SEMINAR 2

miercuri, 30 noiembrie 2022 22:46

REGULI DE TIPARE

$$\frac{\vdots \quad \Big\}\ \text{ipoteze}}{\underbrace{O,M,C}_{\text{context}} \vdash \underbrace{e : T}_{\text{statement}}}$$

Cum citim? => În contextul de tipare pentru obiecte O, metode M, și care conține clasa C, expresia e are tipul T.

+ dacă ipotezele pentru subexpresiile lui e se satisfac, atunci statementul este adevărat

TYPE ENVIRONMENT / CONTEXT

=> are 3 părți
- M -> env. Metode
- O -> env. Obiecte
- C -> clasa curentă în care se află expresia

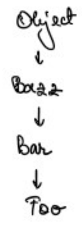
O => $O(v) = T$ => obiectul v are tipul T

M => $M(C,f) = (\underbrace{T_1, \dots T_m}_{\text{param}}, \underset{\substack{\downarrow \\ \text{retur}}}{T_{m+1}})$ => în clasa C, metoda f are param. formali cu tipurile $T_1 \dots T_m$ și tipul de retur T_{m+1}

C -> numele clasei curente

EXERCIȚII

1. Fie următoarea ierarhie de clase:



Să se determine tipul următoarelor expresii:

a)
```
case 3 of
    y : Int => y;
    z : Bool => z;
esac;
```

b) if 0 = 1 then new Bar else new Foo fi;

c) let x : Bazz <- new Bar in x;

a)

$$\frac{\begin{array}{l} O, M, C \vdash \overset{3}{e_0} : \overset{\text{Int}}{T_0} \\ O[T_1/x_1], M, C \vdash \overset{y}{e_1} : T_1'^{\ \text{Int}} \\ \vdots \\ O[T_n/x_n], M, C \vdash \overset{z}{e_n} : \overset{\text{Bool}}{T_n'} \end{array}}{O, M, C \vdash \text{case } e_0 \text{ of } x_1 : T_1 \Rightarrow e_1; \dots x_n : T_n \Rightarrow e_n;\ \text{esac} : \bigsqcup_{1 \le i \le n} T_i'} \quad [\text{Case}]$$

```
case 3 of
    y : Int => y;
    z : Bool => z;
esac;
```

lub (Int, Bool) = Object

b) if 0 = 1 then new Bar else new Foo fi

$$\frac{\begin{array}{l} O, M, C \vdash e_1 : Bool \Rightarrow 0=1 : \text{Bool} \\ O, M, C \vdash e_2 : T_2 \Rightarrow \text{new Bar} : \text{Bar} \\ O, M, C \vdash e_3 : T_3 \Rightarrow \text{new Foo} : \text{Foo} \end{array}}{O, M, C \vdash \text{if } \underset{\downarrow\atop 0=1}{e_1} \text{ then } \underset{\downarrow\atop \text{new Bar}}{e_2} \text{ else } \underset{\downarrow\atop \text{new Foo}}{e_3} \text{ fi} : T_2 \sqcup T_3} \Rightarrow \text{lub}(\text{Bar}, \text{Foo}) = \text{Bar}$$

0 = 1 : Bool

$$\frac{\begin{array}{l} O, M, C \vdash e_1 : T_1 \Rightarrow 0 : \text{Int} \\ O, M, C \vdash e_2 : T_2 \Rightarrow 1 : \text{Int} \\ T_1 \in \{Int, String, Bool\} \vee T_2 \in \{Int, String, Bool\} \Rightarrow T_1 = T_2 \end{array} \quad \text{Int} = \text{Int}}{O, M, C \vdash \underset{\downarrow\atop 0}{e_1} = \underset{\downarrow\atop 1}{e_2} : Bool}$$

new Bar : Bar

$$\frac{T' = \begin{cases} \text{SELF\_TYPE}_C & \text{if } T = \text{SELF\_TYPE} \\ T & \text{otherwise} \end{cases} \Rightarrow T' = \text{Bar}}{O, M, C \vdash new\ \underset{\downarrow\atop \text{Bar}}{T} : T' \Rightarrow \text{Bar}}$$

new Foo : Foo

$$\frac{T' = \begin{cases} \text{SELF\_TYPE}_C & \text{if } T = \text{SELF\_TYPE} \\ T & \text{otherwise} \end{cases} \Rightarrow T' = \text{Foo}}{O, M, C \vdash new\ \underset{\downarrow\atop \text{Foo}}{T} : T' \Rightarrow \text{Foo}}$$

c) let x : Bazz ← new Bar in x;

$$\frac{\begin{array}{l} T_0' = \begin{cases} \text{SELF\_TYPE}_C & \text{if } T_0 = \text{SELF\_TYPE} \\ T_0 & \text{otherwise} \end{cases} \Rightarrow T_0' = \text{Bazz} \\ O, M, C \vdash e_1 : T_1 \Rightarrow T_1 = \text{Bar} \\ T_1 \leq T_0' \Rightarrow \text{Bar} \leq \text{Bazz} \checkmark \\ O[T_0'/x], M, C \vdash e_2 : T_2 \Rightarrow T_2 = \text{Bazz} \end{array}}{O, M, C \vdash \text{let } \underset{\downarrow\atop x}{x} : \underset{\downarrow\atop \text{Bazz}}{T_0} \leftarrow \underset{\downarrow\atop \text{new Bar}}{e_1} \text{ in } \underset{\searrow\atop x}{e_2} : T_2} = \text{Bazz}$$

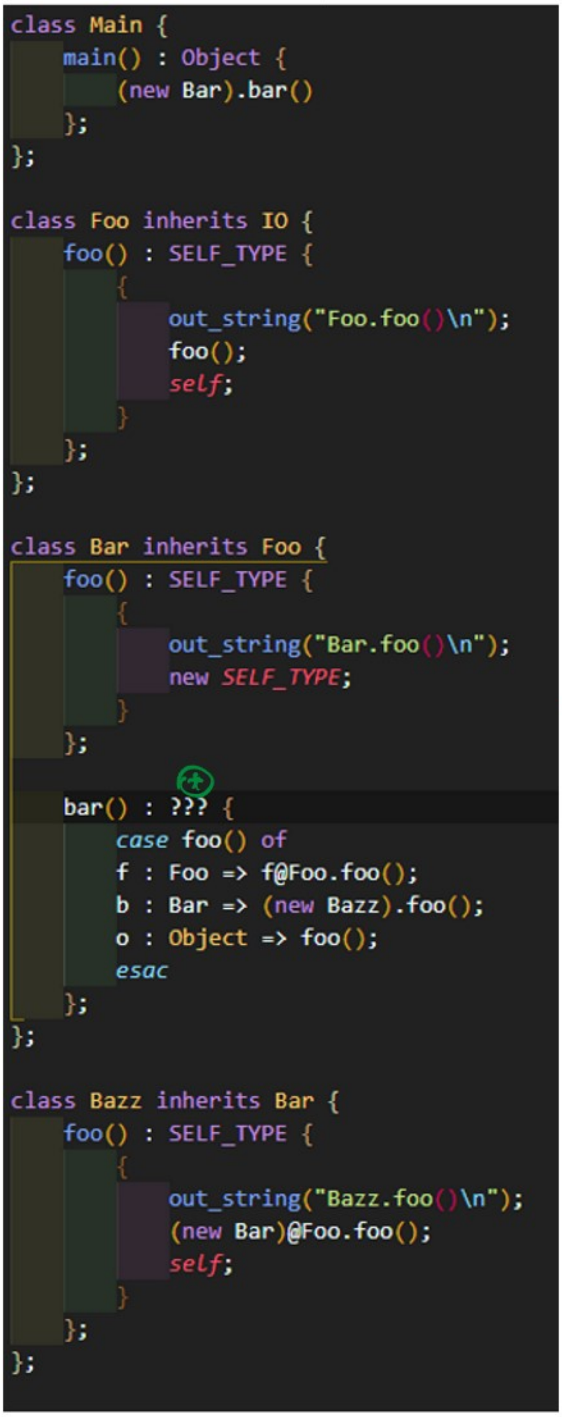
$O\{T_0'/x\}$ înseamnă:

Contextul de tipare O se extinde cu $x : T_0'$

OBS

$$\boxed{\begin{array}{l} O\{T/x\}(x) = T \\ O\{T/x\}(y) = O(y), \text{ dacă } x \neq y \end{array}}$$

② Fie programul Cool de mai jos. Cu ce putem înlocui '???'



```
class Main {
    main() : Object {
        (new Bar).bar()
    };
};

class Foo inherits IO {
    foo() : SELF_TYPE {
        {
            out_string("Foo.foo()\n");
            foo();
            self;
        }
    };
};

class Bar inherits Foo {
    foo() : SELF_TYPE {
        {
            out_string("Bar.foo()\n");
            new SELF_TYPE;
        }
    };

    bar() : ??? {
        case foo() of
        f : Foo => f@Foo.foo();
        b : Bar => (new Bazz).foo();
        o : Object => foo();
        esac
    };
};

class Bazz inherits Bar {
    foo() : SELF_TYPE {
        {
            out_string("Bazz.foo()\n");
            (new Bar)@Foo.foo();
            self;
        }
    };
};
```

⊕

=> trebuie să determinăm ce tip poate întoarce bar()

=> trebuie să determinăm tipul expr. 'case'

DETERMINARE CONTEXTE DE TIPARE

=> Cine sunt O și M în punctul ⊕ ?

O(self) = SELF_TYPE$_{Bar}$

M(Main, main) = Object

M(Foo, foo) = SELF_TYPE

M(Bar, foo) = SELF_TYPE

M(Bar, bar) = ?

M(Bazz, foo) = SELF_TYPE

M(Bazz, bar) = ?

+ metode moștenite de la clase default (io, Object)

M(Foo, out-string) = (String, SELF-TYPE)

M(Foo, out-int) = (Int, SELF-TYPE)

M(Foo, in-string) = String

M(Foo, in-int) = Int

.... => similar pentru Bar, Bazz

..... => + metodele lui Object

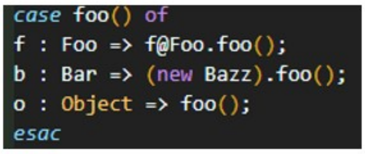
$$\frac{\begin{array}{l} O, M, C \vdash e_0 : T_0 \\ O[T_1/x_1], M, C \vdash e_1 : T_1' \\ \vdots \\ O[T_n/x_n], M, C \vdash e_n : T_n' \end{array}}{O, M, C \vdash \text{case } \underset{\text{foo()}}{e_0} \text{ of } \underset{f}{x_1} : \underset{\text{Foo}}{T_1} \Rightarrow \underset{\text{f@Foo.foo()} \dots}{e_1}; \dots x_n : T_n \Rightarrow e_n;\ \text{esac} : \bigsqcup_{1 \le i \le n} T_i'}$$

O,M,C ⊢ foo() : SELF_TYPE$_{Bar}$

O[Foo/f], M,C ⊢ f@Foo.foo() : Foo

O[Bar/b], M,C ⊢ (new Bazz).foo() : Bazz

O[Object/o], M,C ⊢ foo() : SELF_TYPE$_{Bar}$



```
case foo() of
f : Foo => f@Foo.foo();
b : Bar => (new Bazz).foo();
o : Object => foo();
esac
```

: lub (Foo, Bazz, SELF_TYPE$_{Bar}$) = Foo

foo() <=> self.foo()

$$M(\text{Bar}, \text{foo}) = \text{SELF\_TYPE}$$

$$\frac{\begin{array}{l}
O, M, C \vdash e_0 : T_0 \quad \Rightarrow T_0 = \text{SELF\_TYPE}_{\text{Bar}} \\
O, M, C \vdash \cancel{e_1} : T_1 \\
\vdots \\
O, M, C \vdash \cancel{e_n} : T_n \\
T_0' = \begin{cases} C & \text{if } T_0 = \text{SELF\_TYPE}_C \\ T_0 & \text{otherwise} \end{cases} \Bigg\} \Rightarrow T_0' = \text{Bar} \\
M(T_0', f) = (T_1', \ldots, T_n', T_{n+1}') \Rightarrow M(\text{Bar}, \text{foo}) = \text{SELF\_TYPE} = T_{n+1}' \\
T_i \le T_i' \quad 1 \le i \le n \quad (\text{nu avem param}) \\
T_{n+1} = \begin{cases} T_0 & \text{if } T_{n+1}' = \text{SELF\_TYPE} \\ T_{n+1}' & \text{otherwise} \end{cases} \Bigg\} \Rightarrow T_{n+1} = T_0 = \text{SELF\_TYPE}_{\text{Bar}}
\end{array}}{O, M, C \vdash \underset{\substack{\downarrow \\ \text{Bar}}}{e_0}.\underset{\substack{\downarrow \\ \text{self}}}{f}\underset{\substack{\downarrow \\ \text{foo}}}{(}\cancel{e_1}, \ldots, \cancel{e_n}) : T_{n+1}} = \text{SELF\_TYPE}_{\text{Bar}}$$

f@Foo.foo() + O[Foo|f] : Foo

$$M(\text{Foo}, \text{foo}) = \text{SELF\_TYPE}$$

$$\frac{\begin{array}{l}
O, M, C \vdash e_0 : T_0 \quad \Rightarrow T_0 = \text{Foo} \\
O, M, C \vdash \cancel{e_1} : T_1 \\
\vdots \\
O, M, C \vdash \cancel{e_n} : T_n \\
T_0 \le T \Rightarrow \text{Foo} \le \text{Foo} \ \checkmark \\
M(T, f) = (T_1', \ldots, T_n', T_{n+1}') \quad \Rightarrow T_{n+1}' = \text{SELF\_TYPE} \\
T_i \le T_i' \quad 1 \le i \le n \Rightarrow \text{nu avem param} \\
T_{n+1} = \begin{cases} T_0 & \text{if } T_{n+1}' = \text{SELF\_TYPE} \\ T_{n+1}' & \text{otherwise} \end{cases} \Bigg\} \Rightarrow T_{n+1} = \text{Foo}
\end{array}}{O, M, C \vdash \underset{\substack{\downarrow \\ \text{f}}}{e_0}@\underset{\substack{\downarrow \\ \text{Foo}}}{T}.\underset{\substack{\downarrow \\ \text{foo}}}{f}(\cancel{e_1}, \ldots, \cancel{e_n}) : T_{n+1}} = \text{Foo}$$

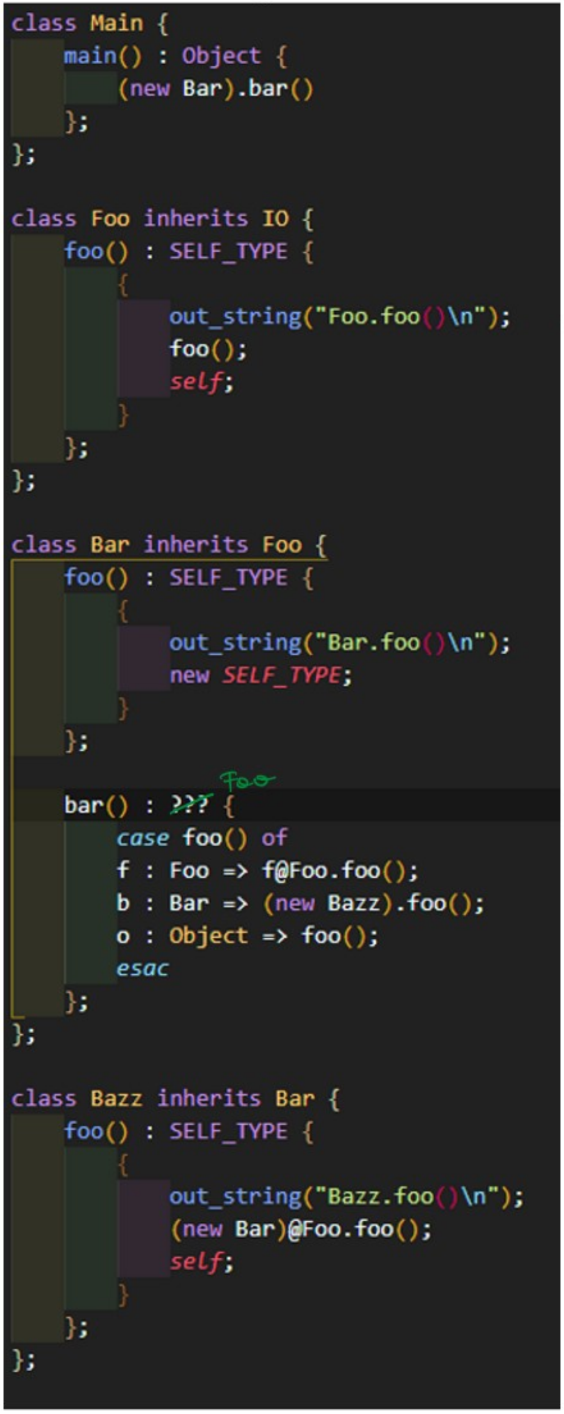
(new Bazz).foo() + O[Bar|b] = Bazz

$$M(\text{Bazz}, \text{foo}) = \text{SELF\_TYPE}$$

$$\frac{\begin{array}{l}
O, M, C \vdash e_0 : T_0 \quad \Rightarrow T_0 = \text{Bazz} \\
O, M, C \vdash \cancel{e_1} : T_1 \\
\vdots \\
O, M, C \vdash \cancel{e_n} : T_n \\
T_0' = \begin{cases} C & \text{if } T_0 = \text{SELF\_TYPE}_C \\ T_0 & \text{otherwise} \end{cases} \Bigg\} \Rightarrow T_0' = \text{Bazz} \\
M(T_0', f) = (T_1', \ldots, T_n', T_{n+1}') \quad T_{n+1}' = \text{SELF\_TYPE} \\
\cancel{T_i \le T_i'} \quad 1 \le i \le n \quad \Rightarrow \text{nu avem param} \\
T_{n+1} = \begin{cases} T_0 & \text{if } T_{n+1}' = \text{SELF\_TYPE} \\ T_{n+1}' & \text{otherwise} \end{cases} \Bigg\} \Rightarrow T_{n+1} = \text{Bazz}
\end{array}}{O, M, C \vdash \underset{\substack{\downarrow \\ \text{(new Bazz)}}}{e_0}.\underset{\substack{\downarrow \\ \text{foo}}}{f}(e_1, \ldots, e_n) : T_{n+1}} = \text{Bazz}$$

foo() <=> self foo() => determinat anterior (SELF_TYPE Bar)

3 Ce o să afișeze programul?



```
class Main {
    main() : Object {
        (new Bar).bar()
    };
};

class Foo inherits IO {
    foo() : SELF_TYPE {
        {
            out_string("Foo.foo()\n");
            foo();
            self;
        }
    };
};

class Bar inherits Foo {
    foo() : SELF_TYPE {
        {
            out_string("Bar.foo()\n");
            new SELF_TYPE;
        }
    };

    bar() : ~~???~~ Foo {
        case foo() of
        f : Foo => f@Foo.foo();
        b : Bar => (new Bazz).foo();
        o : Object => foo();
        esac
    };
};

class Bazz inherits Bar {
    foo() : SELF_TYPE {
        {
            out_string("Bazz.foo()\n");
            (new Bar)@Foo.foo();
            self;
        }
    };
};
```

(new Bar).bar() => apelează bar() din Bar

=> foo() din Bar => "Bar.foo()"

=> (new Bazz).foo() => "Bazz.foo()"

=> (new Bar)@Foo.foo() => "Foo.foo()"

=> foo() din self = Bar => "Bar.foo()"

OBS: În metoda foo() din Foo se apelează foo()

foo() <=> self.foo()

= Bar, deoarece s-a făcut static dispatch pe el

e@B.f() = se apelează metoda f din B pe obiectul care rezultă din evaluarea lui e. (self)

4) Contexte de tipare pt obiecte + metode în punctul ⊛

```
class A {
  i: Int;
  b: Bool;
  x: SELF_TYPE;
  foo(): SELF_TYPE { x };
}
class B inherits A {
  y: SELF_TYPE;
  g(b: Object): Object { (* EXPRESSION *) };   ⊛
}
```

$O(self) = SELF\_TYPE_B$ $M(A, foo) = SELF\_TYPE$

$O(b) = Object$ $M(B, foo) = SELF\_TYPE$

$O(y) = SELF\_TYPE_B$ $M(B, g) = (Object, Object)$

$O(x) = SELF\_TYPE_B$ + metode Object

$O(b)$ ⇒ NU!

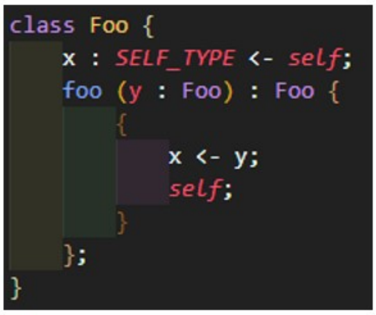
$O(i) = Int$

Q: Care este tipul static al lui EXPRESSION dacă înlocuim cu:

let x : SELF_TYPE ← x in x

$$\left.T_0' = \begin{cases} \text{SELF\_TYPE}_C & \text{if } T_0 = \text{SELF\_TYPE} \\ T_0 & \text{otherwise} \end{cases}\right\} \Rightarrow T_0' = SELF\_TYPE_B$$

$$\frac{\begin{array}{l} O, M, C \vdash e_1 : T_1 \Rightarrow T_1 = SELF\_TYPE_B \\ T_1 \leq T_0' \quad ST_B \leq ST_B \ ✓ \\ O[T_0'/x], M, C \vdash e_2 : T_2 \Rightarrow T_2 = SELF\_TYPE_B \end{array}}{O, M, \underset{B}{C} \vdash \text{let } \underset{x}{x} : \underset{ST}{T_0} \leftarrow \underset{x}{e_1} \text{ in } \underset{x}{e_2} : T_2} \Rightarrow SELF\_TYPE_B$$

5) 

```
class Foo {
    x : SELF_TYPE <- self;
    foo (y : Foo) : Foo {
        {
            x <- y;
            self;
        }
    };
}
```

Poate fi x declarat cu tipul SELF_TYPE?

x : SELF_TYPE ← self

$$\frac{\begin{array}{l} O_C(x) = T_0 \\ O_C[\text{SELF\_TYPE}_C/self], M, C \vdash e_1 : T_1 \Rightarrow T_1 = SELF\_TYPE_{Foo} \\ T_1 \leq T_0 \quad SELF\_TYPE_{Foo} \leq SELF\_TYPE_{Foo} \ ✓ \end{array}}{O_C, M, \underset{Foo}{C} \vdash \underset{x}{x} : \underset{SELF\_TYPE_{Foo}}{T_0} \leftarrow \underset{self}{e_1};}$$

x ← y

$$\frac{\begin{array}{l} O(Id) = T \Rightarrow T = SELF\_TYPE_{Foo} \\ O, M, C \vdash e_1 : T' \Rightarrow T' = Foo \\ T' \leq T \Rightarrow Foo \leq SELF\_TYPE_{Foo} \end{array}}{O, M, \underset{Foo}{C} \vdash \underset{x}{Id} \leftarrow \underset{y}{e_1} : T'}$$

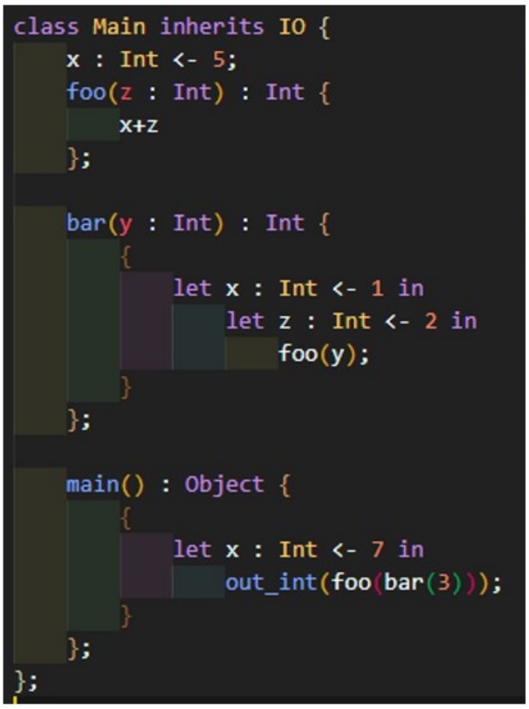
⇒ NU!! ⇒ x nu poate avea tipul SELF_TYPE

OBS $T \leq SELF\_TYPE_C$ ✗ ⇒ nu pot scrie asta

ex:
$$\begin{tikzcd} C \arrow[d] \\ T \arrow[d] \\ A \end{tikzcd}$$

(SELF_TYPE_C se referă la C + copii)

6



```
class Main inherits IO {
    x : Int <- 5;
    foo(z : Int) : Int {
        x+z
    };

    bar(y : Int) : Int {
        {
            let x : Int <- 1 in
                let z : Int <- 2 in
                    foo(y);
        }
    };

    main() : Object {
        {
            let x : Int <- 7 in
                out_int(foo(bar(3)));
        }
    };
};
```

Ce afișează programul în următoarele cazuri?

(1) statically scoped

bar(3) = foo(3) = 5+3 = 8

foo(8) = 5+8 = 13

(2) dinamically scoped

push (x=5)

push (x=7)

bar (3) => push (y=3)

push (x=1)

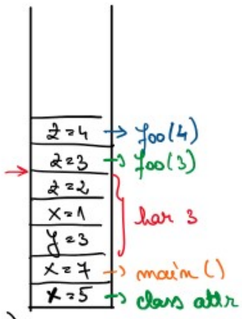
push (z=2)

foo(y) = foo(3) => push(z=3)

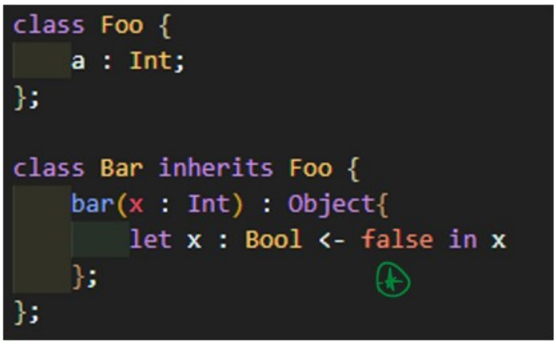
= x+z = 1+3 = 4

foo (4) => push (z=4)

= x+z = 7+4 = 11



7) Contexte de tipare în ⊛?



```
class Foo {
    a : Int;
};

class Bar inherits Foo {
    bar(x : Int) : Object{
        let x : Bool <- false in x
    };      ⊛
};
```

O(self) = SELF_TYPE$_{Bar}$

O(x) = Int

O(a) = Int

M(Bar, bar) = (Int, Object)