PD-13.0 Testing

✓ (Initial value should be100)

✓ (Initial value should not be200)

✗ (Should trigger one fail)

✓ (Should trigger one pass)

PD-13.1 Types of tests

- Requirements / Design review
- Incentive / token design – economics simulation
- Formal verification
- Functional Testing
- Unit tests (solidity & javascript)
- End to end tests
- Security test / audit
- Performance / load
- Compatibility browsers/wallets
- Smart contract interfaces
- Mutation testing / fuzzing
- Regression testing
- API testing
- State machine testing
- CRUD testing
- Audits

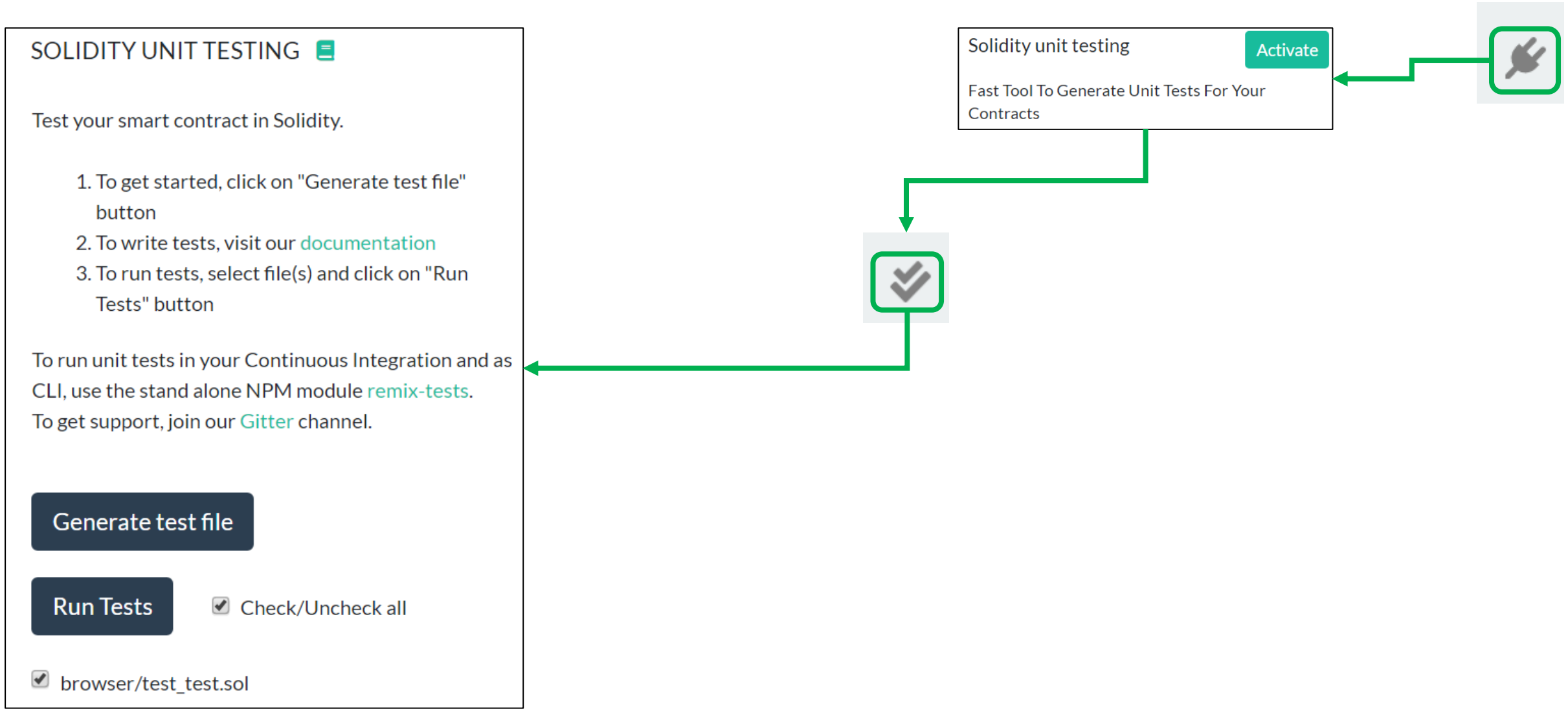
PD-13.1 10 Tips to Writing Good Unit Tests

1. Make Them Short
2. Don't Repeat Yourself
3. Prefer Composition Over Inheritance
4. Make Them Fast
5. Make Them Deterministic
6. Don't Ignore Tests
7. Test Your Tests
8. Name Your Tests Well
9. One Logical Assertion Per Test
10. Design Your Tests

https://dzone.com/articles/10-tips-to-writing-good-unit-tests

https://en.wikipedia.org/wiki/Composition_over_inheritance

PD-13.2 Remix – unit tests



https://remix-ide.readthedocs.io/en/latest/unittesting.html

PD-13.2 Remix test – example 1: SimpleStorage

Home | simple_storage.sol | simple_storage_test.sol

```
1  pragma solidity >= 0.5.0 < 0.7.0;
2  contract SimpleStorage {
3    uint public storedData;
4
5    constructor() public {
6      storedData = 100;
7    }
8
9    function set(uint x) public {
10     storedData = x;
11   }
12
13   function get() public view returns (uint retVal) {
14     return storedData;
15   }
16
17 }
```

Home | simple_storage.sol | simple_storage_test.sol

```
1  pragma solidity >= 0.5.0 < 0.7.0;
2  import "./simple_storage.sol";
3
4  contract MyTest {
5    SimpleStorage foo;
6
7    function beforeAll() public {
8      foo = new SimpleStorage();
9    }
10
11   function initialValueShouldBe100() public returns (bool) {
12     return Assert.equal(foo.get(), 100, "initial value is not correct");
13   }
14
15   function initialValueShouldNotBe200() public returns (bool) {
16     return Assert.notEqual(foo.get(), 200, "initial value is not correct");
17   }
18
19   function shouldTriggerOneFail() public {
20     Assert.equal(uint(1), uint(2), "uint test 1 fails");
21     Assert.notEqual(uint(1), uint(2), "uint test 2 passes");
22   }
23
24   function shouldTriggerOnePass() public {
25     Assert.equal(uint(1), uint(1), "uint test 3 passes");
26   }
27 }
```

Progress: 1 finished (of 1)

FAIL **MyTest**

(browser/tests/simple_storage_test.sol)

✓ Initial value should be100

✓ Initial value should not be200

✗ Should trigger one fail

Error Message:

"uint test 1 fails"

Assertion:

Expected value should be **equal** to 2

Received value:

1

Skipping the remaining tests of the function.

✓ Should trigger one pass

Result for browser/tests/simple_storage_test.sol

Passing: 3

Failing: 1

Total time: 1.11s

https://remix-ide.readthedocs.io/en/latest/unittesting.html

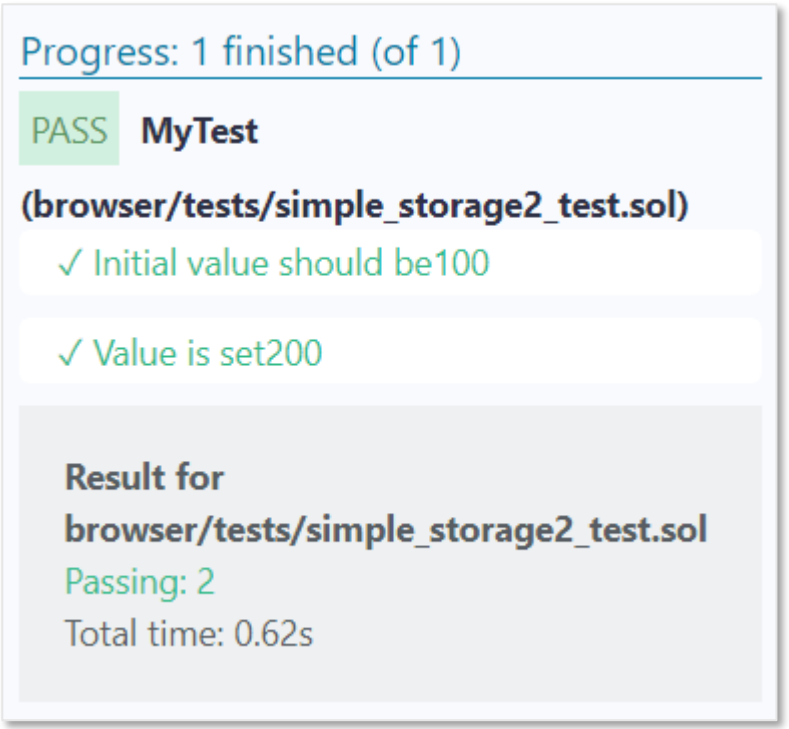
https://remix-ide.readthedocs.io/en/latest/assert_library.html

https://github.com/ethereum/remix-project/blob/master/libs/remix-tests/tests/examples_1/simple_storage_test.sol

PD-13.2 Remix test – example 2: SimpleStorage

simple_storage_test.sol

```
1  pragma solidity >= 0.5.0 < 0.7.0;
2  import "./simple_storage.sol";
3
4  contract MyTest {
5    SimpleStorage foo;
6    uint i = 0;
7
8    function beforeEach() public {
9      foo = new SimpleStorage();
10     if (i == 1) {
11       foo.set(200);
12     }
13     i += 1;
14   }
15
16   function initialValueShouldBe100() public returns (bool) {
17     return Assert.equal(foo.get(), 100,
18         "initial value is not correct");
19   }
20
21   function valueIsSet200() public returns (bool) {
22     return Assert.equal(foo.get(), 200,
23         "value is not correct after first execution");
24   }
25 }
```



https://github.com/ethereum/remix-project/blob/master/libs/remix-tests/tests/examples_2/simple_storage_test.sol

PD-13.2 Remix test: Assert Library

```
Assert.equal(string("a"), "a");
// OK
Assert.equal(uint(100), 100);
// OK
foo.set(200)
Assert.equal(foo.get(), 200);
// OK
Assert.equal(foo.get(), 100, "value should be 200");
// error: value should be 200
```

```
Assert.notEqual(string("a"), "b");
// OK
foo.set(200)
Assert.notEqual(foo.get(), 200, "value should not be 200");
// error: value should not be 200
```

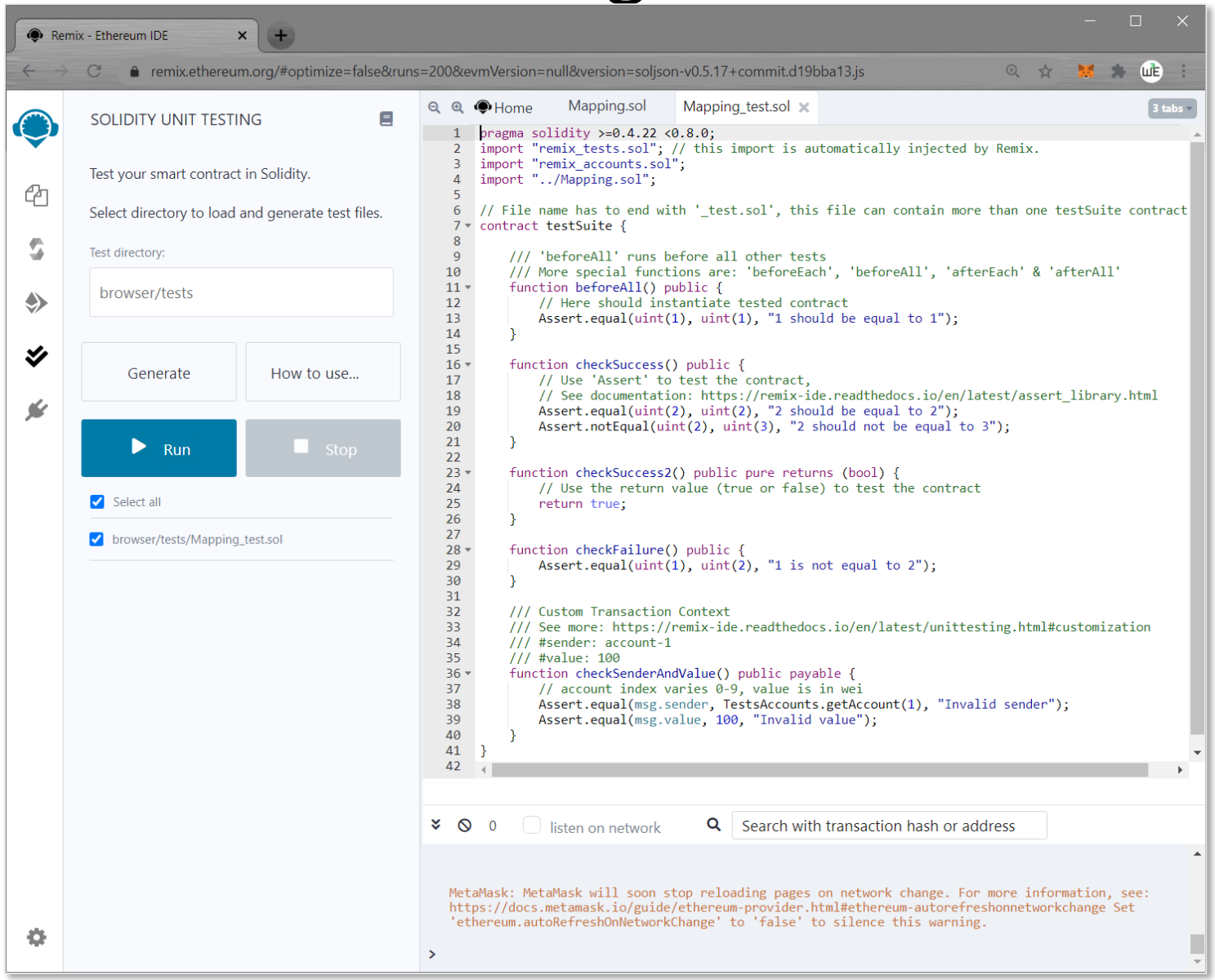
```
Assert.lesserThan(int(-2), int(-1));
// OK
Assert.lesserThan(int(2), int(1), "2 is not lesser than 1");
// error: 2 is not greater than 1
```

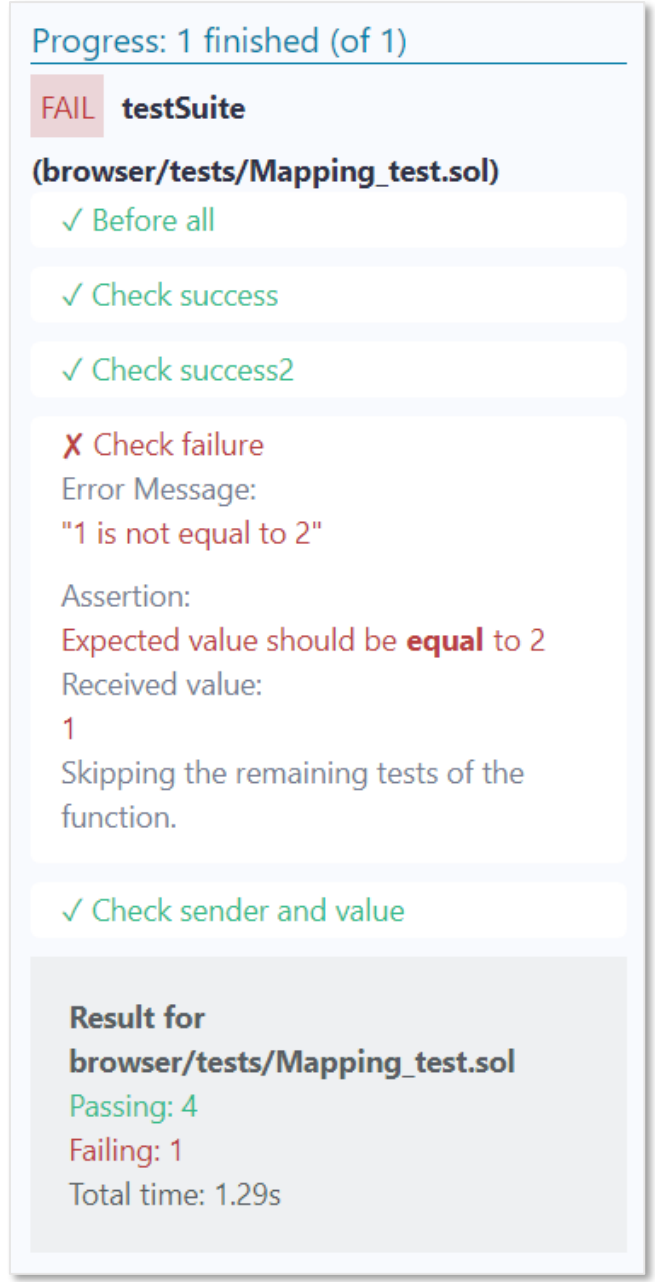
```
Assert.greaterThan(uint(2), uint(1));
// OK
Assert.greaterThan(uint(-2), uint(1));
// OK
Assert.greaterThan(int(2), int(1));
// OK
Assert.greaterThan(int(-2), int(-1), "-2 is not greater than -1");
// error: -2 is not greater than -1
```

```
Assert.ok(true);
// OK
Assert.ok(false, "it\'s false");
// error: it's false
```

https://remix-ide.readthedocs.io/en/latest/assert_library.html

PD-13.3 Remix generate testfile





https://github.com/web3examples/ethereum/blob/master/test_examples/Mapping.sol

PD-13.4 Remix: test Mapping.sol

Mapping.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.7.0;
3
4  contract RegisterParticipants {
5      mapping(address => bool) public MapParticipant;
6      address[] public ListParticipant;
7      mapping(address => uint) public IndexInList;
8      address public owner;
9
10     constructor()  {
11         ListParticipant.push(address(0)); // "use" address 0, to make tests easier
12         owner=msg.sender;
13     }
14     function Participate(bool Join) public {
15         MapParticipant[msg.sender]=Join;
16         uint i=IndexInList[msg.sender];
17
18         if (i > 0) { // Delete previous participation entry
19             ListParticipant[i] = ListParticipant[ListParticipant.length - 1]; // switch
20             IndexInList[msg.sender]=0;
21             IndexInList[ListParticipant[i]]=i;
22             ListParticipant.pop();
23         }
24         if (Join) {
25             ListParticipant.push(msg.sender);
26             IndexInList[msg.sender]=ListParticipant.length-1;
27         }
28     }
29     function NrOfParticipants() public view returns (uint) {
30         return ListParticipant.length-1;
31     }
32 }
```

https://github.com/web3examples/ethereum/blob/master/test_examples/Mapping.sol

PD-13.4 Remix: Mapping_test.sol

Mapping_test.sol

```
pragma solidity ^0.7.0;
import "remix_tests.sol"; // this import is automatically injected by Remix.
import "remix_accounts.sol";
import "../Mapping.sol";

contract TestRegisterParticipants {
    RegisterParticipants RP;

    function beforeEach() public {  // Re-init before each test
        RP = new RegisterParticipants();
        Assert.equal(RP.NrOfParticipants(), 0, "NrOfParticipants should be 0 initially");
        Assert.equal(RP.ListParticipant(0), address(0), "ListParticipant(0) should be 0");
    }
    function AddParticipant() public {
        RP.Participate(true); // address(this) => msg.sender
        Assert.equal(RP.MapParticipant(address(this)),true,"Should be in map");
        Assert.equal(RP.NrOfParticipants(), 1, "NrOfParticipants should be 1 now");
        Assert.equal(RP.ListParticipant(1), address(this), "ListParticipant(1) should be address(this)");
        Assert.equal(RP.IndexInList(address(this)),1,"Index should be 1");
    }
    function RemoveParticipant() public {
        RP.Participate(false);
        Assert.equal(RP.NrOfParticipants(), 0, "NrOfParticipants should be 0");
    }
    function AddAndRemoveOneParticipant() public {
        AddParticipant();
        RemoveParticipant();
    }
}
```

Progress: 1 finished (of 1)

PASS **TestRegisterParticipants**

(browser/tests/Mapping_test.sol)

- ✓ Before each
- ✓ Add participant
- ✓ Before each
- ✓ Remove participant
- ✓ Before each
- ✓ Add and remove one participant

Result for browser/tests/Mapping_test.sol

Passing: 6

Total time: 4.64s

https://github.com/web3examples/ethereum/blob/master/test_examples/Mapping_test.sol

PD-13.5 Remix try and catch

sub.sol

```
pragma solidity ^0.7.0;

library SafeMathSubLib {
    function sub(uint256 _a, uint256 _b) internal pure returns (uint256) {
        require(_b <= _a,"a should be larger or equal to b");
        uint256 c = _a - _b;
        return c;
    }
}

contract SafeMathProxy { // Need external function for try/catch
    function sub(uint256 _a, uint256 _b) external pure returns (uint256) {
        return SafeMathSubLib.sub(_a,_b);
    }
}
```

SafeMath_test.sol

```
pragma solidity ^0.7.0;
import "remix_tests.sol";
import "sub.sol";

contract Test {
    SafeMathProxy safemathproxy;
    function beforeAll() public {
        safemathproxy = new SafeMathProxy();
    }
    function safeSubtractShouldRevert() public returns (bool) {
        try safemathproxy.sub(0, 1) returns (uint256 res) {
            Assert.ok(false,"Should revert");
        } catch Error(string memory reason) {
            Assert.equal(reason,"a should be larger or equal to b","Should revert");
        }
    }
    function safeSubtractShouldNotRevert() public returns (bool) {
        try safemathproxy.sub(3, 2) returns (uint256 res) {
            Assert.equal(res,1,"Should be 1");
        } catch Error(string memory reason) {
            Assert.ok(false,"Should not revert");
        }
    }
}
```

Progress: 1 finished (of 1)

PASS **Test**

(browser/tests/SafeMath_test.sol)

✓ Safe subtract should revert

✓ Safe subtract should not revert

Result for browser/tests/SafeMath_test.sol

Passing: 2

Total time: 0.69s

https://github.com/web3examples/ethereum/blob/master/test_examples/SafeMath_test.sol

PD-13.6 Truffle tests - Solidity

Solidity
Assert.equal(x, y, “error");
Assert.balanceEqual(x, y, message) Assert.balanceIsNotZero(x, message) Assert.balanceIsZero(x, message) Assert.balanceNotEqual(x, y, message) Assert.equal(x, y, message) Assert.fail(message) Assert.isAbove(x, y, message) Assert.isAtLeast(x, y, message) Assert.isAtMost(x, y, message) Assert.isBelow(x, y, message) Assert.isEmpty(x, message) Assert.isFalse(x, message) Assert.isNotEmpty(x, message) Assert.isNotZero(x, message) Assert.isTrue(x, message) Assert.isZero(x, message) Assert.lengthEqual(x, y, message) Assert.lengthNotEqual(x, y, message) Assert.notEqual(x, y, message)

https://www.trufflesuite.com/docs/truffle/testing/writing-tests-in-solidity

https://github.com/trufflesuite/truffle/blob/develop/packages/resolver/solidity/Assert.sol

PD-13.6 Solidity: before / after hooks

`beforeAll`, `beforeEach`, `afterAll` and `afterEach`

```
import "truffle/Assert.sol";

contract TestHooks {
  uint someValue;

  function beforeEach() {
    someValue = 5;
  }

  function beforeEachAgain() {
    someValue += 1;
  }

  function testSomeValueIsSix() {
    uint expected = 6;

    Assert.equal(someValue, expected, "someValue should have been 6");
  }
}
```

https://www.trufflesuite.com/docs/truffle/testing/writing-tests-in-solidity#before-after-hooks

PD-13.6 Truffle tests: MetaCoin.sol

MetaCoin.sol

```
1  pragma solidity >=0.4.25 <0.6.0;
2
3  import "./ConvertLib.sol";
4  // This is just a simple example of a coin-like contract.
5  // It is not standards compatible and cannot be expected to talk to other
6  // coin/token contracts. If you want to create a standards-compliant
7  // token, see: https://github.com/ConsenSys/Tokens. Cheers!
8
9  contract MetaCoin {
10     mapping (address => uint) balances;
11     event Transfer(address indexed _from, address indexed _to, uint256 _value);
12     constructor() public {
13         balances[tx.origin] = 10000;
14     }
15     function sendCoin(address receiver, uint amount) public returns(bool sufficient) {
16         if (balances[msg.sender] < amount) return false;
17         balances[msg.sender] -= amount;
18         balances[receiver] += amount;
19         emit Transfer(msg.sender, receiver, amount);
20         return true;
21     }
22     function getBalanceInEth(address addr) public view returns(uint){
23         return ConvertLib.convert(getBalance(addr),2);
24     }
25     function getBalance(address addr) public view returns(uint) {
26         return balances[addr];
27     }
28 }
```

https://github.com/web3examples/ethereum/tree/master/test_examples/truffle_test/contracts

PD-13.6 Truffle - solidity test

TestMetaCoin.sol

```
1  pragma solidity >=0.4.25 <0.6.0;
2
3  import "truffle/Assert.sol";
4  import "truffle/DeployedAddresses.sol";
5  import "../contracts/MetaCoin.sol";
6
7  contract TestMetaCoin {
8
9    function testInitialBalanceUsingDeployedContract() public {
10     MetaCoin meta = MetaCoin(DeployedAddresses.MetaCoin());
11
12     uint expected = 10000;
13
14     Assert.equal(meta.getBalance(tx.origin), expected, "Owner should have 10000 MetaCoin initially");
15   }
16   function testInitialBalanceWithNewMetaCoin() public {
17     MetaCoin meta = new MetaCoin();
18
19     uint expected = 10000;
20
21     Assert.equal(meta.getBalance(tx.origin), expected, "Owner should have 10000 MetaCoin initially");
22   }
23 }
```

https://github.com/web3examples/ethereum/tree/master/test_examples/truffle_test/test

PD-13.6 Testresult TestMetaCoin.sol

```
>truffle test test\TestMetaCoin.sol
Using network 'development'.

Compiling your contracts...
===========================
> Compiling .\contracts\ConvertLib.sol
> Compiling .\contracts\MetaCoin.sol
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\ConvertLib.sol
> Compiling .\contracts\MetaCoin.sol
> Compiling .\test\TestMetaCoin.sol


  TestMetaCoin
    √ testInitialBalanceUsingDeployedContract (211ms)
    √ testInitialBalanceWithNewMetaCoin (208ms)


  2 passing (14s)
```

```
>tree /f
|    LICENSE
|    truffle-config.js
|
|---build
|    \---contracts
|            ConvertLib.json
|            MetaCoin.json
|            Migrations.json
|
|---contracts
|        ConvertLib.sol
|        MetaCoin.sol
|        Migrations.sol
|
|---migrations
|        1_initial_migration.js
|        2_deploy_contracts.js
|
\---test
         metacoin.js
         TestMetaCoin.sol
```

https://github.com/web3examples/ethereum/tree/master/test_examples/truffle_test/test

PD-13.7 Truffle tests - Javascript

When we call `setValue()`, this creates a transaction. From Javascript:

```
const result = await instance.setValue(5);
```

We can call `setValue()` without creating a transaction by explicitly using `.call`:

```
const value = await instance.setValue.call(5);
```

Javascript

```
it('should ….', async () => {
  assert.equal(x, y, “error");
 });
```

assert.equal(x, y, message)
assert.notEqual(x, y, message)
….

https://mochajs.org/

https://www.chaijs.com

https://www.trufflesuite.com/docs/truffle/reference/contract-abstractions#contract-abstraction-api

https://www.trufflesuite.com/docs/truffle/reference/contract-abstractions#contract-instance-api

https://www.trufflesuite.com/docs/truffle/testing/writing-tests-in-javascript

PD-13.7 Truffle tests Javascript

metacoin.js

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

contract('MetaCoin', (accounts) => {
  it('should put 10000 MetaCoin in the first account', async () => {
    const metaCoinInstance = await MetaCoin.deployed();
    const balance = await metaCoinInstance.getBalance.call(accounts[0]);
    assert.equal(balance.valueOf(), 10000, "10000 wasn't in the first account");
  });
  it('should call a function that depends on a linked library', async () => {
    const metaCoinInstance = await MetaCoin.deployed();
    const metaCoinBalance = (await metaCoinInstance.getBalance.call(accounts[0])).toNumber();
    const metaCoinEthBalance = (await metaCoinInstance.getBalanceInEth.call(accounts[0])).toNumber();
    assert.equal(metaCoinEthBalance, 2 * metaCoinBalance, 'Library function returned unexpected function, linkage may be broken');
  });
  it('should send coin correctly', async () => {
    const metaCoinInstance = await MetaCoin.deployed();
    // Setup 2 accounts.
    const accountOne = accounts[0];
    const accountTwo = accounts[1];
    // Get initial balances of first and second account.
    const accountOneStartingBalance = (await metaCoinInstance.getBalance.call(accountOne)).toNumber();
    const accountTwoStartingBalance = (await metaCoinInstance.getBalance.call(accountTwo)).toNumber();
    // Make transaction from first account to second.
    const amount = 10;
    await metaCoinInstance.sendCoin(accountTwo, amount, { from: accountOne });
    // Get balances of first and second account after the transactions.
    const accountOneEndingBalance = (await metaCoinInstance.getBalance.call(accountOne)).toNumber();
    const accountTwoEndingBalance = (await metaCoinInstance.getBalance.call(accountTwo)).toNumber();
    assert.equal(accountOneEndingBalance, accountOneStartingBalance - amount, "Amount wasn't correctly taken from the sender");
    assert.equal(accountTwoEndingBalance, accountTwoStartingBalance + amount, "Amount wasn't correctly sent to the receiver");
  });
});
```

https://www.trufflesuite.com/docs/truffle/testing/writing-tests-in-javascript

https://github.com/web3examples/ethereum/tree/master/test_examples/truffle_test/test

PD-13.7 Testresult metacoin.js

```
>truffle test test\metacoin.js
Using network 'development'.

Compiling your contracts...
===========================
> Compiling .\contracts\ConvertLib.sol
> Compiling .\contracts\MetaCoin.sol
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\ConvertLib.sol

  Contract: MetaCoin
    √ should put 10000 MetaCoin in the first account (80ms)
    √ should call a Assert.that depends on a linked library (133ms)
    √ should send coin correctly (621ms)

  3 passing (1s)
```

```
>tree /f
|   LICENSE
|   truffle-config.js
|
+---build
|   \---contracts
|           ConvertLib.json
|           MetaCoin.json
|           Migrations.json
|
+---contracts
|       ConvertLib.sol
|       MetaCoin.sol
|       Migrations.sol
|
+---migrations
|       1_initial_migration.js
|       2_deploy_contracts.js
|
\---test
        metacoin.js
        TestMetaCoin.sol
```

https://github.com/web3examples/ethereum/tree/master/test_examples/truffle_test/test

PD-13.7 Extra functions / test events

- truffleAssert.eventEmitted(…)
- truffleAssert.eventNotEmitted(…)

https://github.com/rkalis/truffle-assertions

PD-13.7 Testresult both tests

```
>truffle test
Using network 'development'.

Compiling your contracts...
===========================
> Compiling .\contracts\ConvertLib.sol
> Compiling .\contracts\MetaCoin.sol
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\ConvertLib.sol
> Compiling .\contracts\MetaCoin.sol
> Compiling .\test\TestMetaCoin.sol

  TestMetaCoin
    √ testInitialBalanceUsingDeployedContract (170ms)
    √ testInitialBalanceWithNewMetaCoin (134ms)

  Contract: MetaCoin
    √ should put 10000 MetaCoin in the first account (78ms)
    √ should call a Assert.that depends on a linked library (132ms)
    √ should send coin correctly (603ms)

  5 passing (15s)
```

```
>tree /f
|   LICENSE
|   truffle-config.js
|
+---build
|   \---contracts
|           ConvertLib.json
|           MetaCoin.json
|           Migrations.json
|
+---contracts
|       ConvertLib.sol
|       MetaCoin.sol
|       Migrations.sol
|
+---migrations
|       1_initial_migration.js
|       2_deploy_contracts.js
|
\---test
        metacoin.js
        TestMetaCoin.sol
```

https://github.com/web3examples/ethereum/tree/master/test_examples/truffle_test/test