1.4.1.1
Input	inoutMoney=5
State	Obj=purchase Obj.productID=1, obj.productPrice=2.5, inputMoney=5, obj.remainChange=0, obj.getProduct=1
Expected Output	record=[~, 1, 2.5, 5, 0, 1]

- Test Coverage: 1/1
- Test Result: 1 passed

T1.5 Maintain System Unit Test

T1.5.1 Test fixUserProblem()

```
function flag = fixUserProblem(obj)
    % statement T cover 1.5.1.1
    flag = 1;
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.5.1.1
Coverage Item	T cover 1.5.1.1
Input	---
State	Obj=maintainSys
Expected Output	Flag=1

- Test Coverage: 1/1
- Test Result: 1 passed

T1.6 User Panel Unit Test

T1.6.1 Test setYouChosePartEditability()

```
function setYouChosePartEditability(app, flag)
    % make the 'You chose' part look editable(1) or uneditable(0).
    if flag == 1
        % brach T cover 1.6.1.1
        app.productEditField.BackgroundColor = [1.0 1.0 1.0];
        app.ItcostsEditField.BackgroundColor = [1.0 1.0 1.0];
    elseif flag == 0
        % brach T cover 1.6.1.2
        app.productEditField.BackgroundColor = [0.9 0.9 0.9];
        app.ItcostsEditField.BackgroundColor = [0.9 0.9 0.9];
    end
end
```

- Coverage Criteria: Branch coverage
- Test Case

	Test Case T1.6.1.1	Test Case T1.6.1.2
Coverage Item	T cover 1.6.1.1	T cover 1.6.1.2
Input	Flag=1	Flag=0
State	App=Panel	App=panel
Expected Output	The background color of “product” and “it costs” in the Display panel become white	The background color of “product” and “it costs” in the Display panel become grey.

- Test Coverage: 2/2
- Test Result: 2 passed

T1.6.2 Test setYouHaveInsertedPartEditability()

```
function setYouHaveInsertedPartEditability(app, flag)
    % make the 'You have inserted' part look editable(1) or uneditable(0).
    if flag == 1
        % brach T cover 1.6.2.1
        app.coin1display.BackgroundColor = [1.0 1.0 1.0];
        app.cash1display.BackgroundColor = [1.0 1.0 1.0];
        app.cash5display.BackgroundColor = [1.0 1.0 1.0];
        app.coinHalfDisplay.BackgroundColor = [1.0 1.0 1.0];
        app.TotalmoneyinsertedEditField.BackgroundColor = [1.0 1.0 1.0];
    elseif flag == 0
        % brach T cover 1.6.2.2
         app.coin1display.BackgroundColor = [0.9 0.9 0.9];
         app.cash1display.BackgroundColor = [0.9 0.9 0.9];
         app.cash5display.BackgroundColor = [0.9 0.9 0.9];
         app.coinHalfDisplay.BackgroundColor = [0.9 0.9 0.9];
         app.TotalmoneyinsertedEditField.BackgroundColor = [0.9 0.9 0.9];
    end
end
```

- Coverage Criteria: Branch coverage
- Test Case

	Test Case T1.6.2.1	Test Case T1.6.2.2
Coverage Item	T cover 1.6.2.1	T cover 1.6.2.2
Input	Flag=1	Flag=0
State	App=Panel	App=panel
Expected Output	The background color of money display texts (separate&total) become white, making it look editable	The background color of money display texts (separate&total) become grey, making it look uneditable

- Test Coverage: 2/2
- Test Result: 2 passed

T1.6.3 Test resetDisplayRegion()

```
function resetDisplayRegion(app)
    % statement T cover 1.6.3.1
    app.TotalmoneyinsertedEditField.Value = 0;
    app.cash1display.Value = 0;
    app.cash5display.Value = 0;
    app.coinHalfDisplay.Value = 0;
    app.coin1display.Value = 0;
    app.ItcostsEditField.Value = 0;
    app.productEditField.Value = 0;
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.6.3.1
Coverage Item	T cover 1.6.3.1
Input	---
State	App=panel

Expected Output	The value in the display part are all reset to 0.

- Test Coverage: 1/1
- Test Result: 1 passed

T1.6.4 Test resetMoneyRegion()

```
function resetMoneyRegion(app)
    % statement T cover 1.6.4.1
   app.InsertCoinButton.Enable = "off";
   app.InsertCashButton.Enable = "off";
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.6.4.1
Coverage Item	T cover 1.6.4.1
Input	---
State	App=panel
Expected Output	The insert buttons are not allowed to be pressed

- Test Coverage: 1/1
- Test Result: 1 passed

T1.6.5 Test resetOutputRegion()

```
function resetOutputRegion(app)
    % statement T cover 1.6.5.1
    app.ChangeoutLamp.Color = [0.8 0.8 0.8];
    app.ProductOutLamp.Color = [0.8 0.8 0.8];
    app.changeCoin1display.Value = 0;
    app.changeCoinHalfDisplay.Value = 0;
    app.changeCash1rDisplay.Value = 0;
    app.changeCash5rDisplay.Value = 0;
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.6.5.1
Coverage Item	T cover 1.6.5.1
Input	---
State	App=panel
Expected Output	The value in the output part are all reset to 0; the indication lamps are set to grey.

- Test Coverage: 1/1
- Test Result: 1 passed

T1.7 Offline Maintain Panel Unit Test

T1.7.1 Test inventoryLightIndicator()

```
function color = inventoryLightIndicator(app, inventory)
    if inventory ==  0
                    % branch T cover 1.7.1.1
            color = [1 0 0];
    elseif inventory < 5 && inventory > 0
                    % branch T cover 1.7.1.2
        color = [1 1 0.07];
    elseif inventory >= 5 && inventory <= 95
                    % branch T cover 1.7.1.3
        color = [0 1 0];
    elseif inventory > 95 && inventory < 100
                    % branch T cover 1.7.1.4
        color = [0 0 1];
    elseif inventory == 100
                    % branch T cover 1.7.1.5
        color = [0 0 0];
    else
                    % branch T cover 1.7.1.6
        % should not happen
        color = [1 1 1];
    end
end
```

- Coverage Criteria: Branch coverage
- Test Case

	Test Case T1.7.1.1	Test Case T1.7.1.2	Test Case T1.7.1.3	Test Case T1.7.1.4	Test Case T1.7.1.5	Test Case T1.7.1.6
Coverage Item	T cover 1.7.1.1	T cover 1.7.1.2	T cover 1.7.1.3	T cover 1.7.1.4	T cover 1.7.1.5	T cover 1.7.1.6
Input	Inventory=0	Inventory=3	Inventory=32	Inventory=99	Inventory=100	Inventory=2333
State	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel
Expected Output	[1 0 0](Red)	[1 1 0.7](yellow)	[0 1 0](green)	[0 0 1](blue)	[0 0 0](black)	[1 1 1] white

- Test Coverage: 6/6
- Test Result: 6 passed

T1.7.2 Test moneyLightIndicator()

```
function color = moneyLightIndicator(app, amount)
    if amount ==  0
        % branch T cover 1.7.2.1
            color = [1 0 0];
    elseif amount < 50 && amount > 0
        % branch T cover 1.7.2.2
        color = [1 1 0.07];
    elseif amount >= 50 && amount <= 950
        % branch T cover 1.7.2.3
        color = [0 1 0];
    elseif amount > 950 && amount < 1000
        % branch T cover 1.7.2.4
        color = [0 0 1];
    elseif amount == 1000
        % branch T cover 1.7.2.5
        color = [0 0 0];
    else
        % branch T cover 1.7.2.6
        % should not happen
        color = [1 1 1];
    end
end
```

- Coverage Criteria: Branch coverage
- Test Case

	Test Case T1.7.2.1	Test Case T1.7.2.2	Test Case T1.7.2.3	Test Case T1.7.2.4	Test Case T1.7.2.5	Test Case T1.7.2.6
Coverage Item	T cover 1.7.2.1	T cover 1.7.2.2	T cover 1.7.2.3	T cover 1.7.2.4	T cover 1.7.2.5	T cover 1.7.2.6
Input	amount=0	amount=30	amount=320	amount=990	amount=1000	Inventory=2333
State	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel
Expected Output	[1 0 0](Red)	[1 1 0.7](yellow)	[0 1 0](green)	[0 0 1](blue)	[0 0 0](black)	[1 1 1] white

- Test Coverage: 6/6
- Test Result: 6 passed

T1.7.3 Test updateCellLamp()

```
% lamp indication
function updateCellLamp(app, inventory1, inventory2, inventory3, inventory4)
    % statement t cover 1.7.3.1
    %i = [inventory1, inventory2, inventory3, inventory4];
    app.inventory1.Value = inventory1;
    app.inventory2.Value = inventory2;
    app.inventory3.Value = inventory3;
    app.inventory4.Value = inventory4;

    app.Lamp1.Color = app.inventoryLightIndicator(inventory1);
    app.Lamp2.Color = app.inventoryLightIndicator(inventory2);
    app.Lamp3.Color = app.inventoryLightIndicator(inventory3);
    app.Lamp4.Color = app.inventoryLightIndicator(inventory4);
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.7.3.1
Coverage Item	T cover 1.7.3.1, T cover 1.7.1.2, T cover 1.7.1.3, T cover 1.7.1.4, T cover 1.7.1.5
Input	1,20,99,100
State	App= OfflineMaintainPanel
Expected Output	The inventory indication of 1,2,3,4 becomes yellow, green, blue, black

- Test Coverage: 1/1
- Test Result: 1 passed

T1.7.4 Test updateMoneyLamp()

```
% lamp indication
function updateMoneyLamp(app, cash1r, cash5r, coinHalfr, coin1r)
     % statement t cover 1.7.4.1
    %m = [cash1r, cash5r, coinHalfr, coin1r];
    app.AmountEditField.Value = cash1r;
    app.AmountEditField_2.Value = cash5r;
    app.AmountEditField_3.Value = coinHalfr;
    app.AmountEditField_4.Value = coin1r;

    app.MoneyLamp1.Color = app.moneyLightIndicator(cash1r);
    app.MoneyLamp2.Color = app.moneyLightIndicator(cash5r);
    app.MoneyLamp3.Color = app.moneyLightIndicator(coinHalfr);
    app.MoneyLamp4.Color = app.moneyLightIndicator(coin1r);
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.7.4.1
Coverage Item	T cover 1.7.4.1, T cover 1.7.2.2, T cover 1.7.2.3, T cover 1.7.2.4, T cover 1.7.2.5

Input	1,200,990,1000
State	App= OfflineMaintainPanel
Expected Output	The inventory indication of 1,2,3,4 becomes yellow, green, blue, black

- Test Coverage: 1/1
- Test Result: 1 passed

T1.7.5 Test deltaCell()

```
function deltaCell(app, cellID, sign)
    if sign == 1
        % branch T cover 1.7.5.1
        name = "app.Replenish"+string(cellID)+".Value";
    else
        % branch T cover 1.7.5.2
        name = "app.Withdraw"+string(cellID)+".Value";
    end

    status = app.vending.cellsSys.updateInventory(cellID, eval(string(sign)+"*"+name));
    if status == 1
        % branch T cover 1.7.5.3
        eval(name + "=0");
        app.vending.updateOfflineMaintainPanel();
        app.vending.updateOnlineMaintainPanel(-1);
        app.vending.updateUserPanelPhysically();
    end
end
```

- Coverage Criteria: Branch coverage
- Test Case

	Test Case T1.7.5.1	Test Case T1.7.5.2
Coverage Item	T cover 1.7.5.1, T cover 1.7.5.3	T cover 1.7.5.2, T cover 1.7.5.3
Input	sign=1, cellid=1	sign=0, cellid=1
State	App= OfflineMaintainPanel	App= OfflineMaintainPanel
Expected Output	The amount of product 1 on the offline maintain should be its original value+1	The amount of product 1 on the offline maintain should be its original value-1

- Test Coverage: 2/2
- Test Result: 2 passed

T1.7.6 Test deltaMoney()

```
function deltaMoney(app, moneyID, sign)
    if sign == 1
        % branch T cover 1.7.6.1
        name = "app.Replenish"+string(moneyID)+"_2.Value";
    else %sign == -1
        % branch T cover 1.7.6.2
        name = "app.Withdraw"+string(moneyID)+"_2.Value";
    end

    if moneyID == 1
        % branch T cover 1.7.6.3
        status = app.vending.moneyProcessor.updateContainer(eval(string(sign)+"*"+name),0,0,0);
    elseif moneyID == 2
        % branch T cover 1.7.6.4
        status = app.vending.moneyProcessor.updateContainer(0,eval(string(sign)+"*"+name),0,0);
    elseif moneyID == 3
        % branch T cover 1.7.6.5
        status = app.vending.moneyProcessor.updateContainer(0,0,eval(string(sign)+"*"+name),0);
    else %moneyID == 4
        % branch T cover 1.7.6.6
        status = app.vending.moneyProcessor.updateContainer(0,0,0,eval(string(sign)+"*"+name));
    end

    if status(moneyID) == 1
        % branch T cover 1.7.6.7
        eval(name + "=0");
        app.vending.updateOfflineMaintainPanel();
        app.vending.updateOnlineMaintainPanel(-1);
        app.vending.updateUserPanelPhysically();
    end
end
```

- Coverage Criteria: Branch coverage
- Test Case

	Test Case T1.7.6.1	Test Case T1.7.6.2
Coverage Item	T cover 1.7.6.1, T cover 1.7.6.3-6 , T cover 1.7.6.7	T cover 1.7.6.2, T cover 1.7.6.3-6 , T cover 1.7.6.7
Input	sign=1, cellid=1	sign=0, cellid=1
State	App= OfflineMaintainPanel	App= OfflineMaintainPanel
Expected Output	The amount of all products on the offline maintain should be its original value+1	The amount of all products on the offline maintain should be its original value-1

- Test Coverage: 7/7
- Test Result: 2 passed

T1.8 Online Maintain Panel Unit Test

T1.8.1 Test inventoryLightIndicator()

```
function color = inventoryLightIndicator(app, inventory)
    if inventory ==  0
                    % branch T cover 1.8.1.1
            color = [1 0 0];
    elseif inventory < 5 && inventory > 0
                    % branch T cover 1.8.1.2
        color = [1 1 0.07];
    elseif inventory >= 5 && inventory <= 95
                    % branch T cover 1.8.1.3
        color = [0 1 0];
    elseif inventory > 95 && inventory < 100
                    % branch T cover 1.8.1.4
        color = [0 0 1];
    elseif inventory == 100
                    % branch T cover 1.8.1.5
        color = [0 0 0];
    else
                    % branch T cover 1.8.1.6
        % should not happen
        color = [1 1 1];
    end
end
```

- Coverage Criteria: Branch coverage
- Test Case

	Test Case T1.7.1.1	Test Case T1.8.1.2	Test Case T1.8.1.3	Test Case T1.8.1.4	Test Case T1.8.1.5	Test Case T1.8.1.6
Coverage Item	T cover 1.8.1.1	T cover 1.8.1.2	T cover 1.8.1.3	T cover 1.8.1.4	T cover 1.8.1.5	T cover 1.8.1.6
Input	Inventory= 0	Inventory= 3	Inventory=32	Inventory= 99	Inventory= 100	Inventory= 2333
State	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel
Expected Output	[1 0 0](Red)	[1 1 0.7](yellow)	[0 1 0](green)	[0 0 1](blue)	[0 0 0](black)	[1 1 1] white

- Test Coverage: 6/6
- Test Result: 6 passed

T1.8.2 Test moneyLightIndicator()

```
function color = moneyLightIndicator(app, amount)
    if amount ==  0
        % branch T cover 1.8.2.1
            color = [1 0 0];
    elseif amount < 50 && amount > 0
        % branch T cover 1.8.2.2
        color = [1 1 0.07];
    elseif amount >= 50 && amount <= 950
        % branch T cover 1.8.2.3
        color = [0 1 0];
    elseif amount > 950 && amount < 1000
        % branch T cover 1.8.2.4
        color = [0 0 1];
    elseif amount == 1000
        % branch T cover 1.8.2.5
        color = [0 0 0];
    else
        % branch T cover 1.8.2.6
        % should not happen
        color = [1 1 1];
    end
end
```

- Coverage Criteria: Branch coverage
- Test Case

	Test Case T1.8.2.1	Test Case T1.8.2.2	Test Case T1.8.2.3	Test Case T1.8.2.4	Test Case T1.8.2.5	Test Case T1.8.2.6
Coverage Item	T cover 1.8.2.1	T cover 1.8.2.2	T cover 1.8.2.3	T cover 1.8.2.4	T cover 1.8.2.5	T cover 1.8.2.6
Input	amount=0	amount=30	amount=320	amount=990	amount=1000	Inventory=2333
State	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel	App=OfflineMaintainPanel
Expected Output	[1 0 0](Red)	[1 1 0.7](yellow)	[0 1 0](green)	[0 0 1](blue)	[0 0 0](black)	[1 1 1] white

- Test Coverage: 6/6
- Test Result: 6 passed

T1.8.3 Test updateCellLamp()

```
% lamp indication
function updateCellLamp(app, inventory1, inventory2, inventory3, inventory4)
    % statement t cover 18 .3.1
    %i = [inventory1, inventory2, inventory3, inventory4];
    app.inventory1.Value = inventory1;
    app.inventory2.Value = inventory2;
    app.inventory3.Value = inventory3;
    app.inventory4.Value = inventory4;

    app.Lamp1.Color = app.inventoryLightIndicator(inventory1);
    app.Lamp2.Color = app.inventoryLightIndicator(inventory2);
    app.Lamp3.Color = app.inventoryLightIndicator(inventory3);
    app.Lamp4.Color = app.inventoryLightIndicator(inventory4);
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.8.3.1
Coverage Item	T cover 1.8.3.1, T cover 1.8.1.2, T cover 1.8.1.3, T cover 1.8.1.4, T cover 1.8.1.5
Input	1,20,99,100
State	App= OfflineMaintainPanel
Expected Output	The inventory indication of 1,2,3,4 becomes yellow, green, blue, black

- Test Coverage: 1/1
- Test Result: 1 passed

T1.8.4 Test updateMoneyLamp()

```
% lamp indication
function updateMoneyLamp(app, cash1r, cash5r, coinHalfr, coin1r)
     % statement t cover 1.8.4.1
    %m = [cash1r, cash5r, coinHalfr, coin1r];
    app.AmountEditField.Value = cash1r;
    app.AmountEditField_2.Value = cash5r;
    app.AmountEditField_3.Value = coinHalfr;
    app.AmountEditField_4.Value = coin1r;

    app.MoneyLamp1.Color = app.moneyLightIndicator(cash1r);
    app.MoneyLamp2.Color = app.moneyLightIndicator(cash5r);
    app.MoneyLamp3.Color = app.moneyLightIndicator(coinHalfr);
    app.MoneyLamp4.Color = app.moneyLightIndicator(coin1r);
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.8.4.1
Coverage Item	T cover 1.8.4.1, T cover 1.8.2.2, T cover 1.8.2.3, T cover 1.8.2.4, T cover 1.8.2.5

Input	1,200,990,1000
State	App= OfflineMaintainPanel
Expected Output	The inventory indication of 1,2,3,4 becomes yellow, green, blue, black

- Test Coverage: 1/1
- Test Result: 1 passed

T1.9 Vending Test

T1.9.1 Test updateUserPanel Physically()

```
% User Panel
function updateUserPanelPhysically(obj)
    % Inventory part
    inventoryButton = [obj.userPanel.IwantthisButton, obj.userPanel.IwantthisButton_2, obj.userPanel.IwantthisButton_3, obj.userPanel.IwantthisButton_4];
    for i = 1:4
         if obj.cellsSys.getInventory(i) > 0
             % branch T cover - 1.9.1.1
             inventoryButton(i).Enable = "on";
             eval("obj.userPanel.Lamp"+string(i)+".Color = [0 1 0]");
         else
             % branch T cover - 1.9.1.2
             inventoryButton(i).Enable = "off";
             eval("obj.userPanel.Lamp"+string(i)+".Color = [1 0 0]");
         end
    end
    % Money part
    moneyButton = [obj.userPanel.cash1, obj.userPanel.cash5, obj.userPanel.coinHalf, obj.userPanel.coin1];
    for i = 1:4
        if obj.moneyProcessor.getMoneyAmount(i) < 1000
            % branch T cover - 1.9.1.3
            moneyButton(i).Enable = "on";
        else
            % branch T cover - 1.9.1.4
            moneyButton(i).Enable = "off";
        end
    end
end
```

- Coverage Criteria: Branch coverage
- Test Case

	Test Case T1.9.1.1	Test Case T1.9.1.2
Coverage Item	T cover 1.9.1.1, T cover 1.9.1.3	T cover 1.9.1.2, T cover 1.9.1.4
Input	---	---
State	Obj=Vending Obj.cells(1).inventory=0 Obj.moneyProcessor.cash1num=0	Obj=Vending Obj.cells(1).inventory=1 Obj.moneyProcessor.cash1num=1
Expected Output	The first product is not allowed to be selected, insertion allowed for cash 1.	The first product is allowed to be selected, insertion allowed for cash 1.

- Test Coverage: 4/4
- Test Result: 2 passed

T1.9.2 Test updateOnlineMaintainPanel()

```
% Online Maintain Panel
function updateOnlineMaintainPanel(obj, record)
    if ~isequal(-1,record)
        % branch T cover 1.9.2.1.1
        obj.logSys.addOneRecord(record);
        obj.logSys.refreshOnlinePanel();
    end
    % statement T cover 1.9.2.2.1
    obj.maintainSys.onlinePanel.updateCellLamp(obj.cellsSys.getInventory(1), obj.cellsSys.getInventory(2), ...
        obj.cellsSys.getInventory(3), obj.cellsSys.getInventory(4));
    obj.maintainSys.onlinePanel.updateMoneyLamp(obj.moneyProcessor.getMoneyAmount(1), obj.moneyProcessor.getMoneyAmount(2), ...
        obj.moneyProcessor.getMoneyAmount(3), obj.moneyProcessor.getMoneyAmount(4));
end
```

- Coverage Criteria: Branch coverage, Statement Coverage
- Test Case

	Test Case T1.9.2.1
Coverage Item	T cover 1.9.2.1.1, T cover 1.9.2.2.1, T cover 1.8.1.1, T cover 1.8.2.1
Input	Record=[618121211, 1, 2.5, 5, 0, 1]
State	Obj=Vending [Obj.cells(i).inventory=0 for i in 4] Pro= Obj.moneyProcessor : pro.container.coin1rmb=0, pro.container.coinHalfrmb=0, pro.container.cash1rmb=0, pro.container.cash5rmb=0
Expected Output	All indication lamps of cell and money on online maintain panel are red; One record [618121211, 1, 2.5, 5, 0, 1] is added to the last row of the online maintain panel.

- Test Coverage: 2/2
- Test Result: 1 passed

T1.9.3 TestOfflineMaintainPanel()

```
% Offline Maintain Panel
function updateOfflineMaintainPanel(obj)
    % statement T cover 1.9.3.1
    obj.maintainSys.offlinePanel.updateCellLamp(obj.cellsSys.getInventory(1), obj.cellsSys.getInventory(2), ...
        obj.cellsSys.getInventory(3), obj.cellsSys.getInventory(4));
    obj.maintainSys.offlinePanel.updateMoneyLamp(obj.moneyProcessor.getMoneyAmount(1), obj.moneyProcessor.getMoneyAmount(2), ...
        obj.moneyProcessor.getMoneyAmount(3), obj.moneyProcessor.getMoneyAmount(4));
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.9.3.1
Coverage Item	T cover 1.9.3.1, T cover 1.8.1.1, T cover 1.8.2.1
Input	1,200,990,1000
State	Obj=Vending [Obj.cells(i).inventory=0 for i in 4] Pro= Obj.moneyProcessor : pro.container.coin1rmb=0, pro.container.coinHalfrmb=0, pro.container.cash1rmb=0, pro.container.cash5rmb=0
Expected Output	All indication lamps of cell and money on online maintain panel are red and texts are 0's.

- Test Coverage: 1/1
- Test Result: 1 passed

T1.9.4 Test getUserProblem()

```
function getUserProblem(obj)
     % statement T cover 1.9.4.1
    obj.maintainSys.offlinePanel.UserproblemLamp.Color = [1 0 0];
    obj.maintainSys.onlinePanel.GetCalledbyUserLamp.Color = [1 0 0];

end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.9.4.1
Coverage Item	T cover 1.9.4.1
Input	---
State	Obj=Vending
Expected Output	All indication lamps user problem on 2 maintain panels become red.

- Test Coverage: 1/1
- Test Result: 1 passed

T1.9.5 Test fixUserProblem()

```
function fixUserProblem(obj)
     % statement T cover 1.9.5.1
    obj.maintainSys.fixUserProblem();
    obj.userPanel.CellsPanel.Enable = "on";
    obj.userPanel.MoneyInputPanel.Enable = "on";
    obj.userPanel.DisplayRegionPanel.Enable = "on";
    obj.maintainSys.offlinePanel.UserproblemLamp.Color = [0.8 0.8 0.8];
    obj.maintainSys.onlinePanel.GetCalledbyUserLamp.Color = [0.8 0.8 0.8];
end
```

- Coverage Criteria: Statement coverage
- Test Case

	Test Case T1.9.5.1
Coverage Item	T cover 1.9.5.1
Input	---
State	Obj=Vending
Expected Output	All indication lamps user problem on 2 maintain panels become grey.

- Test Coverage: 1/1
- Test Result: 1 passed

T1.9.6 Test finishOnePurchase()

```
% What need to be done after finishing one purchase:
function finishOnePurchase(obj, cid, dCash1, dCash5, dCoinHalf, dCoin1)
    % 0. update Cell (inventory) & Money Container
    if cid ~= -1
        % branch T cover 1.9.6.1.1
        obj.cellsSys.updateInventory(cid, -1);
    end
    % statement T cover 1.9.6.2.1
    obj.moneyProcessor.updateContainer(dCash1, dCash5, dCoinHalf, dCoin1);

    % 1. update online & offline maintain panel
    record = obj.userPanel.purchase.generateOneRecord(obj.userPanel.TotalmoneyinsertedEditField.Value);
    obj.updateOfflineMaintainPanel();
    obj.updateOnlineMaintainPanel(record);

    % 2. reset User Panel
    %   indicator according to physical amount
    obj.updateUserPanelPhysically();
    %   change & product output region reset
    obj.userPanel.resetOutputRegion();
    %   display region reset
    obj.userPanel.resetDisplayRegion();
    %   money input region reset
    obj.userPanel.resetMoneyRegion();
    %   set editable (look like at least)
    obj.userPanel.setYouChosePartEditability(1);
    obj.userPanel.setYouHaveInsertedPartEditability(1);
    % start a new purchase
    obj.userPanel.purchase = Purchase();
end
```

- Coverage Criteria: Branch coverage, Statement Coverage
- Test Case

	Test Case T1.9.6.1
Coverage Item	T cover 1.9.6.1.1, T cover 1.9.6.2.1, T cover 1.9.1.1-1.9.1.4, T cover 1.9.2.1.1, T cover 1.9.2.2.1, T cover 1.9.3.1, T cover 1.1.2.1, T cover 1.1.2.3, T cover 1.1.2.5, T cover 1.2.2.7, T cover 1.1.3.1-1.1.3.6
Input	[3, 0, 1, 0, 0]
State	Obj=Vending [Obj.cells(i).inventory=0 for i in 4] Pro= Obj.moneyProcessor : pro.container.coin1rmb=0, pro.container.coinHalfrmb=0, pro.container.cash1rmb=0, pro.container.cash5rmb=0
Expected Output	The User Panel is reset and the indication lamps and text are updated according to the amount in the containers.

- Test Coverage: 2/2
- Test Result: 1 passed

T2: Integration Test

T2.1 Vending+UserPanel+MaintainPanels+MoneyProcessor+MoneyContainer+Cellsys+Cells +MaintainSys Integration

```
function status = IntegrationTest1(tc)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % A. User Purchase and its effect %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %   a. Save value for checking
    product4_inventory = tc.vending.cellsSys.getInventory(4);
    cash5_amount = tc.vending.moneyProcessor.getMoneyAmount(2);
    coin1_amount = tc.vending.moneyProcessor.getMoneyAmount(4);
    %   b. Choosing product and cofirm
    tc.press(tc.userapp.IwantthisButton_4);
    tc.press(tc.userapp.ConfirmButton);
    %   c. Insert Money: cash5-cash5
    tc.press(tc.userapp.cash5);
    tc.press(tc.userapp.InsertCashButton);
    tc.press(tc.userapp.cash5);
    tc.press(tc.userapp.InsertCashButton);
    tc.press(tc.userapp.FinishInsertingButton);
    %   d. Fetch Change and product
    tc.press(tc.userapp.fetchallButton);
    %   A check: money container and cell system are updated correctly.
    statusA = isequal(tc.offlineapp.inventory4.Value, product4_inventory-1)&&...
              isequal(tc.offlineapp.AmountEditField_4.Value, coin1_amount-1);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % B. User Calls Maintainer and Maintainer's response %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    tc.press(tc.userapp.CallMaintainerButton);
    tc.press(tc.offlineapp.FixButton);
    %  B check: light off
    statusB = isequal(tc.offlineapp.UserproblemLamp.Color, [0.8 0.8 0.8]) && ...
              isequal(tc.onlineapp.GetCalledbyUserLamp.Color, [0.8 0.8 0.8]);

    %%%%%%%%%%%%%%%%
    % C: Maintain %
    %%%%%%%%%%%%%%%%
    %   1. Fill Cell 2
    product2_inventory = tc.vending.cellsSys.getInventory(2);
    tc.type(tc.offlineapp.Replenish2, 50);
    tc.press(tc.offlineapp.FillButton_2);
    %   2. Empty Cell 1
    product1_inventory = tc.vending.cellsSys.getInventory(1);
    tc.type(tc.offlineapp.Withdraw1,100);
    tc.press(tc.offlineapp.FetchButton);
    %   3. Fill cash 1
    cash1_amount = tc.vending.moneyProcessor.getMoneyAmount(1);
    tc.type(tc.offlineapp.Replenish1_2,100);
    tc.press(tc.offlineapp.FillButton_5);
    %   4. Fetch coin 1
    coin1_amount = tc.vending.moneyProcessor.getMoneyAmount(4);
    tc.type(tc.offlineapp.Withdraw4_2,100);
    tc.press(tc.offlineapp.FetchButton_8);
    %   C check
    statusC = isequal(tc.offlineapp.inventory2.Value, product2_inventory+50) && ...
              isequal(tc.offlineapp.inventory1.Value, product1_inventory-100) && ...
              isequal(tc.offlineapp.AmountEditField.Value, cash1_amount+100) && ...
              isequal(tc.offlineapp.AmountEditField_4.Value, coin1_amount-100);

    status = [statusA, statusB, statusC];
end
```

T2.1.1 Test IntegrationTest1(): buying, maintaining and interacting

- Test Case

	Test Case T3.1.1
Coverage Item	T cover 1.1.1.1, T cover 1.1.1.4, T cover 1.1.2.2, T cover 1.1.2.4, T cover 1.1.2.6, T cover 1.1.2.8, T cover 1.1.3.1-1.1.3.5, T cover 1.1.5.1, T cover 1.1.6.1, T cover 1.2.1.1, T cover 1.2.4.2, T 1.3.1.2, T cover 1.3.1.3, T cover 1.4.1.1, T cover 1.5.1.1, T cover 1.6.1.1, T cover1.6.1.2, T cover 1.6.2.1, T cover 1.6.2.2, T cover 1.6.3.1, T cover 1.6.4.1, T cover 1.6.5.1, T cover 1.7.1.1-1.7.1.5, T cover 1.7.2.1-1.7.2.5, T cover 1.7.3.1, T cover 1.7.4.1, T cover 1.7.5.1, T cover 1.7.5.2, T cover 1.7.5.3, T cover 1.7.6.1-1.7.6.7, T cover 1.8.1.1-1.8.1.5, T cover 1.5.2.1-1.8.2.5, T cover 1.8.3.1, T cover 1.8.4.1, T cover 1.9.1.1-1.9.1.4, T cover 1.9.2.1.1., T cover 1.9.2.2.1, T cover 1.9.4.1, T cover 1.9.5.1, T cover 1.9.6.1.1, T cover 1.9.6.2.1
Input	As shown in the annotation of code: User: Choose product 4, press confirm, insert 2 cash 5, press finish inserting, press fetch all, press call maintainer. Maintainer: Press fix user problem, fill cell2 with 50 product 2, fetch 100 from cell 1, fill 100 cash 1, fetch 100 coin1.

state	Obj=TestCase
	obj.vending = Vending(1,0.0);
	obj.userapp = obj.vending.userPanel;
	obj.offlineapp = obj.vending.maintainSys.offlinePanel;
	obj.onlineapp = obj.vending.maintainSys.onlinePanel;

Expected Output	Status=[1,1,1]

- Test Result: 1 passed
- Test Coverage: 70/70.

T3: Functional Test

T3.1 Use Case "User: Choose a product"

T3.1.1 Test choosing a product

- Test Case

	Test Case T3.1.1
State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty
Operation	Press "I want this" buttons of cell1, cell2, cell3 and cell4 respectively, and push "confirm" in "you choose" part.
Expected Behavior	The ID and price is updated in the display region of user panel correctly and in real time; When confirm button is pressed, the ID and price are frozen.

- Test Result: 1 passed

T3.2 Use Case "User: Insert Money"

T3.2.1 Test inserting fake

- Test Case

	Test Case T3.2.1
State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty, product 4 has just been chosen and confirmed. vending.moneyProcessor.coin1 = 500, vending.moneyProcessor.coinHalf = 500
Operation	Select "fake&illegal" in cash, press "insert", press "fetch" in cash and coin parts respectively.
Expected Behavior	A fake panel is out and the text on the "fake&illegal" side becomes 1 and lamp becomes green. Insertion is not allowed until "fetch" is pressed, leading to the lamp turning red and text becoming 0 as well.

- Test Result: 1 passed

T3.2.2 Test inserting legal coin

- Test Case

	Test Case T3.2.2
State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty vending.moneyProcessor.coin1 = 500, vending.moneyProcessor.coinHalf = 500
Operation	Product 4 is chosen and confirmed. Select "1r" in cash, press "insert"
Expected Behavior	The text in "total money inserted" and "cash:1r" increase by 1.

- Test Result: 1 passed

T3.2.3 Test inserting legal cash

- Test Case

	Test Case T3.2.2

State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty. vending.moneyProcessor.coin1 = 500, vending.moneyProcessor.coinHalf = 500
Operation	Product 4 is chosen and confirmed. Select “1r” in coin, press “insert”
Expected Behavior	The text in “total money inserted” and “coin:1r” increase by 1.

- Test Result: 1 passed

T3.3 Use Case “User: Get Change”

T3.3.1 Test getting change

- Test Case

	Test Case T3.3.1
State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty, vending.moneyProcessor.coin1 = 500, vending.moneyProcessor.coinHalf = 500
Operation	product 4 has been chosen and confirmed, two cash of 5r have been inserted and confirmed. View and press fetch all.
Expected Behavior	Product Out and change out lamps are green and coin1 text is 1. After pressing “fetch all”, lamps become grey and coin 1 text becomes 0.

- Test Result: 1 passed

T3.4 Use Case “User: Get Product”

T3.4.1 Test getting product

- Test Case

	Test Case T3.4.1
State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty, vending.moneyProcessor.coin1 = 500, vending.moneyProcessor.coinHalf = 500
Operation	product 4 has been chosen and confirmed, two cash of 5r have been inserted and confirmed. View and press fetch all.
Expected Behavior	Product Out and change out lamps are green and coin1 text is 1. After pressing “fetch all”, lamps become grey and coin 1 text becomes 0.

- Test Result: 1 passed

T3.5 Use Case “User: Call Maintainer”

T3.5.1 Test calling maintainer

- Test Case

	Test Case T3.3.1
State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty, vending.moneyProcessor.coin1 = 500, vending.moneyProcessor.coinHalf = 500

Operation	Press call maintainer
Expected Behavior	A lamp in each maintain panel indicating that user needs help becomes red.

- Test Result: 1 passed

T3.6 Use Case "Maintainer: Handle Money"

T3.6.1 Test checking money

- Test Case

	Test Case T3.6.1
State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty, vending.moneyProcessor.coin1 = 500, vending.moneyProcessor.coinHalf = 1000 vending.moneyProcessor.cash1=0, vending.moneyProcessor.cash5=0
Operation	Just Vending(1, 0.0) in command line and observe.
Expected Behavior	In maintain panels: The lamps of cells should be black, red, black, black and the number of inventory should be 100, 0, 100, 100. The price should be 2.5, 3, 5, 100. The lamps of money should be red, red, black, green, and the number should be 0, 0, 1000, 00.

- Test Result: 1 passed

T3.6.2 Test filling money

- Test Case

	Test Case T3.6.2
State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty, vending.moneyProcessor.coin1 = 500, vending.moneyProcessor.coinHalf = 1000 vending.moneyProcessor.cash1=0, vending.moneyProcessor.cash5=0
Operation	Type Vending(1, 0.0) in command line, write 100 in the replenish text of cash 1r and press "fill"
Expected Behavior	In maintain panels: The lamps of cells should be black, red, black, black and the number of inventory should be 100, 0, 100, 100. The price should be 2.5, 3, 5, 100. The lamps of money should be green, red, black, green, and the number should be 100, 0, 1000, 500.

- Test Result: 1 passed

T3.6.3 Test fetching money

- Test Case

	Test Case T3.6.3

State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty, vending.moneyProcessor.coin1 = 500, vending.moneyProcessor.coinHalf = 1000 vending.moneyProcessor.cash1=0, vending.moneyProcessor.cash5=0
Operation	Type Vending(1, 0.0) in command line, write 100 in the withdraw text of coin 1r and press “fetch”
Expected Behavior	In maintain panels: The lamps of cells should be black, red, black, black and the number of inventory should be 100, 0, 100, 100. The price should be 2.5, 3, 5, 100. The lamps of money should be red, red, black, green, and the number should be 0, 0, 1000, 400.

- Test Result: 1 passed

T3.7 Use Case “Maintainer: Handle Product”

T3.7.1 Test checking product

- Test Case

	Test Case T3.7.1
State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty, vending.moneyProcessor.coin1 = 500, vending.moneyProcessor.coinHalf = 1000 vending.moneyProcessor.cash1=0, vending.moneyProcessor.cash5=0
Operation	Just Vending(1, 0.0) in command line and observe.
Expected Behavior	In maintain panels: The lamps of cells should be black, red, black, black and the number of inventory should be 100, 0, 100, 100. The price should be 2.5, 3, 5, 100. The lamps of money should be red, red, black, green, and the number should be 0, 0, 1000, 500.

- Test Result: 1 passed

T3.7.2 Test filling money

- Test Case

	Test Case T3.7.2
State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty, vending.moneyProcessor.coin1 = 500, vending.moneyProcessor.coinHalf = 1000 vending.moneyProcessor.cash1=0, vending.moneyProcessor.cash5=0
Operation	Type Vending(1, 0.0) in command line, write 100 in the replenish text of cell2 and press “fill”
Expected Behavior	In maintain panels: The lamps of cells should be black, black, black, black and the number of inventory should be 100, 100, 100, 100. The price should be 2.5, 3, 5, 100.

	The lamps of money should be green, red, black, green, and the number should be 0, 0, 1000, 500.

- Test Result: 1 passed

T3.7.3 Test fetching money

- Test Case

	Test Case T3.7.3
State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty, vending.moneyProcessor.coin1 = 500, vending.moneyProcessor.coinHalf = 1000 vending.moneyProcessor.cash1=0, vending.moneyProcessor.cash5=0
Operation	Type Vending(1, 0.0) in command line, write 100 in the withdraw text of product 3 and press “fetch”
Expected Behavior	In maintain panels: The lamps of cells should be black, red, red, black and the number of inventory should be 100, 0, 0, 100. The price should be 2.5, 3, 5, 100. The lamps of money should be red, red, black, green, and the number should be 0, 0, 1000, 500.

Test Result: 1 passed

T3.8 Use Case “Maintainer: Handle Log”

T3.8.1 Test viewing log

- Test Case

	Test Case T3.8.1
State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty, vending.moneyProcessor.coin1 = 500, vending.moneyProcessor.coinHalf = 1000 vending.moneyProcessor.cash1=0, vending.moneyProcessor.cash5=0
Operation	Vending(1, 0.0) in command line, select product 1, confirm, insert cash 5r, push “finish inserting”, push “fetch all”
Expected Behavior	There is a new report on the online panel. This has been validated in T1.3.1 and 1.3.3.

- Test Result: 1 passed

T3.8.1 Test exporting log

- Test Case

	Test Case T3.8.2
State	Vending(1,0.0), cell1, cell3 and cell4 are full and cell 2 is empty, vending.moneyProcessor.coin1 = 500, vending.moneyProcessor.coinHalf = 1000 vending.moneyProcessor.cash1=0, vending.moneyProcessor.cash5=0

Operation	Vending(1, 0.0) in command line, on user panel, select product 1, confirm, insert cash 5r, push “finish inserting”, push “fetch all”. Push “clear and export to excel” in the online panel.
Expected Behavior	There is a new report on the online panel. This has been validated in T1.3.1 and 1.3.3. After pressing the “clear and export to excel”, the table is empty. This has been validated in T1.3.2.

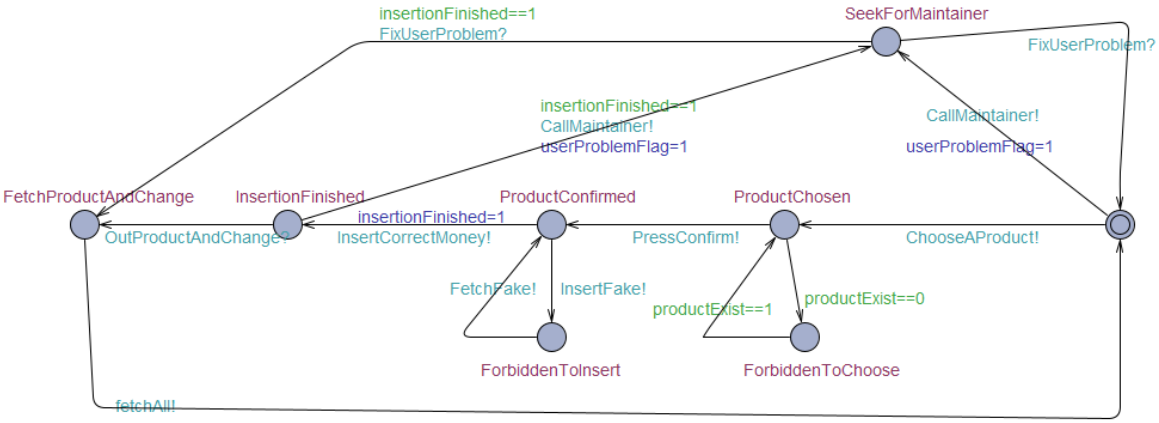
- Test Result: 1 passed

T4: Model Checking

A UPPAAL model of this vending machine is built for model checking.

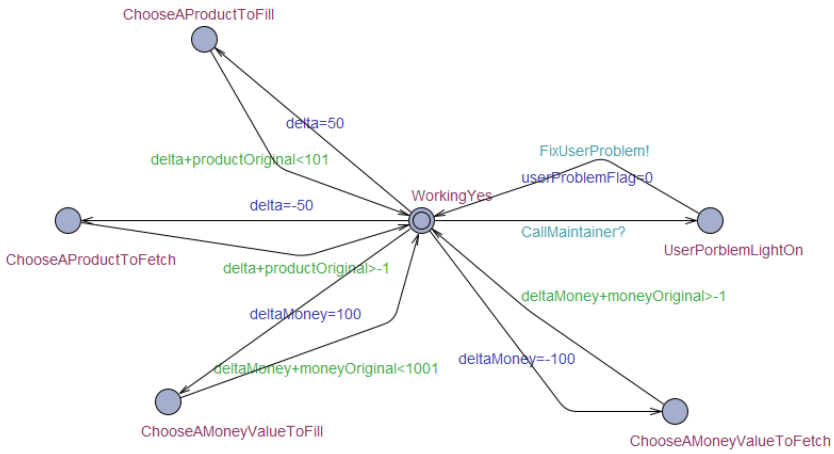
Full Model

User Model



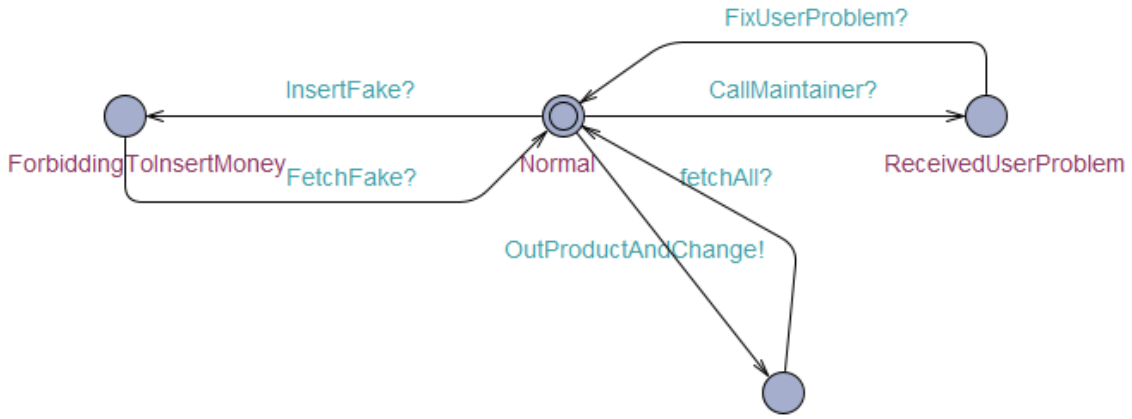
User can choose a existing product, insert legal cash and money, fetch product and change in a sequential order. When problem exists before choosing product (i.e. the product he/she want can not be chosen) or when he/she finishes inserting and cannot get product or change smoothly, he/she can seek for maintainer. The transfer condition is shown in the pic.

Maintainer Model



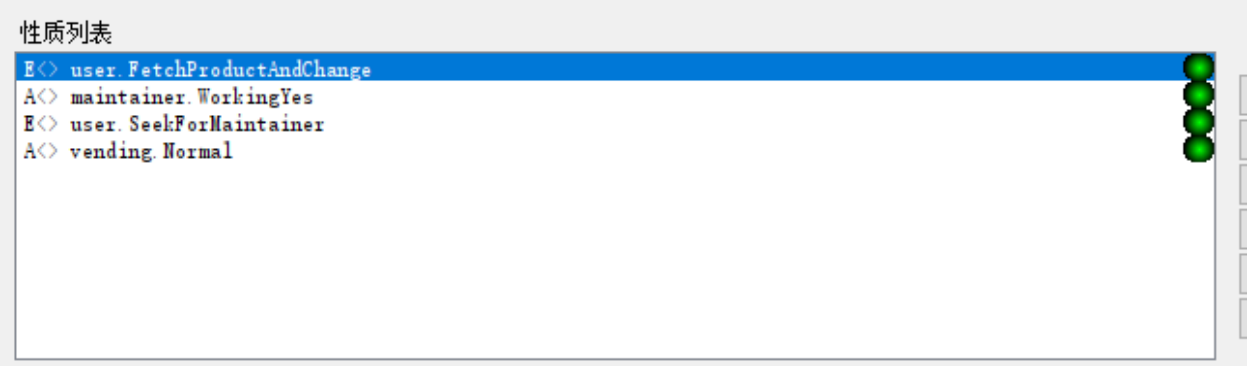
Maintainer has several states to reach. He/she can fill/fetch a certain product or cash when physic laws is observed. He/she can solve user's problem when user calls maintainer. The transfer condition is shown in the pic.

Vending Model



Vending can be of 4 states: normal, forbidding to insert money, forbidding to start a new purchase, receiving user's problem. The transfer condition is shown in the pic.

Check Properties



P4.1

Property	E<> user.FetchProductAndChange
Description	User can fetch product and change
Result	Pass

P4.2

Property	A<> maintainer.WorkingYes
Description	Maintainer can return to work smoothly all the time
Result	Pass

P4.3

Property	E<> user.SeekForMaintainer
Description	User can seek for maintainer
Result	Pass

P4.4

Property	A<> user.SeekForMaintainer
Description	Vending can return to normal all the time
Result	Pass