Rental Application

The deployment of a decentralized rental application utilizing Ethereum blockchain technology is the main focus of this report. Truffle Suite is utilized in developing the environment, testing and deploying smart contracts. Ganache is used to simulate ethereum. Depending on the accessibility, admins, owners (landlords), and tenants can carry out specific tasks in this application.

The main objectives include decentralization, transparency, efficiency and flexibility. Decentralization helps in eliminating intermediaries, improve transparency by recording all transactions and reduce risk associated with manipulation. Tasks that might slow down transactions and minimize human error are automated by this program. Both owners and tenants have flexibility over a number of functions.

Rental Application has 4 smart contracts namely,

- DataModel.sol has all declarations of structures and variables.
- GetSet.sol has functions that set and get values.
- Rental.sol has functions that are accessible by either admin, owner or tenant.
- Migration.sol ensures that each migration is only performed once by keeping track of the contracts that have been put on the blockchain.

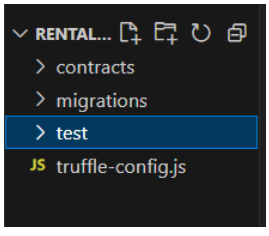
Prior steps

Create

Create a folder called rentalApp. In the directory, initiate truffle

```
PS D:\Blockchain\rentalApp> truffle init

Starting init...
================

> Copying project files to D:\Blockchain\rentalApp

Init successful, sweet!

Try our scaffold commands to get started:
  $ truffle create contract YourContractName # scaffold a contract
  $ truffle create test YourTestName         # scaffold a test

http://trufflesuite.com/docs
```

These files are created upon successful initiation



Migrate

Migrate project. (--reset is used to force re-migration)

```
PS D:\Blockchain\rentalApp> truffle migrate --reset
This version of µWS is not compatible with your Node.js build:

Error: Cannot find module '../binaries/uws_win32_x64_120.node'
Require stack:
- C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\node_modules\ganache\node_modules\@trufflesuite\uws-js-unofficial\src\uws.js
- C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\node_modules\ganache\dist\node\core.js
- C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\build\migrate.bundled.js
- C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\node_modules\original-require\index.js
- C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\build\cli.bundled.js
Falling back to a NodeJS implementation; performance may be degraded.



Compiling your contracts...
===========================
> Compiling .\contracts\DataModel.sol
> Compiling .\contracts\GetSet.sol
> Compiling .\contracts\GetSet.sol
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\Rental.sol
> Artifacts written to D:\Blockchain\rentalApp\build\contracts
> Compiled successfully using:
   - solc: 0.5.1+commit.c8a2cb62.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)
```

1st Migration file

```
const Migrations = artifacts.require("Migrations");

module.exports = function(deployer) {
  deployer.deploy(Migrations);
};
```

The file named 1_intial_migration.js deployed Migration.sol smart contract

```
1_initial_migration.js
======================

   Replacing 'Migrations'
   ----------------------
   > transaction hash:    0x06a725ed5557af213bd5f9c247b7c504dd8e3eab0cc514ac4981eea0094b883c
   > Blocks: 0            Seconds: 0
   > contract address:    0xaBd9704146352963B3442FB55361e14D73bCB259b
   > block number:        106
   > block timestamp:     1714367708
   > account:             0x94D6194CA6426fE3C739c7C5513B5947BE8f1dB3
   > balance:             99.866489452577511618
   > gas used:            244640 (0x3bba0)
   > gas price:           2.500005594 gwei
   > value sent:          0 ETH
   > total cost:          0.00061160136851616 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:     0.00061160136851616 ETH
```

2nd Migration file

```
const rental = artifacts.require("Rental");

module.exports = function(deployer) {
    deployer.deploy(rental);
};
```

The file named 2_contracts_migration.js deployes Rental.sol smart contract

```
2_contracts_migration.js
========================

   Replacing 'Rental'
   ------------------
   > transaction hash:    0xf5064d5ec0e7dc9b8eec7e616daa2bc5da2c8c5e1ca6d3e8a41833532defc68f
   > Blocks: 0            Seconds: 0
   > contract address:    0xEfd2bBa8b90c82Cded7C16FBc385997Ea86dAAd3
   > block number:        108
   > block timestamp:     1714367708
   > account:             0x94D6194CA6426fE3C739c7C5513B5947BE8f1dB3
   > balance:             99.856711400586703788
   > gas used:            3865446 (0x3afb66)
   > gas price:           2.500004337 gwei
   > value sent:          0 ETH
   > total cost:          0.009663631764439302 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:     0.009663631764439302 ETH
```

Summary

```
Summary
=======
> Total deployments:   2
> Final cost:          0.010275233132955462 ETH
```

Console

Truffle console command helps in interacting with smart contracts. This will enable a development environment.

```
PS D:\Blockchain\rentalApp> truffle console
This version of µWS is not compatible with your Node.js build:

Error: Cannot find module '../binaries/uws_win32_x64_120.node'
Require stack:
- C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\node_modules\ganache\node_modules\@trufflesuite\uws-js-unofficial\src\uws.js
- C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\node_modules\ganache\dist\node\core.js
- C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\build\console.bundled.js
- C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\node_modules\original-require\index.js
- C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\build\cli.bundled.js
Falling back to a NodeJS implementation; performance may be degraded.


truffle(development)>
```

Instance

The below command obtains an instance of deployed contract named rental.
The await keyword is used to wait asynchronously for deployed()'s promise to resolve.

```
truffle(development)> let instance = await Rental.deployed()
undefined
```

DataModel.sol

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

contract DataModel {
    uint32 public tenant_id = 1;
    uint32 public owner_id = 0;
    uint32 public house_id = 0;
    address admin;

    constructor () public {
        admin = msg.sender;
    }
```

Variables tenant_id, Owner_id, house_id are type uint32 that correspond to ids of tenant, owner and house respectively and are initialized with 1, 0, 0 respectively. Tenant_id = 0 indicates that the tenant does not exist. Variable admin is of type address. Current address is given to the admin in the constructor.

```
struct house {
    string houseNumber;
    string houseAddress;
    address ownerAddress;
    uint32 tenantId;
    uint32 rent;
    uint32 tenancy; //in months
    uint32 startDate;
    uint32 nextPaymentDate;
}
mapping(uint32 => house) public houses;
```

Structure house is declared and has 8 variables in it.

- houseNumber is type string, indicates either unit number or house number
- houseAddress is type string, indicates address of house
- ownerAddress is type address, indicates the ethereum address of house owner (from ganache)
- tenantId is type uint32, indicates id of tenant residing in the house
- rent is type uint32, indicates rent of the house
- Tenancy is type uint32, indicates tenancy in number of months
- startDate is type uint32, indicates start date of tenancy
- nextPaymentDate is type uint32, indicating next month payment date i.e., tenant will have to pay on or before this date

houses will map an integer to a house

```
struct tenant {
    string tenantName;
    string bcId;
    string phoneNumber;
    address tenantAddress;
}
mapping(uint32 => tenant) public tenants;
```

Structure tenant is declared with 4 variables.

- tenantName is type string, indicates name of tenant
- bcId is type string, indicating BCID of tenant (considering tenant in BC, Canada). To cover wider range it can be changed to govtId)
- phoneNumber is type string, indicates phone number of tenant
- tenantAddress is type address, indicates ethereum address of tenant

Tenants will map an integer to a tenant

```
struct owner {
    string ownerName;
    string phoneNumber;
    address ownerAddress;
}
mapping(uint32 => owner) public owners;
```

Structure owner is declared with 3 variables.

- ownerName is type string, indicates name of owner (landlord)
- phoneNumber is type string, indicates phone number of owner
- tenantAddress is type address, indicates ethereum address of owner

Owners will map an integer to owner.

```
mapping(uint32 => uint32[]) public trackHouseTenants;
mapping(uint32 => uint32[]) public trackHouseOwners;

mapping(uint32 => uint32[]) public trackOwnedHouses;
```

trackHouseTenants will map an integer to an array of integers indicating tenant ids. It is the list of tenants who rented a particular house.
trackHouseTenants will map an integer to an array of integers (similar to trackHouseTenants) indicating owner ids. It is the list of owners who bought a particular house.
trackOwnedHouses will map an integer to an array of integers indicating house Ids. It is the list of houses that the owner bought.

GetSet.sol

Events

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

import "./DataModel.sol";

contract GetSet is DataModel {

    event transferOwner(uint32 houseId);
    event tranferTenant(uint32 houseId);
    event paymentReceived(uint32 houseId, uint32 rent);
```

This file has 3 events

- transferOwner - takes houseId of type uint32 as input. This event is emitted when an owner sells a house
- transferTenant - takes houseId of type uint32 as input. When a tenancy for a residence is transferred, this event is released.
- paymentReceived - takes houseId and rent both of type uint32 as input. This event is emitted when rent is paid.

Functions

```
function addOwner(string memory _oName,
        string memory _phNum,
        address _oAddr) public returns (uint32){
    uint32 Id = owner_id++;
    owners[Id].ownerName= _oName;
    owners[Id].phoneNumber = _phNum;
    owners[Id].ownerAddress = _oAddr;
    return Id;
}
function getOwner(uint32 _OwnerId) public view returns (string memory, string memory, address) {
    return (owners[_OwnerId].ownerName,
            owners[_OwnerId].phoneNumber,
            owners[_OwnerId].ownerAddress);
}
```

- addOwner
    - Input
        - _oName: name of the owner (landlord) of type string
        - _phNum: phone number of owner of type string
        - _oAddr: ethereum address of owner (from ganache)
    - The "addOwner" function registers new property owners by validating and storing their details securely, often to ensure authorization.
    - Output: Id that is assigned to owner
    - Example:

```
truffle(development)> instance.addOwner("Steven", "7894561239", "0x0255c9291D65059DE3Bd80741680ADEf73595AD8")
{
  tx: '0xdf2f2f18db4ae837b1592b39d0e1731c007e90bb93b832a22515d22dbb2a3fd4',
  receipt: {
    transactionHash: '0xdf2f2f18db4ae837b1592b39d0e1731c007e90bb93b832a22515d22dbb2a3fd4',
    transactionIndex: 0,
    blockNumber: 110,
    blockHash: '0x6f4937f036ed3800026fd294b0e00d263fbe9ea028b657b2c219f8ed2616cf61',
    from: '0x94d6194ca6426fe3c739c7c5513b5947be8f1db3',
    to: '0xefd2bba8b90c82cded7c16fbc385997ea86daad3',
    cumulativeGasUsed: 96179,
    gasUsed: 96179,
    contractAddress: null,
    logs: [],
    logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000'
,
    status: true,
    effectiveGasPrice: 2500003871,
    type: '0x2',
    rawLogs: []
  },
  logs: []
}
```

- getOwner
  - Input: _ownerId - Id of owner that we want to view details
  - This function is used to get details of the owner.
  - Output: variables of owner structure of a particular owner
  - Example:

```
truffle(development)> instance.getOwner(0)
Result {
  '0': 'Steven',
  '1': '7894561239',
  '2': '0x0255c9291D65059DE3Bd80741680ADEf73595AD8'
}
```

```
function addTenant(
        string memory _tName,
        string memory _bcId,
        string memory _tphNum,
        address _tAddr) public returns (uint32){
    uint32 Id = tenant_id++;
    tenants[Id].tenantName= _tName;
    tenants[Id].bcId = _bcId;
    tenants[Id].phoneNumber = _tphNum;
    tenants[Id].tenantAddress = _tAddr;
    return Id;
}
function getTenant(uint32 _tenantId) public view returns (string memory,string memory, string memory, address) {
    return (tenants[_tenantId].tenantName,
            tenants[_tenantId].bcId,
            tenants[_tenantId].phoneNumber,
            tenants[_tenantId].tenantAddress);
}
```

- addTenant
  - Input
    - _tName: name of the tenant of type string
    - _bcId: Id of type string
    - _phNum: phone number of tenant of type string
    - _tAddr: ethereum address of tenant (from ganache)

- ○ Adds a new tenant to the application, storing their details in the "tenants" array. This function facilitates tenant onboarding, enabling engagement in rental agreements and interaction with property owners.
- ○ Output: Id that is assigned to tenant
- ○ Example:

```
truffle(development)> instance.addTenant("David", "123456789", "7784591263", "0xc9488A5784Eb8473027C2B1817c5B4C7c4d54ea0")
{
  tx: '0x304a6d4c3b3542382930d4cacffa13ae80dc762649d3980bb3e670f2bef9c1fa',
  receipt: {
    transactionHash: '0x304a6d4c3b3542382930d4cacffa13ae80dc762649d3980bb3e670f2bef9c1fa',
    transactionIndex: 0,
    blockNumber: 111,
    blockHash: '0xfd12b069fd9a4bcd4e34d153a1ecd84ec80e18a2fdd4c0b4da230beef262fbf8',
    from: '0x94d6194ca6426fe3c739c7c5513b5947be8f1db3',
    to: '0xefd2bba8b90c82cded7c16fbc385997ea86daad3',
    cumulativeGasUsed: 119810,
    gasUsed: 119810,
    contractAddress: null,
    logs: [],
    logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000'
,
    status: true,
    effectiveGasPrice: 2500003401,
    type: '0x2',
    rawLogs: []
  },
  logs: []
}
```

- getTenant
  - ○ Input: _tenantId - Id of tenant that we want to view details
  - ○ This function is used to get details of the tenant.
  - ○ Output: variables of particular tenant
  - ○ Example:
  - ○

```
truffle(development)> instance.getTenant(1)
Result {
  '0': 'David',
  '1': '123456789',
  '2': '7784591263',
  '3': '0xc9488A5784Eb8473027C2B1817c5B4C7c4d54ea0'
}
```

```
function addHouse(uint32 _ownerId,
                  uint32 _tenantId,
                    string memory _houseNumber,
                    string memory _houseAddress,
                    uint32 _rent,
                    uint32 _tenancy) public returns (uint32) {
        uint32 Id = house_id++;
        houses[Id].houseNumber = _houseNumber;
        houses[Id].houseAddress = _houseAddress;
        houses[Id].ownerAddress = owners[_ownerId].ownerAddress;
        houses[Id].tenantId = _tenantId;
        houses[Id].rent = _rent;
        houses[Id].tenancy = _tenancy;
        houses[Id].startDate = uint32(now);
        houses[Id].nextPaymentDate = houses[Id].startDate + 30 days;

        trackOwnedHouses[_ownerId].push(Id);
        trackHouseOwners[Id].push(_ownerId);
        trackHouseTenants[Id].push(_tenantId);
        return Id;
    }
```

- addHouse
    - Input
        - _ownerId: Id of the owner who owns the house, type string
        - _tenantId: Id of the tenant who resides in the residence, type string
        - _houseNumber: unit number or house number, type string
        - _houseAddress: address of house, type string
        - _rent: rent amount if there’s tenant, type uint32
        - _tenancy: tenancy in number months, type uint32
    - Adds a new house to the application, storing its details in the "houses" array. This function facilitates the listing of properties for rent, associating them with their respective owners and tenants (if applicable).
    - Output: Id that is assigned to house
    - Example: tenant Id = 0, indicating there’s no tenant

```
truffle(development)> instance.addHouse(0,0, "5846", "Sherbrooke St, Vancouver", 0, 0)
{
  tx: '0xca793a76092e4a6d1772ea37b66805aab90139316f991406b3ada8c4d8852671',
  receipt: {
    transactionHash: '0xca793a76092e4a6d1772ea37b66805aab90139316f991406b3ada8c4d8852671',
    transactionIndex: 0,
    blockNumber: 112,
    blockHash: '0x94a0f44317adac46715ec72d5aca0a189f128b3789bb747ff86a3885eb84126f',
    from: '0x94d6194ca6426fe3c739c7c5513b5947be8f1db3',
    to: '0xefd2bba8b90c82cded7c16fbc385997ea86daad3',
    cumulativeGasUsed: 147833,
    gasUsed: 147833,
    contractAddress: null,
    logs: [],
    logsBloom: '0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000'
,
    status: true,
    effectiveGasPrice: 2500002992,
    type: '0x2',
    rawLogs: []
  },
  logs: []
}
```

```
function getHouse(uint32 _houseId) public view returns (string memory,string memory,address,uint32,uint32, uint32, uint32, uint32){
    house memory h = houses[_houseId];
    return (h.houseNumber,
            h.houseAddress,
            h.ownerAddress,
            h.tenantId,
            h.rent,
            h.tenancy,
            h.startDate,
            h.nextPaymentDate);
}
```

- getHouse
    - Input: _houseId - Id of house that we want to view details
    - This function is used to get details of the house.
    - Output: variables of specific house
    - Example:

```
truffle(development)> instance.getHouse(0)
Result {
  '0': '5846',
  '1': 'Sherbrooke St, Vancouver',
  '2': '0x0255c9291D65059DE3Bd80741680ADEf73595AD8',
  '3': BN {
    negative: 0,
    words: [ 0, <1 empty item> ],
    length: 1,
    red: null
  },
  '4': BN {
    negative: 0,
    words: [ 0, <1 empty item> ],
    length: 1,
    red: null
  },
  '5': BN {
    negative: 0,
    words: [ 0, <1 empty item> ],
    length: 1,
    red: null
  },
  '6': BN {
    negative: 0,
    words: [ 36646442, 25, <1 empty item> ],
    length: 2,
    red: null
  },
  '7': BN {
    negative: 0,
    words: [ 39238442, 25, <1 empty item> ],
    length: 2,
    red: null
  }
}
```

Rental.sol

Modifiers

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

import "./DataModel.sol";
import "./GetSet.sol";

contract Rental is DataModel, GetSet{

    modifier onlyAdmin() {
        require(msg.sender == admin, "You are not the admin");
        _;
    }
    modifier onlyOwner(uint32 _houseId) {
        require(msg.sender == houses[_houseId].ownerAddress, "You are not the owner");
        _;
    }
    modifier onlyTenant(uint32 _houseId, uint32 _tenantId) {
        require(_tenantId == houses[_houseId].tenantId, "You are not the tenant");
        _;
    }
    modifier isRented(uint32 _houseId) {
        require(houses[_houseId].tenantId != 0, "No tenant");
        _;
    }
```

- onlyAdmin: only admin will be able to access
- onlyOwner: only current owner (landlord) will be able to access
- onlyTenant: only current tenant will be able to access
- isRented: checks if a house is rented or not

Functions

viewTenantDetails

```
function viewTenantDetails(uint32 _houseId) isRented(_houseId) onlyOwner(_houseId) public view returns(uint32, string memory, st
    uint32 Id = houses[_houseId].tenantId;
    return (Id,
    tenants[Id].tenantName,
    tenants[Id].bcId,
    tenants[Id].phoneNumber,
    tenants[Id].tenantAddress);
}
```

- Input: _houseId- Id of house, type integer
- Checks whether the house is rented or not, accessible only to the owner. Retrieves tenant details based on the house ID. This function enables owners to quickly view tenant details associated with a specific house.
- Output: details of tenants residing in the house
- Example:

```
truffle(development)> instance.viewTenantDetails(0)
Uncaught Error: VM Exception while processing transaction: revert No tenant
    at evalmachine.<anonymous>
    at sigintHandlersWrap (node:vm:279:12)
    at Script.runInContext (node:vm:146:14)
    at runScript (C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\build\webpack:\packages\core\lib\console.js:505:1)
    at Console.interpret (C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\build\webpack:\packages\core\lib\console.js:520:1)
    at bound (node:domain:432:15)
    at REPLServer.runBound [as eval] (node:domain:443:12)
    at REPLServer.onLine (node:repl:927:10)
    at REPLServer.emit (node:events:519:28)
    at REPLServer.emit (node:domain:488:12)
    at REPLServer.[_onLine] [as _onLine] (node:internal/readline/interface:416:12)
    at REPLServer.[_line] [as _line] (node:internal/readline/interface:887:18)
    at REPLServer.[_ttyWrite] [as _ttyWrite] (node:internal/readline/interface:1265:22)
```

Error that there is no tenant. This proves that the isRented() modifier is working properly. House Id = 0 is a new house owned by Steven and there are no tenants yet. Later Steven rents it out to David whose tenant Id = 1 using the following function.

newTenant

```
function newTenant(uint32 _houseId, uint32 _newTenantId, uint32 _rent, uint32 _tenancy) onlyOwner(_houseId)public returns (uint32) {
    houses[_houseId].tenantId = _newTenantId;
    houses[_houseId].rent = _rent;
    houses[_houseId].tenancy = _tenancy;
    houses[_houseId].startDate = uint32(now);

    trackHouseTenants[_houseId].push(_newTenantId);
    emit tranferTenant(_houseId);
    return _newTenantId;
}
```

- Input:
    - _houseId - Id of house, type integer
    - _newTenantId - Id of new resident, type integer
    - _rent - rent of house, type integer
    - _tenancy - number of months of tenancy, type integer

- This function facilitates the transition of property to new tenants, accessible only by the owner (landlord).
- Output: Id of new tenant, type integer
- Example: There is no tenant at the moment i.e., tenant Id = 0. Now new tenant id = 1.

```
truffle(development)> instance.newTenant(0, 1, 2500, 6)
Uncaught:
Error: VM Exception while processing transaction: revert You are not the owner -- Reason given: You are not the owner.
    at evalmachine.<anonymous>
    at sigintHandlersWrap (node:vm:279:12)
    at Script.runInContext (node:vm:146:14)
    at runScript (C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\build\webpack:\packages\core\lib\console.js:505:1)
    at Console.interpret (C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\build\webpack:\packages\core\lib\console.js:520:1)
    at bound (node:domain:432:15)
    at REPLServer.runBound [as eval] (node:domain:443:12)
    at REPLServer.onLine (node:repl:927:10)
    at REPLServer.emit (node:events:519:28)
    at REPLServer.emit (node:domain:488:12)
    at REPLServer.[_onLine] [as _onLine] (node:internal/readline/interface:416:12)
    at REPLServer.[_line] [as _line] (node:internal/readline/interface:887:18)
    at REPLServer.[_ttyWrite] [as _ttyWrite] (node:internal/readline/interface:1265:22)
    at REPLServer.self._ttyWrite (node:repl:1022:9)
    at ReadStream.onkeypress (node:internal/readline/interface:264:20)
    at ReadStream.emit (node:events:519:28)
```

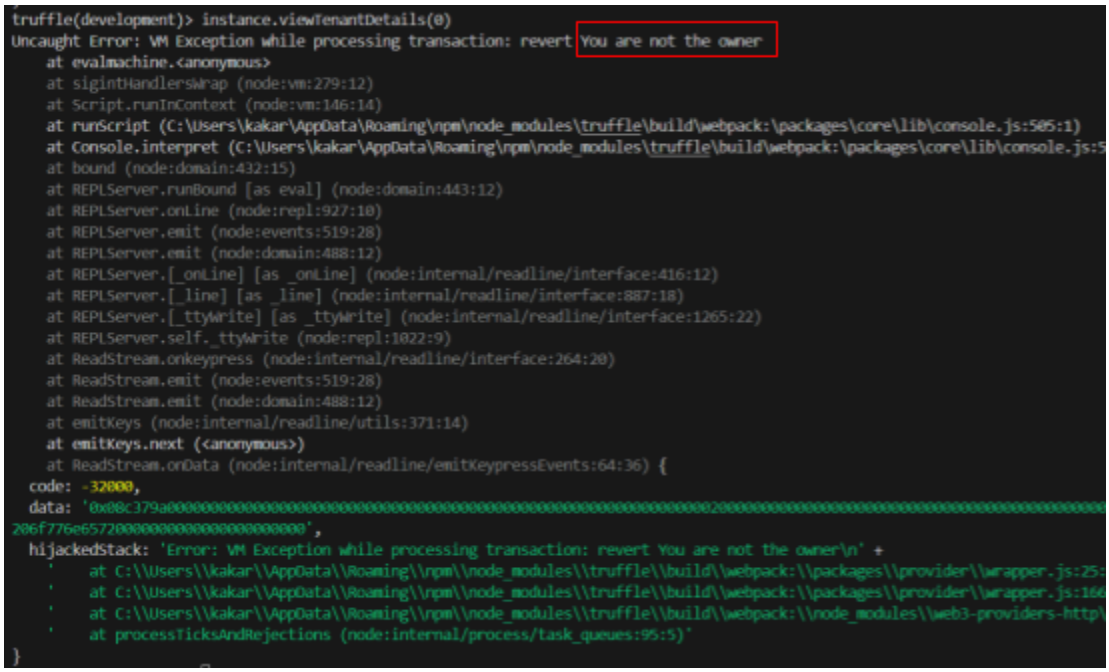
This function is accessed by the admin. Hence, there is an error that you are not the owner.

This proves that onlyOwner() modifier is working properly.

```
truffle(development)> instance.newTenant(0, 1, 2500, 6, {from: "0x0255c9291D65059DE3Bd80741680ADEf73595AD8"})
{
  tx: '0x9e2013490721cb391b89d1932b9b5b6d64d0c7f7787089eef39c7b1e6128c24d',
  receipt: {
    transactionHash: '0x9e2013490721cb391b89d1932b9b5b6d64d0c7f7787089eef39c7b1e6128c24d',
    transactionIndex: 0,
    blockNumber: 113,
    blockHash: '0x08243689c646e9ae6e2dae7242f0659969f9144eba55759663d9d5e4025faa1b',
    from: '0x0255c9291d65059de3bd80741680adef73595ad8',
    to: '0xefd2bba8b90c82cded7c16fbc385997ea86daad3',
    cumulativeGasUsed: 79248,
    gasUsed: 79248,
    contractAddress: null,
    logs: [ [Object] ],
    logsBloom: '0x00000000000000000000000010000000000000040000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000001000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
',
    status: true,
    effectiveGasPrice: 2500002635,
    type: '0x2',
    rawLogs: [ [Object] ]
  },
  logs: [
    {
      address: '0xEfd2bBa8b90c82Cded7C16FBc385997Ea86dAAd3',
      blockHash: '0x08243689c646e9ae6e2dae7242f0659969f9144eba55759663d9d5e4025faa1b',
      blockNumber: 113,
      logIndex: 0,
      removed: false,
      transactionHash: '0x9e2013490721cb391b89d1932b9b5b6d64d0c7f7787089eef39c7b1e6128c24d',
      transactionIndex: 0,
      id: 'log_ce24025c',
      event: 'tranferTenant',
      args: [Result]
    }
  ]
}
```

Tenant is transferred from id 0 to 1 by the owner. Event transferTenant() is emitted.

As house Id 0 is rented, we can view tenant details now.



Only the owner can access this function to view tenant details.

```
truffle(development)> instance.viewTenantDetails(0, {from:"0x0255c9291D65059DE3Bd80741680ADEf73595AD8"})
Result {
  '0': BN {
    negative: 0,
    words: [ 1, <1 empty item> ],
    length: 1,
    red: null
  },
  '1': 'David',
  '2': '123456789',
  '3': '7784591263',
  '4': '0xc9488A5784Eb8473027C2B1817c5B4C7c4d54ea0'
}
```

When a tenant requests owner for extension of rental, the owner will verify everything and extend the rental making necessary changes to rent (if applicable). This can be done using the following function.

modifyTenant

```
//Modify
function modifyTenant(uint32 _houseId, uint32 _rent, uint32 _tenancy) isRented(_houseId) onlyOwner(_houseId) public returns(uint32) {
    houses[_houseId].rent = _rent;
    houses[_houseId].tenancy += _tenancy;
    return houses[_houseId].tenantId;
}
```

- Input:
  - _houseId - Id of house, type integer
  - _rent - rent of house, type integer
  - _tenancy - extended number of months of tenancy, type integer
- This function enables adjustments to rental agreements, such as increasing the rent or extending the tenancy duration.
- Output: Id of tenant residing, type integer
- Example: Owner extends rental by 6 months by increasing rent to 2700.

```
truffle(development)> instance.modifyTenant(0, 2700, 6, {from:"0x0255c9291D65059DE3Bd80741680ADEf73595AD8"})
{
  tx: '0x36cf9bc2215af4328c7c587fb8716bbc83a9b02a77cb43cc3eca62ce96f95b41',
  receipt: {
    transactionHash: '0x36cf9bc2215af4328c7c587fb8716bbc83a9b02a77cb43cc3eca62ce96f95b41',
    transactionIndex: 0,
    blockNumber: 114,
    blockHash: '0x17e113704b1f9b279bea548074236f5a727cf65459b23852d071943cc28594b8',
    from: '0x0255c9291d65059de3bd80741680adef73595ad8',
    to: '0xefd2bba8b90c82cded7c16fbc385997ea86daad3',
    cumulativeGasUsed: 28895,
    gasUsed: 28895,
    contractAddress: null,
    logs: [],
    logsBloom: '0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
,
    status: true,
    effectiveGasPrice: 2500002314,
    type: '0x2',
    rawLogs: []
  },
  logs: []
}
```

Modified details are reflected in house details

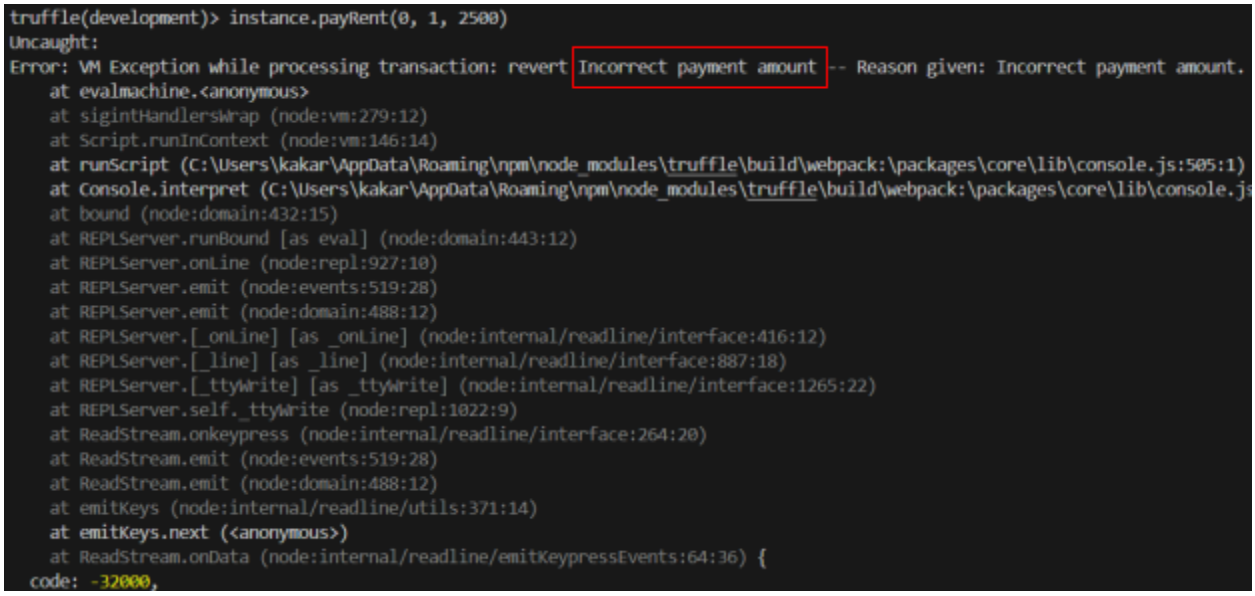
```
truffle(development)> instance.getHouse(0)
Result {
  '0': '5846',
  '1': 'Sherbrooke St, Vancouver',
  '2': '0x0255c9291D65059DE3Bd80741680ADEf73595AD8',
  '3': BN {
    negative: 0,
    words: [ 1, <1 empty item> ],
    length: 1,
    red: null
  },
  '4': BN {
    negative: 0,
    words: [ 2700, <1 empty item> ],
    length: 1,
    red: null
  },
  '5': BN {
    negative: 0,
    words: [ 12, <1 empty item> ],
    length: 1,
    red: null
  },
  '6': BN {
    negative: 0,
    words: [ 36646850, 25, <1 empty item> ],
    length: 2,
    red: null
  },
  '7': BN {
    negative: 0,
    words: [ 39238442, 25, <1 empty item> ],
    length: 2,
    red: null
  }
}
```

payRent

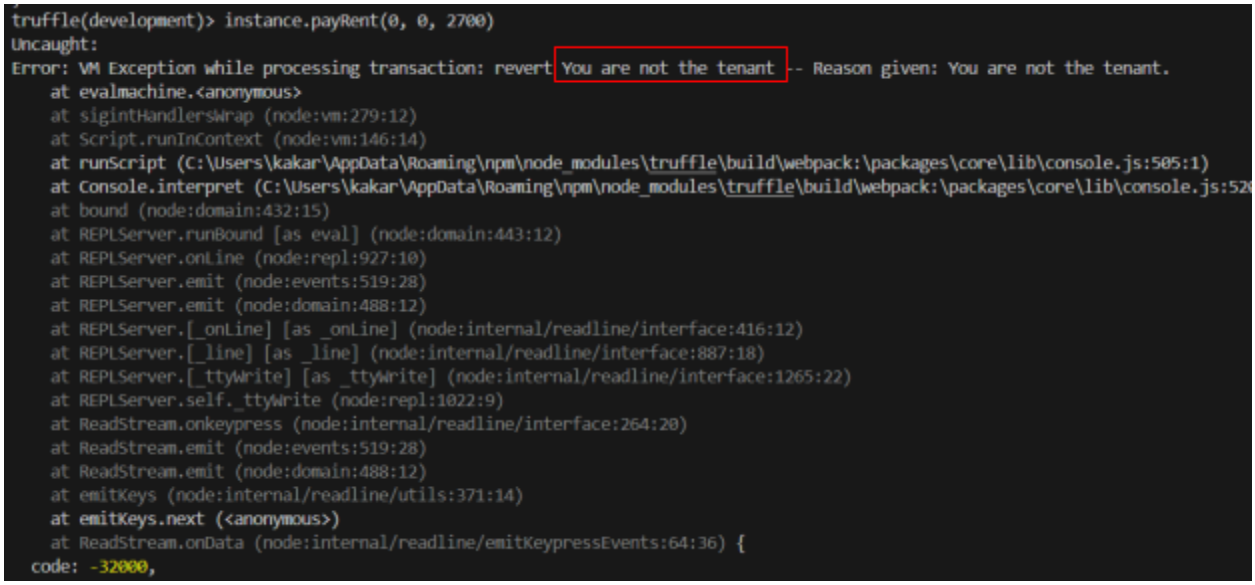
```
//PayRent
function payRent(uint32 _houseId,uint32 _tenantId, uint32 _rent) isRented(_houseId) onlyTenant(_houseId, _tenantId) public returns (bool) {
    require(_rent == houses[_houseId].rent, "Incorrect payment amount");
    require(block.timestamp <= houses[_houseId].nextPaymentDate, "Rent is already paid for the current month");
    houses[_houseId].nextPaymentDate += 30 days;
    emit paymentReceived(_houseId, _rent);
    return true;
}
```

- Input:
    - _houseId - Id of house, type integer
    - _tenantId - Id of tenant, type integer
    - _rent - current rent of house, type integer
- Allows the tenant to pay rent for the rented property. Accessible only by current tenants. Tenants must enter the correct amount for successful payment. Assumes the tenant pays before the next payment due and pays only once.
- Output: Type bool

- Example: tenant id = 1 is paying rent for house Id = 0 that he is residing in.



If, tenant entered the previous amount by mistake. In this case, an error occurred which states an incorrect payment amount.



If the wrong tenant_id is entered, the above error occurs stating you are not the tenant. This proves that onlyTenant() modifier is working properly.

```
truffle(development)> instance.payRent(0, 1, 2700)
{
  tx: '0xa1e2ceb20b94ccc70b8ed0354f335df804fc028b703f2793d136fe7e8d4b587f',
  receipt: {
    transactionHash: '0xa1e2ceb20b94ccc70b8ed0354f335df804fc028b703f2793d136fe7e8d4b587f',
    transactionIndex: 0,
    blockNumber: 161,
    blockHash: '0xe39c2e01fefa174362ca35d29c75abf5c8380f5c3970fbb63360005bbd3aef3c',
    from: '0x94d6194ca6426fe3c739c7c5513b5947be8f1db3',
    to: '0xac1e08ab427c1376b1f6c9ac9a811d781291680e',
    cumulativeGasUsed: 32262,
    gasUsed: 32262,
    contractAddress: null,
    logs: [ [Object] ],
    logsBloom: '0x00000000000000000000000010000000000000000000000000000000000000000000000000000000000000
00008000000000008000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000000000000000000000
',
    status: true,
    effectiveGasPrice: 2500000016,
    type: '0x2',
    rawLogs: [ [Object] ]
  },
  logs: [
    {
      address: '0xAc1E08AB427c1376B1f6c9Ac9A811d781291680E',
      blockHash: '0xe39c2e01fefa174362ca35d29c75abf5c8380f5c3970fbb63360005bbd3aef3c',
      blockNumber: 161,
      logIndex: 0,
      removed: false,
```

Upon successful completion, paymentReceived() event is emitted.

EVENT NAME
paymentReceived

CONTRACT	TX HASH	LOG INDEX	BLOCK TIME
Rental	0x30c9dca94eb229d7fadcedc698f505827082cb223d9d8f a1f82905b5b1b4c8f8	0	2024-04-28 22:58:46

checkRentDue

```
//Check rent due
function checkRentDue(uint32 _houseId) isRented(_houseId) onlyOwner(_houseId) public view returns(bool) {
    return block.timestamp >= houses[_houseId].nextPaymentDate;
}
```

- Input: _houseId - Id of house, type integer
- Owner can check if rent is due by tenant using this function
- Allows the owner to check if rent is due by the tenant for the specified house. This function provides a convenient way for owners to monitor rent payment status..
- Output: type bool
- Example:

```
truffle(development)> instance.checkRentDue(0, {from:"0x0255c9291D65059DE3Bd80741680ADEf73595AD8"})
false
```

houseTenants

```
function houseTenants(uint32 _houseId) onlyAdmin() external view returns (uint32[] memory) {
    return trackHouseTenants[_houseId];
}
```

- Input: _houseId - Id of house, type integer
- Allows the admin to monitor or track tenants of a particular house. This function retrieves an array of tenants associated with the specified house, enabling the admin to view the list of tenants residing in that property.
- Output: array of tenants
- Example: initially there's no tenant i.e., tenant id was 0. Then, tenant id = 1 is residing in house id = 0

```
truffle(development)> instance.houseTenants(0)
[
  BN {
    negative: 0,
    words: [ 0, <1 empty item> ],
    length: 1,
    red: null
  },
  BN {
    negative: 0,
    words: [ 1, <1 empty item> ],
    length: 1,
    red: null
  }
]
```

```
truffle(development)> instance.houseTenants(0, {from:"0x0255c9291D65B59DF38da0741680ADEf73595AD8"})
Uncaught Error: VM Exception while processing transaction: revert You are not the admin
    at evalmachine.<anonymous>
    at sigintHandlersWrap (node:vm:279:12)
    at Script.runInContext (node:vm:146:14)
    at runScript (C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\build\webpack:\packages\core\lib\console.js:5
    at Console.interpret (C:\Users\kakar\AppData\Roaming\npm\node_modules\truffle\build\webpack:\packages\core\lib\cons
    at bound (node:domain:432:15)
    at REPLServer.runBound [as eval] (node:domain:443:12)
    at REPLServer.onLine (node:repl:927:10)
    at REPLServer.emit (node:events:519:28)
    at REPLServer.emit (node:domain:488:12)
    at REPLServer.[_onLine] [as _onLine] (node:internal/readline/interface:416:12)
    at REPLServer.[_line] [as _line] (node:internal/readline/interface:887:18)
    at REPLServer.[_ttyWrite] [as _ttyWrite] (node:internal/readline/interface:1265:22)
    at REPLServer.self._ttyWrite (node:repl:1022:9)
    at ReadStream.onkeypress (node:internal/readline/interface:264:20)
    at ReadStream.emit (node:events:519:28)
    at ReadStream.emit (node:domain:488:12)
    at emitKeys (node:internal/readline/utils:371:14)
    at emitKeys.next (<anonymous>)
    at ReadStream.onData (node:internal/readline/emitKeypressEvents:64:36) {
  code: -32000,
  data: '0x08c379a0000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000000000000000000000000000000000000001a
2061646d696e0000000000000000000000000',
  hijackedStack: 'Error: VM Exception while processing transaction: revert You are not the admin\n' +
    '    at C:\\Users\\kakar\\AppData\\Roaming\\npm\\node_modules\\truffle\\build\\webpack:\\packages\\provider\\wrappe
    '    at C:\\Users\\kakar\\AppData\\Roaming\\npm\\node_modules\\truffle\\build\\webpack:\\packages\\provider\\wrappe
    '    at C:\\Users\\kakar\\AppData\\Roaming\\npm\\node_modules\\truffle\\build\\webpack:\\node_modules\\web3-provide
    '    at processTicksAndRejections (node:internal/process/task_queues:95:5)'
}
```

- If a landlord is accessing this function it will throw an error saying you are not an admin error. This proves that onlyAdmin() function is working properly.

houseOwners

```
function houseOwners(uint32 _houseId) onlyAdmin() external view returns (uint32[] memory) {
    return trackHouseOwners[_houseId];
}
```

- Input: _houseId - Id of house, type integer
- Enables the admin to monitor or track landlords of a particular house. This function retrieves an array of owners associated with the specified house, allowing the admin to view the list of landlords who own the property.
- Output: array of owners
- Example:

```
truffle(development)> instance.houseOwners(0)
[
  BN {
    negative: 0,
    words: [ 0, <1 empty item> ],
    length: 1,
    red: null
  }
]
```

OwnedHouses

```
//Tracking
function OwnedHouses(uint32 _ownerId) onlyAdmin() public view returns(uint32[] memory) {
    return trackOwnedHouses[_ownerId];
}
```

- Input: _ownerId - Id of house, type integer
- This function retrieves an array of houses owned by the specified landlord, enabling the admin to view the list of properties associated with that owner.
- Output: array of houses
- Example: owner id = 0 owns house id = 0

```
truffle(development)> instance.OwnedHouses(0)
[
  BN {
    negative: 0,
    words: [ 0, <1 empty item> ],
    length: 1,
    red: null
  }
]
```

Adding a new owner to execute newOwner() function. An owner can create another landlord.

```
truffle(development)> instance.addOwner("Fely", "7789456213", "0x977a339458fd10611dc81758eB7D51090aC80ebC", {from:"0x0255c9291D65059DE3Bd80741680ADEf73595AD8"})
{
  tx: '0x61bf8ea7947cb6b5f128cef1bab748f3f0f9d4b92a650285155874d9b0f6f011',
  receipt: {
    transactionHash: '0x61bf8ea7947cb6b5f128cef1bab748f3f0f9d4b92a650285155874d9b0f6f011',
    transactionIndex: 0,
    blockNumber: 115,
    blockHash: '0x5f057be4ed65811f66bc0fb6a2af3edfd09b67f824afa200a941fd6acf9731ec',
    from: '0x0255c9291d65059de3bd80741680adef73595ad8',
    to: '0xefd2bba8b90c82cded7c16fbc385997ea86daad3',
    cumulativeGasUsed: 96155,
    gasUsed: 96155,
    contractAddress: null,
    logs: [],
    logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
,
    status: true,
    effectiveGasPrice: 2500002028,
    type: '0x2',
    rawLogs: []
  },
  logs: []
}
```

```
truffle(development)> instance.getOwner(1)
Result {
  '0': 'Fely',
  '1': '7789456213',
  '2': '0x977a339458fd10611dc81758eB7D51090aC80ebC'
}
```

newOwner

```
//Transfer
function newOwner(uint32 _oldOwnerId, uint32 _newOwnerId, uint32 _houseId) onlyOwner(_houseId) public returns (uint32) {
    deleteHouse(_oldOwnerId, _houseId);
    owner memory o2 = owners[_newOwnerId];
    houses[_houseId].ownerAddress = o2.ownerAddress;
    houses[_houseId].tenantId = 0;
    houses[_houseId].rent = 0;
    houses[_houseId].tenancy = 0;

    trackOwnedHouses[_newOwnerId].push(_houseId);
    trackHouseOwners[_houseId].push(_newOwnerId);
    trackHouseTenants[_houseId].push(0);
    emit transferOwner(_houseId);
    return _newOwnerId;
}
```

- Input:
  - _oldOwnerId - Id of old owner, type integer
  - _newOwnerId - Id of new owner, type integer
  - _houseId - Id of house, type integer
- This function transfers house property to the new owner. It can only be accessed by the previous owner (landlord).
- Output: Id of new owner, type integer
- Anytime a landlord sells a home, it needs to be taken off his list and placed to the new landlord list so that it is kept track of which landlords own which houses. To eliminate a property from the previous owner list, deleteHouse() function is used.

```
function deleteHouse(uint32 _ownerId, uint32 _houseId) internal returns(uint32 ) {
    uint32[] memory houses = trackOwnedHouses[_ownerId];
    uint256 l = houses.length;
    uint256 index = 0;
    for (uint256 i = 0; i < l; i++) {
        if (houses[i] == _houseId) {
            index = i;
            for (uint256 j = i+1; j < l; j++) {
                trackOwnedHouses[_ownerId][j-1] = trackOwnedHouses[_ownerId][j];
            }
            trackOwnedHouses[_ownerId].pop();
            break;
        }
    }
    return 0;
}
```

Since it is internal, it can't be accessed in the development environment.

```
truffle(development)> instance.deleteHouse(0,0)
Uncaught TypeError: instance.deleteHouse is not a function
```

- Example: owner Id = 0 transfers property (house Id = 0) to owner Id = 1

```
truffle(development)> instance.newOwner(0, 1, 0, {from: "0x0255c9291D65059DE3Bd80741680ADEf73595AD8"})
{
  tx: '0x86b2e770762a612cd5b8edcf7146c0ffa356242b747393f7c52e82cd0be21f4d',
  receipt: {
    transactionHash: '0x86b2e770762a612cd5b8edcf7146c0ffa356242b747393f7c52e82cd0be21f4d',
    transactionIndex: 0,
    blockNumber: 116,
    blockHash: '0x2dfe21c15b33ea8359c0e994bfe338faa429bd69383374cedd1311139053d61f',
    from: '0x0255c9291d65059de3bd80741680adef73595ad8',
    to: '0xefd2bba8b90c82cded7c16fbc385997ea86daad3',
    cumulativeGasUsed: 117667,
    gasUsed: 117667,
    contractAddress: null,
    logs: [ [Object] ],
    logsBloom: '0x00000000000000000000000010000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000004
00000000001000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000
',
    status: true,
    effectiveGasPrice: 2500001782,
    type: '0x2',
    rawLogs: [ [Object] ]
  },
  logs: [
    {
      address: '0xEfd2bBa8b90c82Cded7C16FBc385997Ea86dAAd3',
      blockHash: '0x2dfe21c15b33ea8359c0e994bfe338faa429bd69383374cedd1311139053d61f',
      blockNumber: 116,
      logIndex: 0,
      removed: false,
      transactionHash: '0x86b2e770762a612cd5b8edcf7146c0ffa356242b747393f7c52e82cd0be21f4d',
      transactionIndex: 0,
      id: 'log_3021498c',
      event: 'transferOwner',
      args: [Result]
    }
  ]
}
```

On successful completion, transferOwner() event is emitted.

Updated values are as follows

1. Owner updated with no tenant

```
truffle(development)> instance.getHouse(0)
Result {
  '0': '5846',
  '1': 'Sherbrooke St, Vancouver',
  '2': '0x977a339458fd10611dc81758eB7D51090aC80ebC',
  '3': BN {
    negative: 0,
    words: [ 0, <1 empty item> ],
    length: 1,
    red: null
  },
  '4': BN {
    negative: 0,
    words: [ 0, <1 empty item> ],
    length: 1,
    red: null
  },
  '5': BN {
    negative: 0,
    words: [ 0, <1 empty item> ],
    length: 1,
    red: null
  },
  '6': BN {
    negative: 0,
    words: [ 36646850, 25, <1 empty item> ],
    length: 2,
    red: null
  },
  '7': BN {
    negative: 0,
    words: [ 39238442, 25, <1 empty item> ],
    length: 2,
    red: null
  }
}
```

2. Owner Id = 1 owns a house id = 0

```
truffle(development)> instance.OwnedHouses(1)
[
  BN {
    negative: 0,
    words: [ 0, <1 empty item> ],
    length: 1,
    red: null
  }
]
```

3. Owner Id = 0 doesn't own any house

```
truffle(development)> instance.OwnedHouses(0)
[]
```

4. Track of tenants of house Id = 0

```
truffle(development)> instance.houseTenants(0)
[
  BN {
    negative: 0,
    words: [ 0, <1 empty item> ],
    length: 1,
    red: null
  },
  BN {
    negative: 0,
    words: [ 1, <1 empty item> ],
    length: 1,
    red: null
  },
  BN {
    negative: 0,
    words: [ 0, <1 empty item> ],
    length: 1,
    red: null
  }
]
```

5. Track of owners of house Id = 0

```
truffle(development)> instance.houseOwners(0)
[
  BN {
    negative: 0,
    words: [ 0, <1 empty item> ],
    length: 1,
    red: null
  },
  BN {
    negative: 0,
    words: [ 1, <1 empty item> ],
    length: 1,
    red: null
  }
]
```

Test cases

Test file contains 3 tests

- Add a house
- View tenant details
- Transfer rental contract

beforeEach function is used to deploy an instance before each test case.
An owner, a tenant and a house are created in each test case.

```
var rental = artifacts.require("Rental");

contract('Rental', async accounts => {

    beforeEach(async () => {
        let instance = await rental.deployed();

        await instance.addOwner("Steven", "7894561239", "0x0255c9291D65059DE3Bd80741680ADEf73595AD8");
        await instance.addTenant("David", "123456789", "7784591263", "0xc9488A5784Eb8473027C2B1817c5B4C7c4d54ea0");
        await instance.addHouse(0, 1, "5846", "Sherbrooke St, Vancouver", 2500, 6);
    });
```

- *Add a house*

```
it("Add a house", async () => {
    let instance = await rental.deployed();
    let house = await instance.getHouse(0);
    assert.equal(house[0], "5846");
});
```

It serves as a purpose of verifying the capability of method addhouse(). It ensures smooth integration of residence into the platform and consistent storage by recording all fields. Functionality is tested by getting house details from the getter method getHouse().

- *View tenant details*

```
it("View tenant details", async () => {
    let instance = await rental.deployed();
    let tenantDetails = await instance.viewTenantDetails(0, {from: "0x0255c9291D65059DE3Bd80741680ADEf73595AD8"});
    assert.equal(tenantDetails[1], "David");
});
```

It checks that the name, address, and other relevant details about the tenant are shown correctly. This verifies the accuracy and consistency of the tenant data that was obtained from the system. It also makes sure that tenant data is blocked for unauthorized persons and is only accessible to the landlord.

- *Transfer rental contract*

```
it("transfer rental contract", async () => {
    let instance = await rental.deployed();

    let tenantId2 = await instance.addTenant("Stacy", "456123789", "7784951263", "0x3F0aD95Fc10c5DAf079fB7365038D5F5FaC2477e");
    let newTenantId = await instance.newTenant(0, 2, 2500, 6, {from: "0x0255c9291D65059DE3Bd80741680ADEf73595AD8"});

    let house = await instance.getHouse(0);
    assert.equal(house[3], 2);
});
```

This test case tests if an owner can move the rental contract from one tenant to another. It verifies that upon the transfer, the information related to the rental contract is updated appropriately i.e.The new tenant's name, contact information, and rental terms are reflected in the contract. Ascertain that the transfer of the rental agreement can only be started by the landlord.

All the test cases passed.

```
Contract: Rental
  ✓ Add a house (77ms)
  ✓ View tenant details (79ms)
  ✓ transfer rental contract (476ms)


3 passing (3s)
```