Student ID: 21180859

Smart contract for marketplaces in DAML

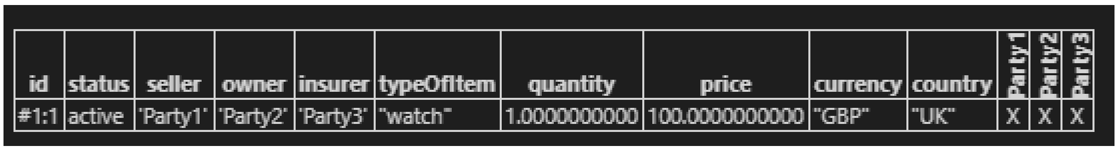
Contract code:

```
module Itemtosell where

template Itemtosell -- (2) Creating the new contract and specifying the
parties, price etc.
    with
        seller: Party
        owner: Party
        insurer: Party
        typeOfItem: Text
        quantity: Decimal
        price: Decimal
        currency: Text
        country: Text

    where                 -- (3) Defining the roles of the parties.
        signatory seller
        observer insurer

        ensure(               -- (4) Insuring the price and quantity are both
larges than zero
            quantity>0.0
            && price>0.0
            )

        controller owner can
            ChangePrice: (ContractId Itemtosell)  -- (5) Adding a function
where the owner can change the price

                with
                    newPrice: Decimal
                do
                    create this with
                        price = newPrice

            Sell: (ContractId Itemtosell)  -- (6) Adding a function where the
owner can sell the item

                with
                    newOwner: Party
                do
                    create this with
                        owner = newOwner
```

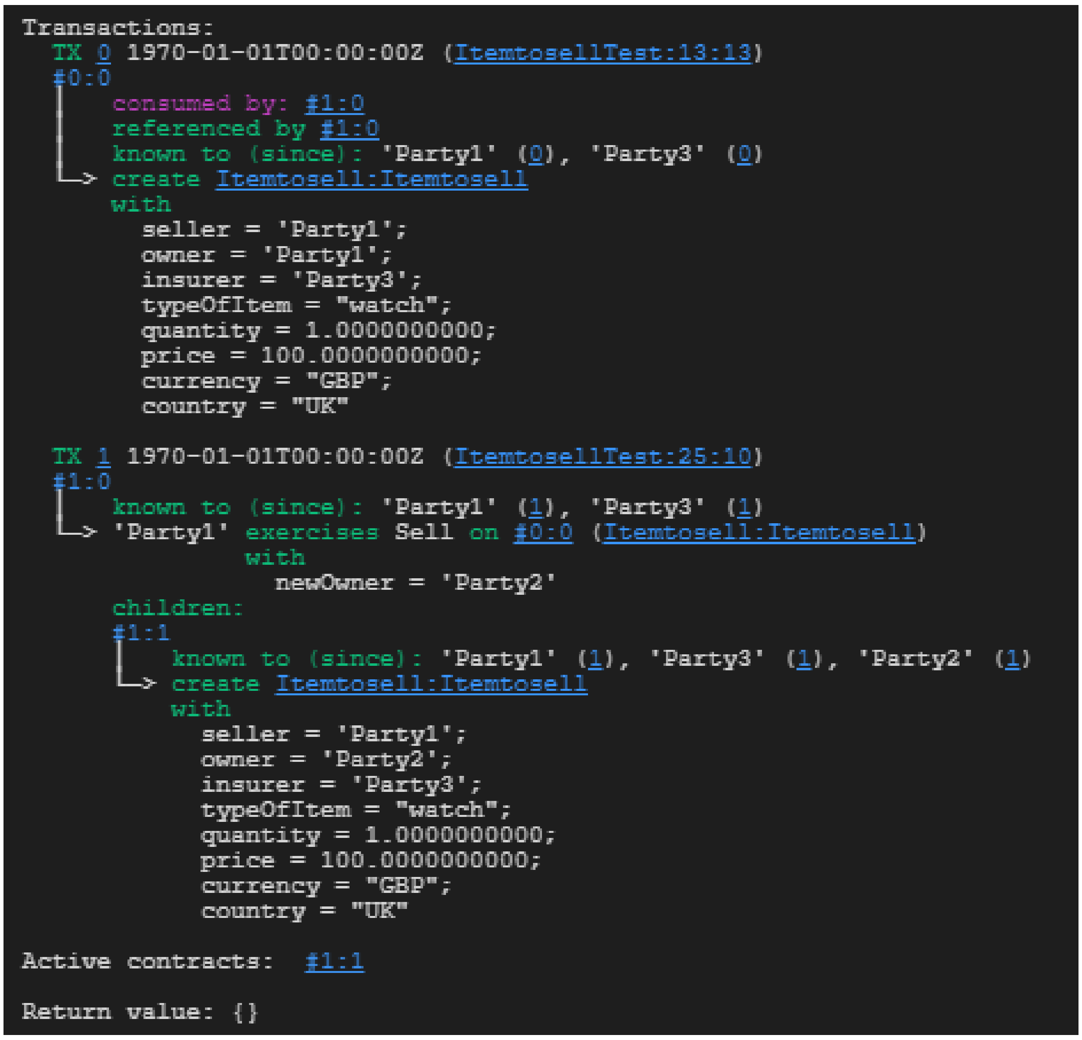
Scenario code:

```
module ItemtosellTest where
import Itemtosell

itemtosellTest : Scenario ()

itemtosellTest = scenario do

party1 <- getParty "Party1" -- (1) Creating the parties
party2 <- getParty "Party2"
party3 <- getParty "Party3"

sellItem <- submit party1 do -- (2) Issuing a new contract

    create Itemtosell with
        seller = party1
        owner = party1
        insurer = party3
        typeOfItem = "watch"
        quantity = 1.0
        price = 100.0
        currency = "GBP"
        country = "UK"

watch <- submit party1 do  -- (3) Party1 transfers the watch to Party2
    exercise sellItem Sell with
        newOwner = party2
return()
```

Ledger Screenshot:



id	status	seller	owner	insurer	typeOfItem	quantity	price	currency	country	Party1	Party2	Party3
#1:1	active	'Party1'	'Party2'	'Party3'	"watch"	1.0000000000	100.0000000000	"GBP"	"UK"	X	X	X



```
Transactions:
  TX 0 1970-01-01T00:00:00Z (ItemtosellTest:13:13)
  #0:0
  │   consumed by: #1:0
  │   referenced by #1:0
  │   known to (since): 'Party1' (0), 'Party3' (0)
  └─> create Itemtosell:Itemtosell
      with
        seller = 'Party1';
        owner = 'Party1';
        insurer = 'Party3';
        typeOfItem = "watch";
        quantity = 1.0000000000;
        price = 100.0000000000;
        currency = "GBP";
        country = "UK"

  TX 1 1970-01-01T00:00:00Z (ItemtosellTest:25:10)
  #1:0
  │   known to (since): 'Party1' (1), 'Party3' (1)
  └─> 'Party1' exercises Sell on #0:0 (Itemtosell:Itemtosell)
                with
                  newOwner = 'Party2'
      children:
      #1:1
      │   known to (since): 'Party1' (1), 'Party3' (1), 'Party2' (1)
      └─> create Itemtosell:Itemtosell
          with
            seller = 'Party1';
            owner = 'Party2';
            insurer = 'Party3';
            typeOfItem = "watch";
            quantity = 1.0000000000;
            price = 100.0000000000;
            currency = "GBP";
            country = "UK"

Active contracts:  #1:1

Return value: {}
```

An example where the execution of the contract would automatically fail would be if the insurer (party 3) tries to sell the item of the seller (party 1). So, instead of this code (which is the working one):

```
watch <- submit party1 do  -- (3) Party1 transfers the watch to Party2
    exercise sellItem Sell with
        newOwner = party2
```

We would have this code (the one which fails):

```
watch <- submit party3 do  -- (3) Party1 transfers the wartch to Party2
    exercise sellItem Sell with
        newOwner = party2
```

The reason this contract fails is because party 3 is an observer, and hence it cannot change anything, it can only observe.