·auction.sol

```
// SPDX-License-Identifier: ATU-1.0
// Verifier: king; wechat group: pkutoken

pragma solidity ^0.8.13;
contract Auction {
    // 拍卖的参数。
    address payable public beneficiary;
    // 时间是unix的绝对时间戳（自1970-01-01以来的秒数）
    uint public auctionEnd;
//// SPDX-License-Identifier: ATU-1.0
// Verifier: king; wechat group: pkutoken

pragma solidity ^0.8.13;
contract Auction {
    // 拍卖的参数。
    address payable public beneficiary;
    // 时间是unix的绝对时间戳（自1970-01-01以来的秒数）
    uint public auctionEnd;
 拍卖的当前状态
address  public highestBidder;
uint public highestBid;

//可以取回的之前的出价
mapping(address => uint) pendingReturns;

// 拍卖结束后设为 true，将禁止所有的变更
bool ended;

// 变更触发的事件
event HighestBidIncreased(address bidder, uint amount);
event AuctionEnded(address winner, uint amount);

// 创建一个简单的拍卖，拍卖时间为 `_biddingTime` 秒。
constructor(uint256 _biddingTime,address payable _beneficiary
)  {
    beneficiary = _beneficiary;
    auctionEnd = block.timestamp + _biddingTime;
}

/// 对拍卖进行出价，具体的出价随交易一起发送。
/// 如果没有在拍卖中胜出，则返还出价。
function bid() public payable {
    // 对于能接收以太币的函数，关键字 payable 是必须的。

    // 如果拍卖已结束，撤销函数的调用。
    require(
        block.timestamp <= auctionEnd,
        "Auction already ended."
    );
```

```
    // 如果出价不够高，返还你的钱
    require(
        msg.value > highestBid,
        "There already is a higher bid."
    );

    if (highestBid != 0) {
        pendingReturns[highestBidder] += highestBid;
    }
    highestBidder = msg.sender;
    highestBid = msg.value;
    emit HighestBidIncreased(msg.sender, msg.value);
}

/// 取回出价（当该出价已被超越）
function withdraw() public returns (bool) {
    uint amount = pendingReturns[msg.sender];

    address payable _payableAddr = payable(msg.sender);
    if (amount > 0) {
        pendingReturns[_payableAddr] = 0;
        if (!_payableAddr.send(amount)) {
            pendingReturns[msg.sender] = amount;
            return false;
        }
    }
    return true;
}

/// 结束拍卖，并把最高的出价发送给受益人
function AuctionEnd() public {
    // 1. 条件
    require(block.timestamp >= auctionEnd, "Auction not yet ended.");
    require(!ended, "auctionEnd has already been called.");

    // 2. 生效
    ended = true;
    emit AuctionEnded(highestBidder, highestBid);

    // 3. 交互
    beneficiary.transfer(highestBid);
}
}
```

·2_deploy_contracts.js

```
var Auction = artifacts.require('./Auction.sol');

module.exports = function(deployer) {
  deployer.deploy(Auction,600,"0x5C3d0BB74671C9D60EE25783a5720Ab6929a67b1");
}
```

·truffle-config.js

/**

- Use this file to configure your truffle project. It's seeded with some
- common settings for different networks and features like migrations,
- compilation, and testing. Uncomment the ones you need or modify
- them to suit your project as necessary.
  *
- More information about configuration can be found at:
  *
- https://trufflesuite.com/docs/truffle/reference/configuration
  *
- Hands-off deployment with Infura

- *
- Do you have a complex application that requires lots of transactions to deploy?
- Use this approach to make deployment a breeze 🏋️‍♀️:
  *
- Infura deployment needs a wallet provider (like @truffle/hdwallet-provider)
- to sign transactions before they're sent to a remote public node.
- Infura accounts are available for free at 🔍: https://infura.io/register
  *
- You'll need a mnemonic - the twelve word phrase the wallet uses to generate
- public/private key pairs. You can store your secrets 🤐 in a .env file.
- In your project root, run `$ npm install dotenv`.
- Create .env (which should be .gitignored) and declare your MNEMONIC
- and Infura PROJECT_ID variables inside.
- For example, your .env file will have the following structure:
  *
- MNEMONIC = <Your 12 phrase mnemonic>
- PROJECT_ID =
  *
- Deployment with Truffle Dashboard (Recommended for best security practice)

- *
- Are you concerned about security and minimizing rekt status 😱?
- Use this method for best security:
  *
- Truffle Dashboard lets you review transactions in detail, and leverages
- MetaMask for signing, so there's no need to copy-paste your mnemonic.
- More details can be found at 🔎:
  *

- https://trufflesuite.com/docs/truffle/getting-started/using-the-truffle-dashboard/
  */

// require('dotenv').config();
// const { MNEMONIC, PROJECT_ID } = process.env;

// const HDWalletProvider = require('@truffle/hdwallet-provider');

module.exports = {
 /**

- Networks define how you connect to your ethereum client and let you set the
- defaults web3 uses to send transactions. If you don't specify one truffle
- will spin up a managed Ganache instance for you on port 9545 when you
- run `develop` or `test`. You can ask a truffle command to use a specific
- network from the command line, e.g
  *
- $ truffle test --network
  */

 networks: {
  // Useful for testing. The `development` name is special - truffle uses it by default
  // if it's defined here and no other network is specified at the command line.
  // You should run a client (like ganache, geth, or parity) in a separate terminal
  // tab if you use this network and you must also set the `host`, `port` and `network_id`
  // options below to some value.
  //
  development: {
   host: "127.0.0.1",   // Localhost (default: none)
   port: 8545,   // Standard Ethereum port (default: none)
   network_id: "*",   // Any network (default: none)
  },
  //
  // An additional network, but with some advanced options...
  // advanced: {
  //   port: 8777,   // Custom port
  //   network_id: 1342,   // Custom network
  //   gas: 8500000,   // Gas sent with each transaction (default: ~6700000)
  //   gasPrice: 20000000000, // 20 gwei (in wei) (default: 100 gwei)
  //   from:

, // Account to send transactions from (default: accounts[0]) // websocket: true // Enable EventEmitter interface for web3 (default: false) // }, // // Useful for deploying to a public network. // Note: It's important to wrap the provider as a function to ensure truffle uses a new provider every time. // goerli: { // provider: () => new HDWalletProvider(MNEMONIC, `https://goerli.infura.io/v3/${PROJECT_ID}`), // network_id: 5, // Goerli's id // confirmations: 2, // # of confirmations to wait between deployments. (default: 0) // timeoutBlocks: 200, // # of blocks before a deployment times out (minimum/default: 50) // skipDryRun: true // Skip dry run before migrations? (default: false for public nets) // }, // // Useful for private networks // private: { // provider: () => new HDWalletProvider(MNEMONIC, `https://network.io`), // network_id: 2111, // This network is yours, in the cloud. // production: true // Treats this network as if it was a public net. (default: false) // } },

```
  // Set default mocha options here, use special reporters, etc.
  mocha: {
    // timeout: 100000
  },

  // Configure your compilers
  compilers: {
    solc: {
      version: "0.8.17" // Fetch exact version from solc-bin (default: truffle's version)
      // docker: true,        // Use "0.5.1" you've installed locally with docker (default: false)
      // settings: {          // See the solidity docs for advice about optimization and evmVersion
      //  optimizer: {
      //    enabled: false,
      //    runs: 200
      //  },
      //  evmVersion: "byzantium"
      // }
    }
  }

  // Truffle DB is currently disabled by default; to enable it, change enabled:
  // false to enabled: true. The default storage location can also be
  // overridden by specifying the adapter settings, as shown in the commented code below.
  //
  // NOTE: It is not possible to migrate your contracts to truffle DB and you should
  // make a backup of your artifacts to a safe location before enabling this feature.
  //
  // After you backed up your artifacts you can utilize db by running migrate as follows:
  // $ truffle migrate --reset --compile-all
  //
  // db: {
  //   enabled: false,
  //   host: "127.0.0.1",
  //   adapter: {
  //     name: "indexeddb",
  //     settings: {
  //       directory: ".db"
  //     }
  //   }
  // }
};
```