i219 Software Design Methodology
14. Case study 3
Mini programming language processor 2

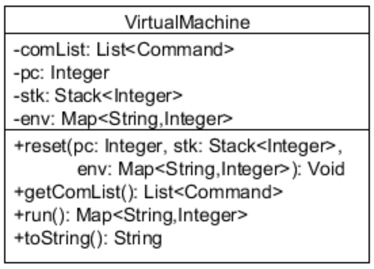
Kazuhiro Ogata (JAIST)

Outline of lecture

- Minila virtual machine
- Minila compiler

Minila virtual machine (1)

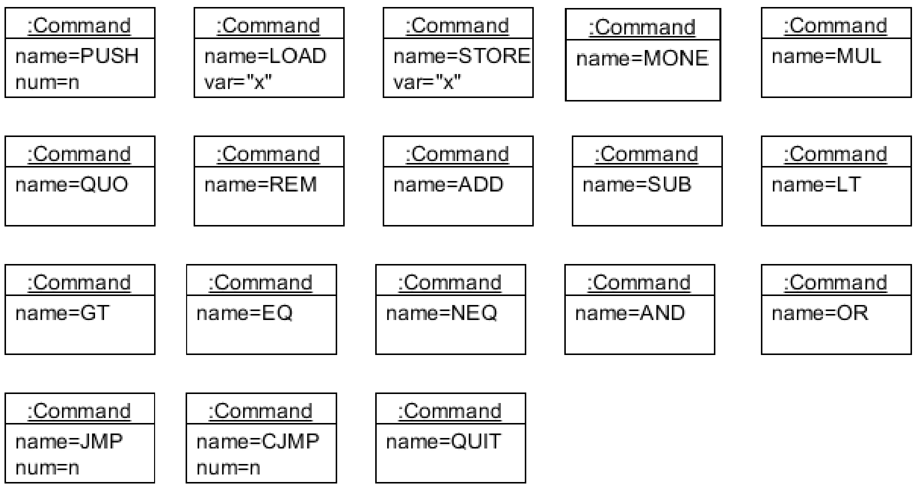
An object of VirtualMachine uses a program counter pc, a stack stk and an environment env to exeute a list comList of instructions (or commands), which is done by run.


VirtualMachine
-comList: List<Command>
-pc: Integer
-stk: Stack<Integer>
-env: Map<String,Integer>
+reset(pc: Integer, stk: Stack<Integer>, env: Map<String,Integer>): Void
+getComList(): List<Command>
+run(): Map<String,Integer>
+toString(): String

✓ reset sets this.pc, this.stk and this.env to the 1st argument pc, the 2nd argument stk and the 3rd argument env.
✓ getComList returns comList.
✓ toString returns "pc: " + pc + ", stack: " + stk + ", env: " + env + ", cl: " + comList.

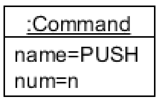
Minila virtual machine (2)

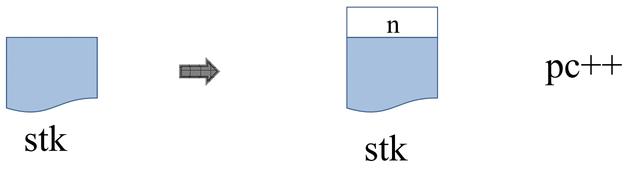
The set of instructions (or commands):



Minila virtual machine (3)

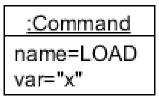
If comList.get(pc) is

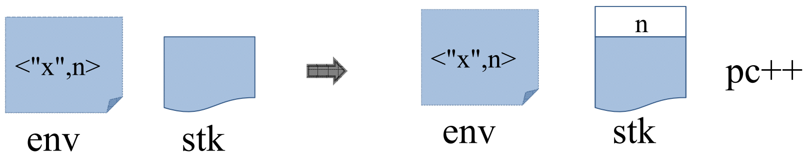




Minila virtual machine (4)

If comList.get(pc) is

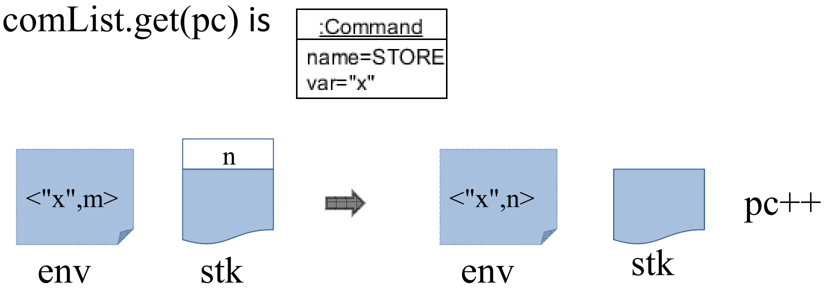




If nothings has been associated with "x" in env, null is returned as the result of env.get("x"). If so, an exception (VMException) is thrown.

Minila virtual machine (5)

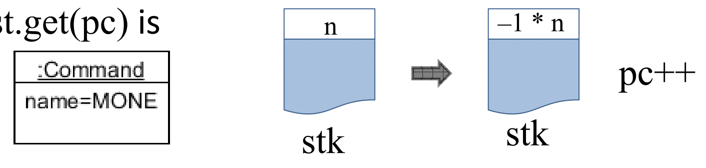
If comList.get(pc) is



If stk is empty, an exception (VMException) is thrown.

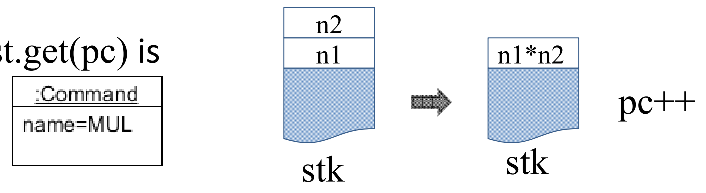
Minila virtual machine (6)

If comList.get(pc) is



If stk is empty, an exception (VMException) is thrown.

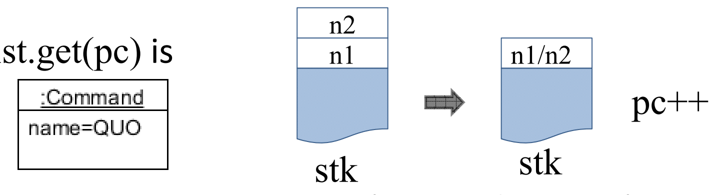
If comList.get(pc) is



If stk does not contain two or more numbers, an exception (VMException) is thrown.

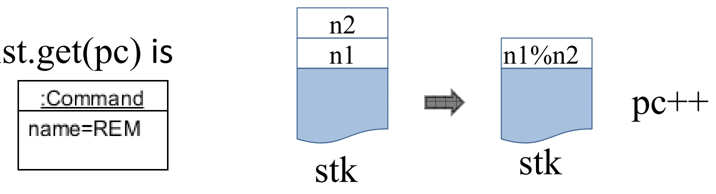
Minila virtual machine (7)

If comList.get(pc) is



If stk does not contain two or more numbers, or n2 is zero, then an exception (VMException) is thrown.

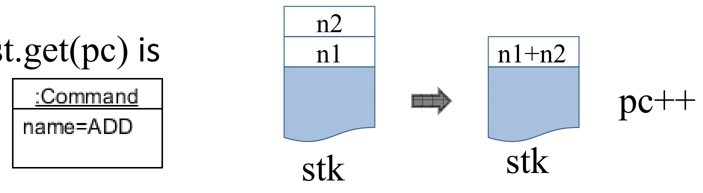
If comList.get(pc) is



If stk does not contain two or more numbers, or n2 is zero, then an exception (VMException) is thrown.

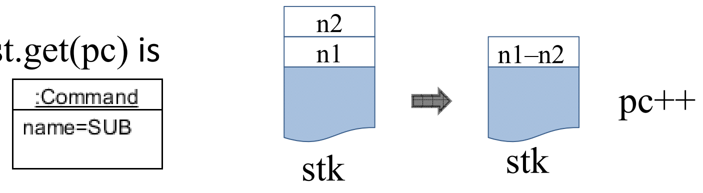
Minila virtual machine (8)

If comList.get(pc) is



If stk does not contain two or more numbers, an exception (VMException) is thrown.

If comList.get(pc) is



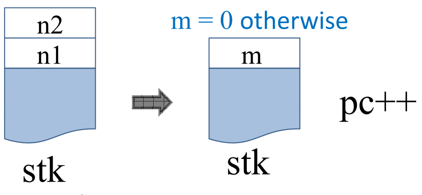
If stk does not contain two or more numbers, an exception (VMException) is thrown.

Minila virtual machine (9)

$m = 1$ if $n1 < n2$
$m = 0$ otherwise

If comList.get(pc) is

:Command
name=LT

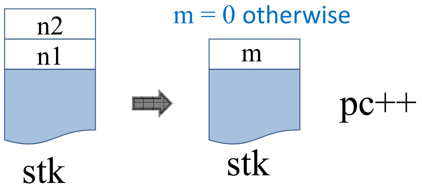


If stk does not contain two or more numbers, an exception (VMException) is thrown.

$m = 1$ if $n1 > n2$
$m = 0$ otherwise

If comList.get(pc) is

:Command
name=GT



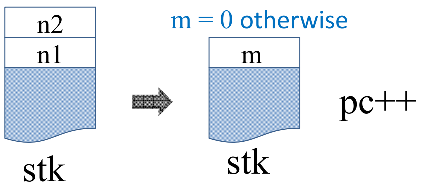
If stk does not contain two or more numbers, an exception (VMException) is thrown.

Minila virtual machine (10)

$m = 1$ if $n1 = n2$
$m = 0$ otherwise

If comList.get(pc) is

:Command
name=EQ

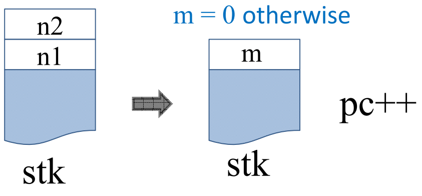


If stk does not contain two or more numbers, an exception (VMException) is thrown.

$m = 1$ if $n1 \neq n2$
$m = 0$ otherwise

If comList.get(pc) is

:Command
name=NEQ



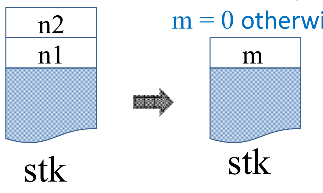
If stk does not contain two or more numbers, an exception (VMException) is thrown.

Minila virtual machine (11)

If comList.get(pc) is

:Command
name=AND

$m = 1$ if $n1 \neq 0 \wedge n2 \neq 0$
$m = 0$ otherwise



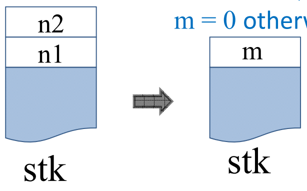
pc++

If stk does not contain two or more numbers, an exception (VMException) is thrown.

If comList.get(pc) is

:Command
name=OR

$m = 1$ if $n1 \neq 0 \vee n2 \neq 0$
$m = 0$ otherwise



pc++

If stk does not contain two or more numbers, an exception (VMException) is thrown.

Minila virtual machine (12)

If comList.get(pc) is

:Command

name=JMP
num=n

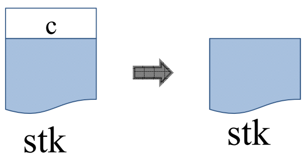
$pc \leftarrow pc+n$

If comList.get(pc) is

:Command

name=CJMP
num=n

$pc \leftarrow pc+n$ if $c \neq 0$
$pc \leftarrow pc+1$ otherwise



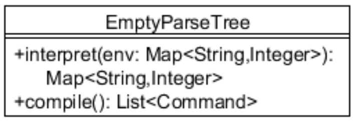
If stk is empty, an exception (VMException) is thrown.

If comList.get(pc) is

:Command
name=QUIT

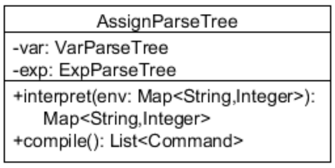
env is returned

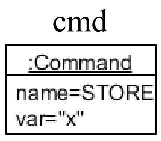
Minila compiler (1)


EmptyParseTree
+interpret(env: Map<String,Integer>): Map<String,Integer>
+compile(): List<Command>

When an object of EmptyParseTree receives compile(), it makes an object of ArrayList<Command> (the empty list of commands) and returns the object.

Minila compiler (2)

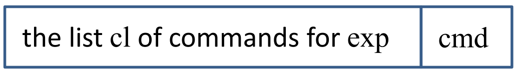

AssignParseTree
-var: VarParseTree
-exp: ExpParseTree
+interpret(env: Map<String,Integer>): Map<String,Integer>
+compile(): List<Command>



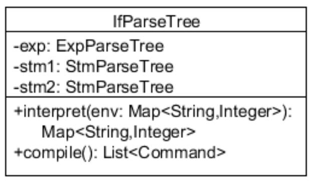
Where “x” is the name of var

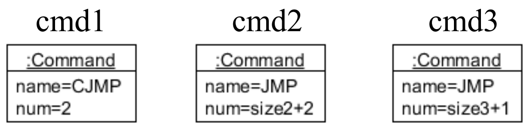
When an object of AssignParseTree receives compile(),

1. it sends compile() to exp and obtains the list cl of commands for exp,
2. adds cmd to cl at the end, and
3. returns cl.



Minila compiler (3)


IfParseTree
-exp: ExpParseTree
-stm1: StmParseTree
-stm2: StmParseTree
+interpret(env: Map<String,Integer>): Map<String,Integer>
+compile(): List<Command>

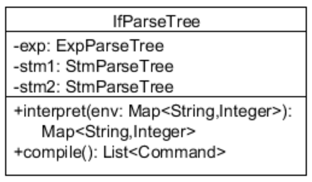


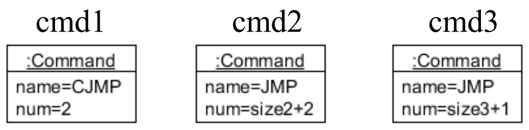
where size2 is the size of cl2 & size3 is the size of cl3

When an object of IfParseTree receives compile(),

1. it sends compile() to exp, stm1 & stm2 to obtain the lists cl1, cl2 & cl3 of commands,
2. adds cmd1 & cmd2 to cl1 at the end in this order,
3. appends cl2 to cl1 at the end,
4. adds cmd3 to cl1 at the end,
5. appends cl3 to cl1 at the end, and
6. returns cl1.

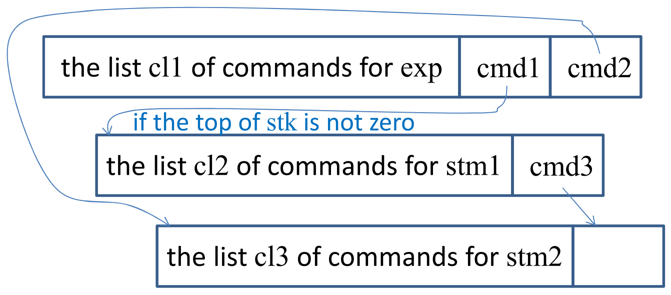
Minila compiler (4)


IfParseTree
-exp: ExpParseTree
-stm1: StmParseTree
-stm2: StmParseTree
+interpret(env: Map<String,Integer>): Map<String,Integer>
+compile(): List<Command>

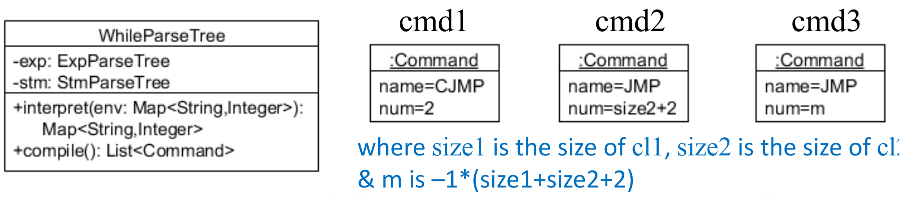


where size2 is the size of cl2 & size3 is the size of cl3

When an object of IfParseTree receives compile(),


the list cl1 of commands for exp
cmd1
cmd2
if the top of stk is not zero
the list cl2 of commands for stm1
cmd3
the list cl3 of commands for stm2

Minila compiler (5)

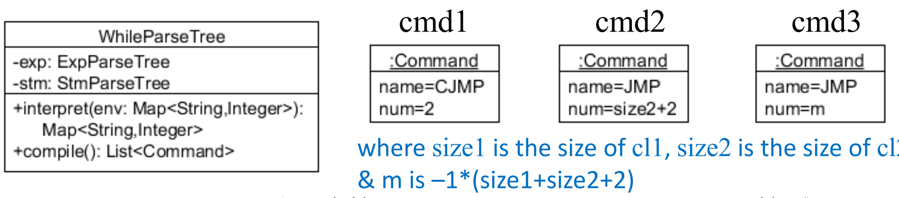


where size1 is the size of cl1, size2 is the size of cl2 & m is –1*(size1+size2+2)

When an object of WhileParseTree receives compile(),

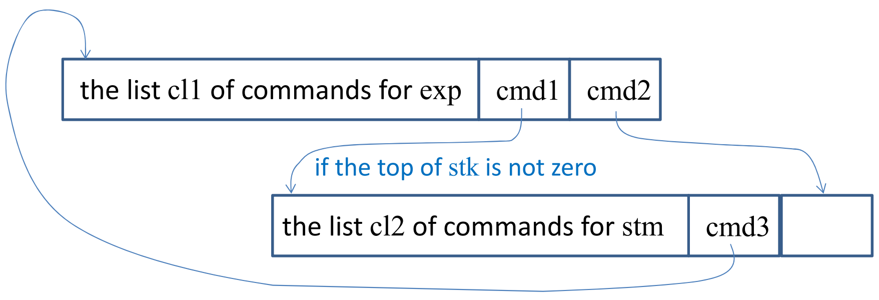
1. it sends compile() to exp & stm to obtain the lists cl1 & cl2 of commands,
2. adds cmd1 & cmd2 to cl1 at the end in this order,
3. appends cl2 to cl1 at the end,
4. adds cmd3 to cl1 at the end, and
5. returns cl1.

Minila compiler (6)

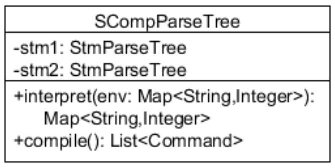


where size1 is the size of cl1, size2 is the size of cl2 & m is –1*(size1+size2+2)

When an object of WhileParseTree receives compile(),

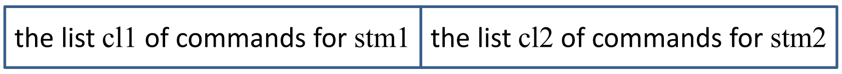

the list cl1 of commands for exp
cmd1
cmd2
if the top of stk is not zero
the list cl2 of commands for stm
cmd3

Minila compiler (7)

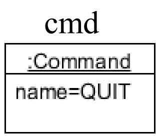

SCompParseTree
-stm1: StmParseTree
-stm2: StmParseTree
+interpret(env: Map<String,Integer>): Map<String,Integer>
+compile(): List<Command>

When an object of SCompParseTree receives compile(),

1. it sends compile() to stm1 & stm2 to obtain the lists cl1 & cl2 of commands,
2. appends cl2 to cl1 at the end, and
3. returns cl1.

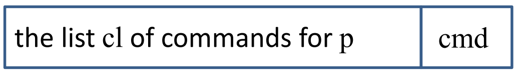

the list cl1 of commands for stm1
the list cl2 of commands for stm2

Minila compiler (8)



When a parse tree object of a Minila program p receives genCode(),

1. it sends compile() to itself to obtain the list cl of commands,
2. adds cmd to cl at the end, and
3. returns cl.



Minila compiler (9)

```
x := 1;  n := 1;
while (n = 10 || n < 10) do
  x := x*n;  n := n+1;
od
```

⇩ lexical analysis & parsing

```
scomp(assign(x,1), scomp(assign(n,1), while(or(eq(n,10),lt(n,10)),
                                      scomp(assign(x,mul(x,n)),
                                            assign(n,add(n,1))))))
```

⇩ compilation

```
[push(1), store(x), push(1), store(n),
load(n), push(10), eq, load(n), push(10), lt, or,
cjmp(2), jmp(10),
load(x), load(n), mul, store(x),
load(n), push(1), add, store(n),
jmp(-17), quit]
```

Summary

- Minila virtual machine
- Minila compiler