pragma solidity ^ 0.5 .1;

import "./Auction.sol";

import "./Strategy.sol";

contract DutchAuction is Auction {

Strategy strategy;

uint creationBlock;

uint winnerBlock;

uint reservePrice;

uint initialPrice;

uint actualPrice;

enum State {

GracePeriod,

Active,

Validating,

Finished

}

State state;

constructor(

string memory \_itemName,

uint \_reservePrice,

uint \_initialPrice,

Strategy \_strategy

) public {

description.seller = msg.sender;

description.itemName = \_itemName;

state = State.GracePeriod;

reservePrice = \_reservePrice;

initialPrice = \_initialPrice;

actualPrice = \_initialPrice;

strategy = \_strategy;

creationBlock = block.number;

}

function activateAuction() public onlySeller {

require(state == State.GracePeriod, "To activate the contract you must be in the Grace Period");

require(block.number - creationBlock > 20, "Grace period is not finished yet");

state = State.Active;

description.startBlock = block.number;

emit auctionStarted();

}

function getActualPrice() public returns(uint) {

uint deltaBlocks = description.startBlock - block.number;

uint tmp = strategy.getPrice(actualPrice, -deltaBlocks);

if (tmp <= reservePrice) {

actualPrice = reservePrice;

} else {

actualPrice = tmp;

}

return actualPrice;

}

function bid() public payable {

require(state == State.Active, "This contract is not active yet");

require(msg.value >= getActualPrice(), "The value sent is not sufficient");

description.winnerAddress = msg.sender;

description.winnerBid = msg.value;

winnerBlock = block.number;

validateAuction();

}

function validateAuction() internal {

require(state == State.Active, "You can't validate a contract before activating it");

state = State.Validating;

}

function finalize() public onlySeller {

require(state == State.Validating, "You can't finalize a contract before validation");

require(block.number - winnerBlock > 12, "For security reasons, you need to wait to validate the contract");

state = State.Finished;

emit auctionFinished(description.winnerAddress, description.winnerBid, address(this).balance);

description.seller.transfer(description.winnerBid);

}

}