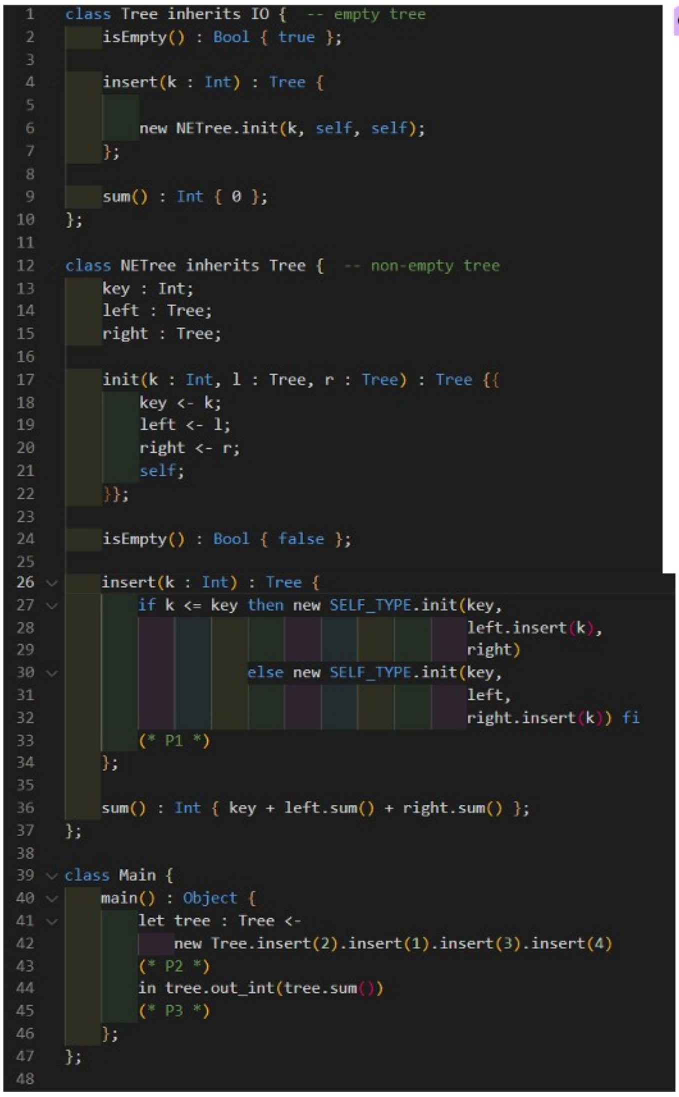
S3(7) - Exemplu

Tuesday, January 10, 2023   12:00 PM

EXEMPLU EXAMEN

1



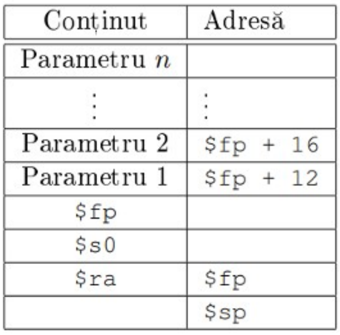
```
class Tree inherits IO {  -- empty tree
    isEmpty() : Bool { true };

    insert(k : Int) : Tree {

        new NETree.init(k, self, self);
    };

    sum() : Int { 0 };
};

class NETree inherits Tree {  -- non-empty tree
    key : Int;
    left : Tree;
    right : Tree;

    init(k : Int, l : Tree, r : Tree) : Tree {{
        key <- k;
        left <- l;
        right <- r;
        self;
    }};

    isEmpty() : Bool { false };

    insert(k : Int) : Tree {
        if k <= key then new SELF_TYPE.init(key,
                                            left.insert(k),
                                            right)
                    else new SELF_TYPE.init(key,
                                            left,
                                            right.insert(k)) fi
        (* P1 *)
    };

    sum() : Int { key + left.sum() + right.sum() };
};

class Main {
    main() : Object {
        let tree : Tree <-
            new Tree.insert(2).insert(1).insert(3).insert(4)
        (* P2 *)
        in tree.out_int(tree.sum())
        (* P3 *)
    };
};
```

1 Organizare directă în memorie

Tree: tag 0
dim 3
Tree_dispTab → Object. ... x3
IO. ... x4
Tree.isEmpty
Tree.insert
Tree.sum

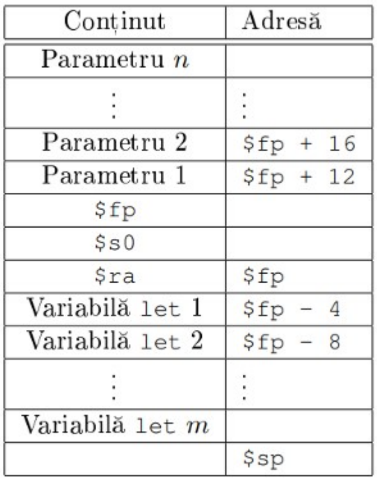
NETree : tag 1
dim 6
NETree_dispTab → Object. ... x3
key
left
right
IO. ... x4
NETree.isEmpty
NETree.insert
NETree.sum
NETree.init

Înregistrarea de activare



Conținut	Adresă
Parametru n	
⋮	⋮
Parametru 2	`$fp + 16`
Parametru 1	`$fp + 12`
`$fp`	
`$s0`	
`$ra`	`$fp`
	`$sp`

Înregistrarea de activare cu variabile LET



Conținut	Adresă
Parametru n	
⋮	⋮
Parametru 2	`$fp + 16`
Parametru 1	`$fp + 12`
`$fp`	
`$s0`	
`$ra`	`$fp`
Variabilă `let` 1	`$fp - 4`
Variabilă `let` 2	`$fp - 8`
⋮	⋮
Variabilă `let` m	
	`$sp`

2. Dimensiunea minimă a înreg. de activare pt. metoda sum din NETree?

1. regiștri : $fp, $s0, $ra => 3
2. parametri : 0
3. locații temporare: 1

Key + left. sum() + right. sum <=> ((Key + left. sum()) + right. sum())

Key + left. sum() => NT = max(NT(Key), 1+ NT(left. sum()))

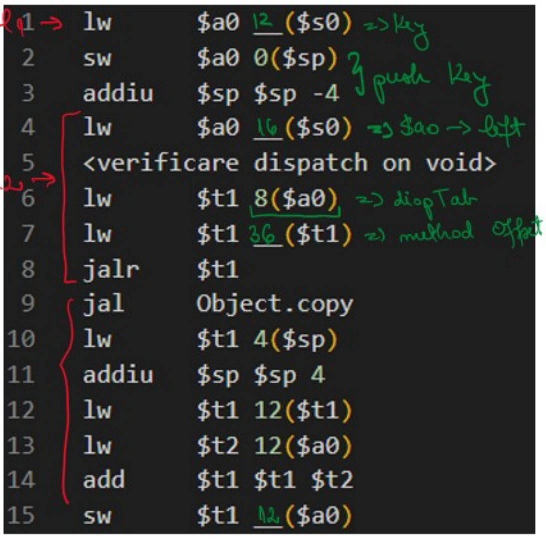
= max(0, 1+0) = 1

Key + left. sum() + right. sum() => NT = max(1, 1+NT(right. sum()))

= max(1, 1+0) = 1

=> DIMENSIUNE : 4 cuvinte

3 Completați spațiile libere : (codul MIPS de mai jos este propus pentru Key + left.sum() din metoda sum() a clasei NETree



```
lw      $a0 12($s0)
sw      $a0 0($sp)
addiu   $sp $sp -4
lw      $a0 16($s0)
<verificare dispatch on void>
lw      $t1 8($a0)
lw      $t1 36($t1)
jalr    $t1
jal     Object.copy
lw      $t1 4($sp)
addiu   $sp $sp 4
lw      $t1 12($t1)
lw      $t2 12($a0)
add     $t1 $t1 $t2
sw      $t1 12($a0)
```

l1 -> 1 lw $a0 12($s0) => Key

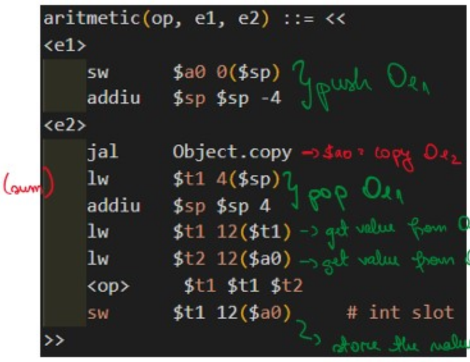
} push Key

lw $a0 16($s0) => $a0 -> left

l2 ->

lw $t1 8($a0) => dispTab

lw $t1 36($t1) => method offset (sum)



```
aritmetic(op, e1, e2) ::= <<
<e1>
    sw      $a0 0($sp)
    addiu   $sp $sp -4
<e2>
    jal     Object.copy
    lw      $t1 4($sp)
    addiu   $sp $sp 4
    lw      $t1 12($t1)
    lw      $t2 12($a0)
    <op>     $t1 $t1 $t2
    sw      $t1 12($a0)     # int slot
>>
```

} push Oe1

Object.copy -> $a0 = copy Oe2 => poate fi -> Int(...) / Bool(...), dacă op este "="

} pop Oe1

lw $t1 12($t1) -> get value from Oe1

lw $t2 12($a0) -> get value from Oe2

sw $t1 12($a0) -> store the value at offset 12 in the new obj ($a0)

Oe1 = ob care rezultă în urma evaluării lui e1

Int:		
0	tag	0
4	dim	4
8	Int_dispTab	
12	0	

Key + left. sum()

1. Key = primul atrib => $a0 + 12
2. left = al doilea atrib => $a0 + 16
3. sum() = offset 36 in dispTab

4. Semantica operațională

new SELF_TYPE din insert (NETree) => Tree. insert(1 => ... new SELF_TYPE

$$\frac{\begin{array}{l} T_0 = \begin{cases} X & \text{if } T = \textbf{SELF\_TYPE} \text{ and } so = X(\ldots) \\ T & \text{otherwise} \end{cases} \\ class(T_0) = (a_1 : T_1 \leftarrow e_1, \ldots, a_n : T_n \leftarrow e_n) \\ l_i = newloc(S_1), \text{ for } i = 1 \ldots n \text{ and each } l_i \text{ is distinct} \\ v_1 = T_0(a_1 = l_1, \ldots, a_n = l_n) \\ S_2 = S_1[D_{T_1}/l_1, \ldots, D_{T_n}/l_n] \\ \cancel{v_1, S_2, [a_1 : l_1, \ldots, a_n : l_n] \vdash \{a_1 \leftarrow e_1; \ldots; a_n \leftarrow e_n;\} \mapsto v_2, S_3} \end{array}}{so, S_1, E \vdash \text{new } T \mapsto v_1, S_3}$$

=> T0 = NETree

=> class (NETree) = (Key: Int, left: Tree, right: Tree)

=> l1, l2, l3 = new loc (S)

=> v1 = NETree(Key = l1, left = l2, right = l3)

=> S2 = S1 [Int(0)/l1, void/l2, void/l3]

-> nu avem expr. de inițializare pt. atribute

T -> SELF-TYPE

so = NETree (Key = lKey, left = ll, right = lr)

E = [Key : lKey, left : ll, right : lr, k : lk]

S = [lKey -> Int(2), ll -> void, lr -> void, lk -> Int(1)]