maTe Program Representation

CS 520
Dept. of Computer Science
Univ. of New Hampshire

maTe is a pure object-oriented language

↳ everything is an object

Java is not pure, since it mixes primitive types,
like int and Float, with objects.

Having a pure O-O language allows for a VM
with a small number of instructions.

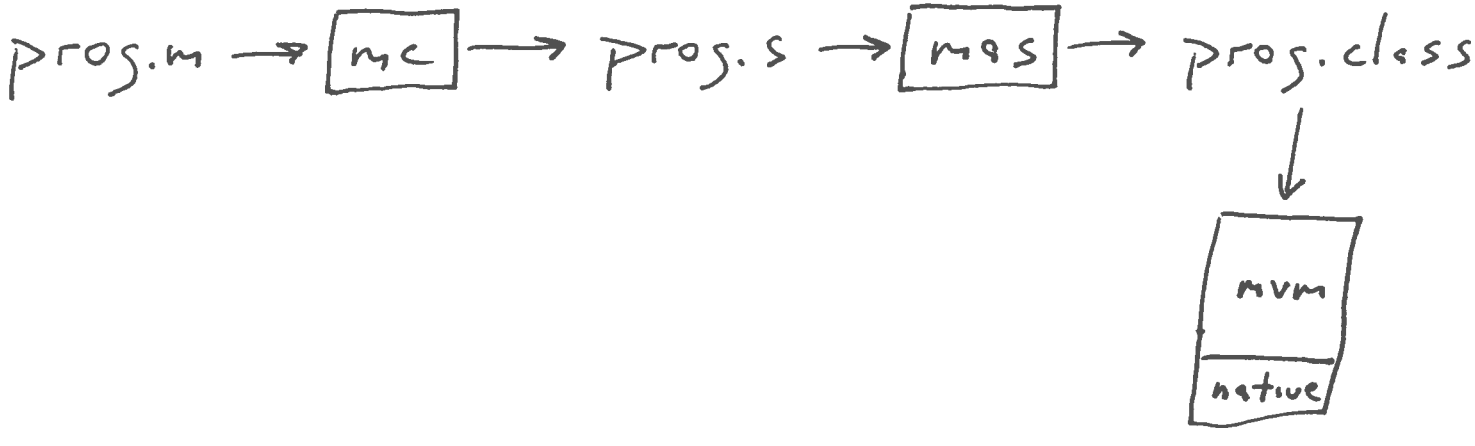
mate has four "native" classes:

- Object
- String
- Integer
- Table

the methods for these classes are implemented with native code

↳ in our case C code that is compiled to Intel

Translating and Executing meTe Programs



```
// "the meaning of life, the universe, and everything"
//
// Among many compiler writers the tradition is that the first program
// you compile with your new compiler produces a single output, 42.

Integer main()
{
  out 42;
  out newline;
}
```

```
# MainBlockNode
$mainBlock 0
4
$Object "Object"
$Integer "Integer"
$String "String"
$Table "Table"
Object:
  0
  0
  3
  0 1 "Object$equals$Object"
  0 2 "Object$hashCode"
  0 3 "Object$toString"
Integer:
  $Object
  0
  19
  0 22 "Integer$equals$Object"
  0 23 "Integer$hashCode"
  0 24 "Integer$toString"
  0 6 "Integer$add$Integer"
  0 14 "Integer$operator+$Integer"
  0 7 "Integer$subtract$Integer"
  0 15 "Integer$operator-$Integer"
  0 8 "Integer$multiply$Integer"
  0 16 "Integer$operator*$Integer"
  0 9 "Integer$divide$Integer"
  0 17 "Integer$operator/$Integer"
  0 10 "Integer$greaterThan$Integer"
  0 18 "Integer$operator>$Integer"
  0 11 "Integer$lessThan$Integer"
  0 19 "Integer$operator<$Integer"
  0 12 "Integer$not"
  0 20 "Integer$operator!"
  0 13 "Integer$minus"
  0 21 "Integer$operator-"
String:
  $Object
  0
  10
  0 34 "String$equals$Object"
  0 33 "String$hashCode"
  0 35 "String$toString"
  0 26 "String$length"
  0 27 "String$substr$Integer$Integer"
  0 29 "String$concat$String"
  0 28 "String$toInteger"
  0 30 "String$operator+$String"
  0 31 "String$operator>$String"
  0 32 "String$operator<$String"
Table:
  $Object
  0
  8
  0 1 "Object$equals$Object"
  0 2 "Object$hashCode"
  0 3 "Object$toString"
  0 38 "Table$get$Object"
  0 39 "Table$put$Object$Object"
  0 40 "Table$remove$Object"
  0 41 "Table$firstKey"
  0 42 "Table$nextKey"
mainBlock:
# BlockStatementNode
# OutStatementNode
# MethodInvocationExpressionNode
# IntegerLiteralNode
  newint 42
  invokevirtual 2 1
  out
# OutStatementNode
# StringLiteralNode
  newstr "
"
  out
  newint 0
mainBlock$.exit:
  areturn
```

class table

code

```
0000000 00 00 0e d8 00 00 00 00 00 00 00 04 00 00 00 8c
0000020 00 00 00 4f 00 00 00 62 00 00 00 6a 00 00 00 65
0000040 00 00 00 63 00 00 00 74 00 00 00 00 00 00 01 84
0000060 00 00 00 49 00 00 00 6e 00 00 00 74 00 00 00 65
0000100 00 00 00 67 00 00 00 65 00 00 00 72 00 00 00 00
0000120 00 00 08 b8 00 00 00 53 00 00 00 74 00 00 00 72
0000140 00 00 00 69 00 00 00 6e 00 00 00 67 00 00 00 00
0000160 00 00 0c 50 00 00 00 54 00 00 00 61 00 00 00 62
0000200 00 00 00 6c 00 00 00 65 00 00 00 00 00 00 00 00
0000220 00 00 00 00 00 00 00 03 00 00 00 00 00 00 00 01
0000240 00 00 00 4f 00 00 00 62 00 00 00 6a 00 00 00 65
0000260 00 00 00 63 00 00 00 74 00 00 00 24 00 00 00 65
0000300 00 00 00 71 00 00 00 75 00 00 00 61 00 00 00 6c
0000320 00 00 00 73 00 00 00 24 00 00 00 4f 00 00 00 62
0000340 00 00 00 6a 00 00 00 65 00 00 00 63 00 00 00 74
0000360 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00 4f
0000400 00 00 00 62 00 00 00 6a 00 00 00 65 00 00 00 63
0000420 00 00 00 74 00 00 00 24 00 00 00 68 00 00 00 61
0000440 00 00 00 73 00 00 00 68 00 00 00 43 00 00 00 6f
0000460 00 00 00 64 00 00 00 65 00 00 00 00 00 00 00 00
0000500 00 00 00 03 00 00 00 4f 00 00 00 62 00 00 00 6a
0000520 00 00 00 65 00 00 00 63 00 00 00 74 00 00 00 24
0000540 00 00 00 74 00 00 00 6f 00 00 00 53 00 00 00 74
0000560 00 00 00 72 00 00 00 69 00 00 00 6e 00 00 00 67
0000600 00 00 00 00 00 00 00 8c 00 00 00 00 00 00 00 13
0000620 00 00 00 00 00 00 00 16 00 00 00 49 00 00 00 6e
0000640 00 00 00 74 00 00 00 65 00 00 00 67 00 00 00 65
0000660 00 00 00 72 00 00 00 24 00 00 00 65 00 00 00 71
0000700 00 00 00 75 00 00 00 61 00 00 00 6c 00 00 00 73
0000720 00 00 00 24 00 00 00 4f 00 00 00 62 00 00 00 6a
0000740 00 00 00 65 00 00 00 63 00 00 00 74 00 00 00 00
0000760 00 00 00 00 00 00 00 17 00 00 00 49 00 00 00 6e
0001000 00 00 00 74 00 00 00 65 00 00 00 67 00 00 00 65
0001020 00 00 00 72 00 00 00 24 00 00 00 68 00 00 00 61
0001040 00 00 00 73 00 00 00 68 00 00 00 43 00 00 00 6f
0001060 00 00 00 64 00 00 00 65 00 00 00 00 00 00 00 00
0001100 00 00 00 18 00 00 00 49 00 00 00 6e 00 00 00 74
0001120 00 00 00 65 00 00 00 67 00 00 00 65 00 00 00 72
0001140 00 00 00 24 00 00 00 74 00 00 00 6f 00 00 00 53
0001160 00 00 00 74 00 00 00 72 00 00 00 69 00 00 00 6e
0001200 00 00 00 67 00 00 00 00 00 00 00 00 00 00 00 06
0001220 00 00 00 49 00 00 00 6e 00 00 00 74 00 00 00 65
0001240 00 00 00 67 00 00 00 65 00 00 00 72 00 00 00 24
0001260 00 00 00 61 00 00 00 64 00 00 00 64 00 00 00 24
0001300 00 00 00 49 00 00 00 6e 00 00 00 74 00 00 00 65
0001320 00 00 00 67 00 00 00 65 00 00 00 72 00 00 00 00
0001340 00 00 00 00 00 00 00 0e 00 00 00 49 00 00 00 6e
0001360 00 00 00 74 00 00 00 65 00 00 00 67 00 00 00 65
0001400 00 00 00 72 00 00 00 24 00 00 00 6f 00 00 00 70
0001420 00 00 00 65 00 00 00 72 00 00 00 61 00 00 00 74
0001440 00 00 00 6f 00 00 00 72 00 00 00 2b 00 00 00 24
0001460 00 00 00 49 00 00 00 6e 00 00 00 74 00 00 00 65
0001500 00 00 00 67 00 00 00 65 00 00 00 72 00 00 00 00
0001520 00 00 00 00 00 00 00 07 00 00 00 49 00 00 00 6e
0001540 00 00 00 74 00 00 00 65 00 00 00 67 00 00 00 65
0001560 00 00 00 72 00 00 00 24 00 00 00 73 00 00 00 75
0001600 00 00 00 62 00 00 00 74 00 00 00 72 00 00 00 61
0001620 00 00 00 63 00 00 00 74 00 00 00 24 00 00 00 49
0001640 00 00 00 6e 00 00 00 74 00 00 00 65 00 00 00 67
0001660 00 00 00 65 00 00 00 72 00 00 00 00 00 00 00 00
0001700 00 00 00 0f 00 00 00 49 00 00 00 6e 00 00 00 74
0001720 00 00 00 65 00 00 00 67 00 00 00 65 00 00 00 72
0001740 00 00 00 24 00 00 00 6f 00 00 00 70 00 00 00 65
0001760 00 00 00 72 00 00 00 61 00 00 00 74 00 00 00 6f
0002000 00 00 00 72 00 00 00 2d 00 00 00 24 00 00 00 49
0002020 00 00 00 6e 00 00 00 74 00 00 00 65 00 00 00 67
```

class file is a sequence of 32-bit integers but an address is a byte-offset in the class file

$ed8_{16}$ = 7330_8

```
0006140 00 00 00 01 00 00 00 4f 00 00 00 62 00 00 00 6a
0006160 00 00 00 65 00 00 00 63 00 00 00 74 00 00 00 24
0006200 00 00 00 65 00 00 00 71 00 00 00 75 00 00 00 61
0006220 00 00 00 6c 00 00 00 73 00 00 00 24 00 00 00 4f
0006240 00 00 00 62 00 00 00 6a 00 00 00 65 00 00 00 63
0006260 00 00 00 74 00 00 00 00 00 00 00 00 00 00 00 02
0006300 00 00 00 4f 00 00 00 62 00 00 00 6a 00 00 00 65
0006320 00 00 00 63 00 00 00 74 00 00 00 24 00 00 00 68
0006340 00 00 00 61 00 00 00 73 00 00 00 68 00 00 00 43
0006360 00 00 00 6f 00 00 00 64 00 00 00 65 00 00 00 00
0006400 00 00 00 00 00 00 00 03 00 00 00 4f 00 00 00 62
0006420 00 00 00 6a 00 00 00 65 00 00 00 63 00 00 00 74
0006440 00 00 00 24 00 00 00 74 00 00 00 6f 00 00 00 53
0006460 00 00 00 74 00 00 00 72 00 00 00 69 00 00 00 6e
0006500 00 00 00 67 00 00 00 00 00 00 00 00 00 00 00 26
0006520 00 00 00 54 00 00 00 61 00 00 00 62 00 00 00 6c
0006540 00 00 00 65 00 00 00 24 00 00 00 67 00 00 00 65
0006560 00 00 00 74 00 00 00 24 00 00 00 4f 00 00 00 62
0006600 00 00 00 6a 00 00 00 65 00 00 00 63 00 00 00 74
0006620 00 00 00 00 00 00 00 00 00 00 00 00 27 00 00 00 54
0006640 00 00 00 61 00 00 00 62 00 00 00 6c 00 00 00 65
0006660 00 00 00 24 00 00 00 70 00 00 00 75 00 00 00 74
0006700 00 00 00 24 00 00 00 4f 00 00 00 62 00 00 00 6a
0006720 00 00 00 65 00 00 00 63 00 00 00 74 00 00 00 24
0006740 00 00 00 4f 00 00 00 62 00 00 00 6a 00 00 00 65
0006760 00 00 00 63 00 00 00 74 00 00 00 00 00 00 00 00
0007000 00 00 00 28 00 00 00 54 00 00 00 61 00 00 00 62
0007020 00 00 00 6c 00 00 00 65 00 00 00 24 00 00 00 72
0007040 00 00 00 65 00 00 00 6d 00 00 00 6f 00 00 00 76
0007060 00 00 00 65 00 00 00 24 00 00 00 4f 00 00 00 62
0007100 00 00 00 6a 00 00 00 65 00 00 00 63 00 00 00 74
0007120 00 00 00 00 00 00 00 00 00 00 00 00 29 00 00 00 54
0007140 00 00 00 61 00 00 00 62 00 00 00 6c 00 00 00 65
0007160 00 00 00 24 00 00 00 66 00 00 00 69 00 00 00 72
0007200 00 00 00 73 00 00 00 74 00 00 00 4b 00 00 00 65
0007220 00 00 00 79 00 00 00 00 00 00 00 00 00 00 00 2a
0007240 00 00 00 54 00 00 00 61 00 00 00 62 00 00 00 6c
0007260 00 00 00 65 00 00 00 24 00 00 00 6e 00 00 00 65
0007300 00 00 00 78 00 00 00 74 00 00 00 4b 00 00 00 65
0007320 00 00 00 79 00 00 00 00 00 00 00 f7 00 00 00 2a
0007340 00 00 00 b6 00 00 00 02 00 00 00 01 00 00 00 f5
0007360 00 00 00 f8 00 00 00 0a 00 00 00 00 00 00 00 f5
0007400 00 00 00 f7 00 00 00 00 00 00 00 b0
0007414
```

new int 42 ~~f0~~ 000000f7 00000029

invokevirtual 2 1

maTe Virtual Machine Instructions

---

aconst_null

Operation

Push null reference.

Format

aconst_null

Forms

aconst_null = 1 (0x00000001)

Operand Stack

... ⇒ ..., *null*

Description

The value denoting a null reference is pushed onto the operand stack.

---

aload

Operation

Load reference from local variable.

Format

aload
index

Forms

aload = 25 (0x00000019)

Operand Stack

... ⇒ ..., *objectref*

Description

Format

new
class

Forms

new = 187 (0x000000bb)

Operand Stack

... ⇒ ..., *objectref*

Description

class must denote a class described in the class table and indicates the class of the object to be created. Memory for a new instance of that class is allocated, instance variables (if any) of the new object are initialized to their default initial values, and the *objectref* for the new object is pushed onto the operand stack.

---

newint

Operation

Create new Integer object.

Format

newint
value

Forms

newint = 247 (0x000000f7)

Operand Stack

... ⇒ ..., *objectref*

Description

value is the value of the new Integer object to be created. Memory for a new instance of Integer is allocated, the value is assigned, and the *objectref* for the new object is pushed onto the operand stack.

---

including the *objectref*. *locals* denotes the number of locals to be used by the method. *objectref* must not be *null*. The *n* argument values, including *objectref*, are popped from the operand stack. A new frame is created for the method being invoked. The *objectref* and the argument values are consecutively made the values of local variables of the new frame, with *objectref* in local variable 0, *arg1* in local variable 1, and so on. The new frame is then made current and execution continues with the first instruction of the method.

---

invokevirtual

Operation

Invoke method with the dispatch based on class.

Format

invokevirtual
index
n

Forms

invokevirtual = 182 (0x000000b6)

Operand Stack

..., *objectref*, [*arg1* [, *arg2* ... [, *argn-1*]]] ⇒ ...

Description

objectref must not be *null*. *index* must be an index into the method table of the class of the object denoted by *objectref*. The *n* argument values, including the *objectref*, are popped from the operand stack. A new frame is created for the method being invoked. The *objectref* and the argument values are consecutively made the values of local variables of the new frame, with *objectref* in local variable 0, *arg1* in local variable 1, and so on. The new frame is then made current and execution continues with the first instruction of the method selected by *index*. However, if the address retrived from the method table is zero, then a native method is invoked as in *invokenative*.

---

new

Operation

Create new object.