Informacijska Sigurnost I Blockchain Tehnologije - Projektni Zadatak Za Završni Ispit

Matija Cerovac FIDIT 2022./2023.

Tema ovog projektnog zadatka je bila stvoriti pametni ugovor koristeći Solidity koji implementira jednostavan sustav tokena te zatim i testirati njegovu valjanost.

1. Stvaranje Pametnog Ugovora

Pametni ugovor je trebao imati sljedeće značajke:

- Ukupnu količinu tokena koja se može odrediti kada se ugovor implementira
- Mogućnost prijenosa tokena s jedne adrese na drugu
- Mogućnost provjere stanja tokena za određenu adresu
- Ostale značajke koje su potrebne za stvoriti token (ime, kratica, ...) je trebalo proizvoljno definirati
- Funkcije koja vlasniku ugovora omogućuju:
    - Povećanje ukupne ponude tokena (mint)
    - Smanjenje ukupne ponude (burn)

Za implementaciju ovoga rješenja je korišten Remix IDE sustav. Ispod je predočen cjelokupan kod korišten u stvaranju ugovora te u nastavku ovog dokumenta će svaka linija biti detaljno objašnjena:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

// Definiranje pametnog ugovora
contract Token {
    string public name = "Moj Token"; // Ime tokena
    string public symbol = "MT"; // Kratica tokena
    uint8 public decimals = 18; // Broj decimalnih mjesta

    // Ukupna količina tokena
    uint256 public totalSupply;

    // Stanje tokena za svaku adresu
    mapping(address => uint256) public balanceOf;

    // Vlasnik ugovora
    address public owner;

    // Događaj koji se emitira prilikom prijenosa tokena
    event Transfer(address indexed from, address indexed to, uint256 value);

    // Konstruktor ugovora
    constructor(uint256 _initialSupply) {
        totalSupply = _initialSupply;
        balanceOf[msg.sender] = totalSupply;
```

```
        owner = msg.sender;
    }

    // Funkcija za prijenos tokena s jedne adrese na drugu
    function transfer(address _to, uint256 _value) public returns (bool success) {
        require(_to != address(0), "Ne mozete slati token na adresu 0x0");
        require(balanceOf[msg.sender] >= _value, "Nemate dovoljno tokena na
racunu");

        balanceOf[msg.sender] -= _value;
        balanceOf[_to] += _value;
        emit Transfer(msg.sender, _to, _value);
        return true;
    }

    // Funkcija koja omogućuje vlasniku ugovora povećanje ukupne ponude tokena
(mint)
    function mint(uint256 _value) public returns (bool success) {
        require(msg.sender == owner, "Samo vlasnik moze mjenjati ukupnu ponudu
tokena");

        totalSupply += _value;
        balanceOf[msg.sender] += _value;
        emit Transfer(address(0), msg.sender, _value);
        return true;
    }

    // Funkcija koja omogućuje vlasniku ugovora smanjenje ukupne ponude tokena
(burn)
    function burn(uint256 _value) public returns (bool success) {
        require(msg.sender == owner, "Samo vlasnik moze mjenjati ukupnu ponudu
tokena");
        require(balanceOf[msg.sender] >= _value, "Nemate dovoljno tokena za
spaljivanje");

        totalSupply -= _value;
        balanceOf[msg.sender] -= _value;
        emit Transfer(msg.sender, address(0), _value);
        return true;
    }
}
```

U nastavku slijedi detaljan opis koda.

```
// SPDX-License-Identifier: MIT
```

Ova linija je komentar koja sadrži deklaraciju licenciranja koda. SPDX-License-Identifier je standardna oznaka koja se koristi za označavanje licence pod kojom je objavljen kod. Za potrebe ovog rada odabrana je MIT licenca.

```
pragma solidity ^0.8.0;
```

Ova linija definira verziju Solidity programskog jezika koja će se koristiti za kompilaciju ugovora. Ovdje se koristi verzija 0.8.0 i "^" označava da se koristi 0.8.0 i više verzije, ali ne više od 0.9.0.

```
// Definiranje pametnog ugovora
contract Token {
```

Ovdje počinje deklaracija pametnog ugovora, koji će se zvati "Token".

```
string public name = "Moj Token"; // Ime tokena
```

Ovo je deklaracija varijable "name" koja će sadržavati ime tokena. "public" znači da će ova varijabla biti dostupna izvan ugovora, pa bilo tko može pročitati ime tokena.

```
string public symbol = "MT"; // Kratica tokena
```

Ovo je deklaracija varijable "symbol" koja će sadržavati kraticu tokena. Također je označena kao "public" kako bi bila dostupna izvan ugovora.

```
uint8 public decimals = 18; // Broj decimalnih mjesta
```

Ovdje se deklarira varijabla "decimals" koja će sadržavati broj decimalnih mjesta za token. Standardno se koristi 18 decimalnih mjesta u Ethereumu, ali možete ga prilagoditi prema vašim potrebama. Također je označena kao "public" kako bi bila dostupna izvan ugovora.

```
// Ukupna količina tokena
uint256 public totalSupply;
```

Ovo je deklaracija varijable "totalSupply" koja će sadržavati ukupnu količinu tokena. Također je označena kao "public" kako bi bila dostupna izvan ugovora.

```
// Stanje tokena za svaku adresu
mapping(address => uint256) public balanceOf;
```

Ovo je deklaracija "mappinga" koji će se koristiti za praćenje stanja tokena za svaku adresu. "mapping" je posebna vrsta podatkovne strukture koja povezuje adrese s njihovim stanjem tokena. Također je označena kao

"public" kako bi bila dostupna izvan ugovora.

```
// Vlasnik ugovora
address public owner;
```

Ovo je deklaracija varijable "owner" koja će sadržavati adresu vlasnika ugovora. Također je označena kao "public" kako bi bila dostupna izvan ugovora.

```
// Događaj koji se emitira prilikom prijenosa tokena
event Transfer(address indexed from, address indexed to, uint256 value);
```

Ovdje se definira događaj "Transfer" koji će se emitirati svaki put kad se tokeni premještaju s jedne adrese na drugu. Ovo omogućuje praćenje i praćenje prijenosa tokena.

```
// Konstruktor ugovora s argumentom za inicijalnu opskrbu
constructor(uint256 _initialSupply) {
    totalSupply = _initialSupply;
    balanceOf[msg.sender] = totalSupply;
    owner = msg.sender;
}
```

Ovo je konstruktor ugovora koji se izvršava prilikom stvaranja ugovora. Konstruktor prima argument "_initialSupply", što je broj tokena koje želite da ugovor ima. Ovdje se postavljaju ukupna opskrba tokenima, stanje tokena vlasnika ugovora (msg.sender) i postavlja se vlasnik ugovora na adresu koja ga je stvorila.

```
// Funkcija za prijenos tokena s jedne adrese na drugu
function transfer(address _to, uint256 _value) public returns (bool success) {
    require(_to != address(0), "Ne mozete slati token na adresu 0x0");
    require(balanceOf[msg.sender] >= _value, "Nemate dovoljno tokena na racunu");

    balanceOf[msg.sender] -= _value;
    balanceOf[_to] += _value;
    emit Transfer(msg.sender, _to, _value);
    return true;
}
```

Ovo je funkcija "transfer" koja omogućuje korisnicima da šalju token s jedne adrese na drugu. Funkcija provjerava nekoliko uvjeta prije nego što omogući prijenos, uključujući provjeru da se tokeni ne šalju na adresu 0x0 i da pošiljatelj ima dovoljno tokena na svom računu. Ako su uvjeti ispunjeni, funkcija izvršava prijenos i emitira događaj "Transfer" kako bi obavijestila druge o prijenosu.

```
// Funkcija koja omogućuje vlasniku ugovora povećanje ukupne ponude tokena (mint)
function mint(uint256 _value) public returns (bool success) {
```

```
    require(msg.sender == owner, "Samo vlasnik moze mjenjati ukupnu ponudu
tokena");

    totalSupply += _value;
    balanceOf[msg.sender] += _value;
    emit Transfer(address(0), msg.sender, _value);
    return true;
}
```

Ovo je funkcija "mint" koja omogućuje vlasniku ugovora da poveća ukupnu opskrbu tokenima. Funkcija provjerava da je pozivač funkcije vlasnik ugovora i zatim povećava ukupnu opskrbu tokenima i dodaje nove tokene na račun vlasnika. Emitira se događaj "Transfer" kako bi se obavijestilo o povećanju opskrbe tokenima.

```
// Funkcija koja omogućuje vlasniku ugovora smanjenje ukupne ponude tokena (burn)
function burn(uint256 _value) public returns (bool success) {
    require(msg.sender == owner, "Samo vlasnik moze mjenjati ukupnu ponudu
tokena");
    require(balanceOf[msg.sender] >= _value, "Nemate dovoljno tokena za
spaljivanje");

    totalSupply -= _value;
    balanceOf[msg.sender] -= _value;
    emit Transfer(msg.sender, address(0), _value);
    return true;
}
```

Ovo je funkcija "burn" koja omogućuje vlasniku ugovora da smanji ukupnu opskrbu tokenima. Funkcija provjerava da je pozivač funkcije vlasnik ugovora i da vlasnik ima dovoljno tokena za spaljivanje. Nakon toga, smanjuje se ukupna opskrba tokenima i umanjuje se stanje tokena vlasnika. Emitira se događaj "Transfer" kako bi se obavijestilo o smanjenju opskrbe tokenima i prijenosu tokena na adresu 0x0 ("burning").

2. Testiranje na Ganache testnoj mreži

Sljedeći zadatak je bio testirati ispravnost pametnog ugovora na Ganache mreži. Sam Ganache je alat za razvoj i testiranje pametnih ugovora na Ethereum mreži u lokalnom okruženju. Omogućuje stvaranje i upravljanje privatnom Ethereum mrežom na računalu. Ganache dolazi s vlastitim Ethereum čvorovima, računima s Etherom i korisničkim sučeljem za praćenje stanja ugovora i izvođenje transakcija. Ovaj alat je posebno koristan za razvoj i testiranje pametnih ugovora bez potrebe za stvarnim Etherom.

Elementi koji su se morali implementirati su sljedeći:

1. Stvaranje tokena
2. Funkcija za mint i burn
3. Prijenos tokena s jedne adrese na drugu
4. Provjera stanja prije i nakon transakcije

2.1. Stvaranje tokena

Za potrebe ovog testiranja se koristio Remix VN (Shanghai) Environment. Kod pokretanja pametnog ugovora stavila se ukupna količina tokena na 1000.

```
[vm]from: 0x5B3...eddC4to: Token.(constructor)value: 0 weidata: 0x608...003e8logs:
0hash: 0xc11...5ded4
status  true Transaction mined and execution succeed
transaction hash
0xc113a538d6ec848a97e6c09eec620cc81430a57a2b07d2681678df2ad4f5ded4
block hash  0x5d01260700abe56f4c8b274dd8d43134926736ad407943d8b506d0fbc89a5129
block number    1
contract address    0xd9145CCE52D386f254917e481eB44e9943F39138
from    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to  Token.(constructor)
gas 1150406 gas
transaction cost    1000669 gas
execution cost  872937 gas
input   0x608...003e8
decoded input   {
    "uint256 _initialSupply": "1000"
}
decoded output   -
logs    []
val 0 wei
```

Nakon stvaranja ugovora i kasnije kod svih transakcija se može primjetiti kako se troši stanj etheriuma za svaku račun. Račun na kojem je ugovor pokrenut je "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4".

2.2. Funkcija mint i burn

Za testiranje funkcionalnosti mint se dodalo 1000 tokena zbog čega je konačan broj porastao na 2000. Ukupna količina prije minta:

```
CALL
[call]from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4to: Token.totalSupply()data:
0x181...60ddd
from    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to  Token.totalSupply() 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost  2448 gas (Cost only applies when called by a contract)
input   0x181...60ddd
decoded input   {}
decoded output  {
    "0": "uint256: 1000"
}
logs    []
```

Zvanje mint funkcije:

```
[vm]from: 0x5B3...eddC4to: Token.mint(uint256) 0xd91...39138value: 0 weidata:
0xa07...003e8logs: 1hash: 0x98f...8b108
status  true Transaction mined and execution succeed
transaction hash
0x98f5c96c73434aa0e8db3523a8dc04041dd68270352f84de921f646bb8f8b108
block hash  0x778bd80de010ce01ee00795cbf081a4da2dae29f05c01ec75a352bca578c1f49
block number    2
from    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to  Token.mint(uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas 41984 gas
transaction cost    36507 gas
execution cost  15291 gas
input   0xa07...003e8
decoded input   {
    "uint256 _value": "1000"
}
decoded output  {
    "0": "bool: success true"
}
logs    [
    {
        "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
        "topic":
"0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
        "event": "Transfer",
        "args": {
            "0": "0x0000000000000000000000000000000000000000",
            "1": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
            "2": "1000",
            "from": "0x0000000000000000000000000000000000000000",
            "to": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
            "value": "1000"
        }
    }
]
val 0 wei
```

Ukupna količina nakon minta:

```
CALL
[call]from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4to: Token.totalSupply()data:
0x181...60ddd
from    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to  Token.totalSupply() 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost  2448 gas (Cost only applies when called by a contract)
input   0x181...60ddd
decoded input   {}
decoded output  {
    "0": "uint256: 2000"
}
logs    []
```

Za testiranje funkcionalnosti burn oduzelo se 500 tokena zbog čega se konačan broj spustio na 1500. Zvanje burn funkcije:

```
[vm]from: 0x5B3...eddC4to: Token.burn(uint256) 0xd91...39138value: 0 weidata:
0x429...001f4logs: 1hash: 0x512...ebb08
status  true Transaction mined and execution succeed
transaction hash
0x512ffa83684dec2cbfe10b15025a56f26ae35bbc3ac88929f4ae48639bbebb08
block hash  0x3a8e094af3ea5ba1f021c4e74f2b6b2dd5557cd4f1345f423198c4ef92b5e6e5
block number    3
from    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to  Token.burn(uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas 42230 gas
transaction cost    36721 gas
execution cost  15505 gas
input   0x429...001f4
decoded input   {
    "uint256 _value": "500"
}
decoded output  {
    "0": "bool: success true"
}
logs    [
    {
        "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
        "topic":
"0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
        "event": "Transfer",
        "args": {
            "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
            "1": "0x0000000000000000000000000000000000000000",
            "2": "500",
            "from": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
            "to": "0x0000000000000000000000000000000000000000",
            "value": "500"
        }
    }
]
val 0 wei
```

Stanje nakon burn funkcije:

```
CALL
[call]from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4to: Token.totalSupply()data:
0x181...60ddd
from    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to  Token.totalSupply() 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost  2448 gas (Cost only applies when called by a contract)
```

```
input   0x181...60ddd
decoded input   {}
decoded output  {
    "0": "uint256: 1500"
}
logs    []
```

2.3. Prijenos tokena s jedne adrese na drugu i Provjera stanja prije i nakon transakcije

Za ovo testiranje korišteni su računi: "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4" koji ćemo dalje zvati račun 1 i "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2" kojega ćemo dalje zvati račun 2.

Provjera stanja računa 1 prije transakcije: logs []

```
CALL
[call]from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2to:
Token.balanceOf(address)data: 0x70a...eddc4
from    0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to  Token.balanceOf(address) 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost  2802 gas (Cost only applies when called by a contract)
input   0x70a...eddc4
decoded input   {
    "address ": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
}
decoded output  {
    "0": "uint256: 1500"
}
logs    []
logs    []
```

Provjera stanja računa 2 prije transakcije:

```
CALL
[call]from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2to:
Token.balanceOf(address)data: 0x70a...35cb2
from    0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to  Token.balanceOf(address) 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost  2802 gas (Cost only applies when called by a contract)
input   0x70a...35cb2
decoded input   {
    "address ": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2"
}
decoded output  {
    "0": "uint256: 0"
}
logs    []
```

Sa računa 1 ćemo na račun 2 prenijeti 250 tokena. Poziv transfer funkcije:

```
[vm]from: 0x5B3...eddC4to: Token.transfer(address,uint256) 0xd91...39138value: 0
weidata: 0xa90...000falogs: 1hash: 0x336...e61bb
status  true Transaction mined and execution succeed
transaction hash
0x336db606771858f118215da9efe00d3f743a2586ffcf32b94cf996cac9de61bb
block hash  0xf2e2e0fc56ce1ff7187b3323da7a005b520e0feb3a5de4fabb8176da2e476e68
block number    5
from    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to  Token.transfer(address,uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas 60228 gas
transaction cost    52372 gas
execution cost  30800 gas
input   0xa90...000fa
decoded input   {
    "address _to": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
    "uint256 _value": "250"
}
decoded output  {
    "0": "bool: success true"
}
logs    [
    {
        "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
        "topic":
"0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
        "event": "Transfer",
        "args": {
            "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
            "1": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
            "2": "250",
            "from": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
            "to": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
            "value": "250"
        }
    }
]
val 0 wei
```

Nakon obavljanja transakcije možemo vidjeti kako se količina tokena na računu 1 smanjila na 1250, dok se količina na računu 2 povećala na 250 što ukazuje da je transakcija uspješno izvedena.

Provjera stanja računa 1 nakon transakcije:

```
CALL
[call]from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4to:
Token.balanceOf(address)data: 0x70a...eddc4
from    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to  Token.balanceOf(address) 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost  2802 gas (Cost only applies when called by a contract)
input   0x70a...eddc4
```

```
decoded input   {
    "address ": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
}
decoded output  {
    "0": "uint256: 1250"
}
logs    []
```

Provjera stanja računa 2 nakon transakcije:

```
CALL
[call]from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4to:
Token.balanceOf(address)data: 0x70a...35cb2
from    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to  Token.balanceOf(address) 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost  2802 gas (Cost only applies when called by a contract)
input   0x70a...35cb2
decoded input   {
    "address ": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2"
}
decoded output  {
    "0": "uint256: 250"
}
logs    []
```

Također ako pogledamo na količinu etheriuma na svakom računu možemo vidjeti da se smanjila na oba računa zbog naplate gasa, što nam ukazuje na korektan rad pametnog ugovora.

Link na repozitorij sa potpunim kodom.