Java compiler implementation in Python: Examples

Chris Lamb

March 2007

1. Compiling Java
    - Peephole optimiser
    - Register colouring
2. Detecting errors
    - Type checking
    - Linting
3. Other languages
    - Lisp
    - BF
    - Abusing the `--lang` switch

Compiling Java - Input

```
public class Example1 {
    public static void main(String args[]) {
        int i = 6;
        i = i + 10;
        i++;

        if (i > 15) {
            System.out.println("i is greater than ten");
        }
    }
}
```

Pre-optimisation output (bytecode)

```
ldc             {value:6}
istore          {localvar:1}
iload           {localvar:1}
ldc             {value:10}
iadd
istore          {localvar:1}
iinc            {localvar:1, value:1}
iload           {localvar:1}
ldc             {value:15}
if_icmpgt       {target:'true-1480989268'}
iconst_0
goto            {target:'tail-1480989268'}
iconst_1        {labels:['true-1480989268']}
nop             {labels:['tail-1480989268']}
ifeq            {target:'tail-1481130004'}
getstatic       {returntype:'Ljava/io/PrintStream;', field:'out',
                     klass:'java/lang/System'}
ldc             {value:'i is greater than ten'}
invokevirtual   {klass:'java/io/PrintStream', args:'(Ljava/lang/String;)V',
                     name:'println'}
nop             {labels:['tail-1481130004']}
return_
```

Post-optimisation output (bytecode)

```
bipush          {value:6}
istore_1
iinc            {localvar:1, value:11}
iload_1
bipush          {value:15}
if_icmple       {target:'tail-1481539092'}
getstatic       {returntype:'Ljava/io/PrintStream;', field:'out',
                      labels:['true-1481521748'], klass:'java/lang/System'}
ldc             {value:'i is greater than ten'}
invokevirtual   {klass:'java/io/PrintStream', args:'(Ljava/lang/String;)V',
                      name:'println'}
return_         {labels:['tail-1481539092']}
```

- 50% of pre-optimised size
- Assembler generates `.class` representation of the code

Comparison with Sun's javac

```
0     bipush  6
2     istore_1
3     iload_1
4     bipush  10
6     iadd
7     istore_1
8     iinc index: 1 const 1
11    iload_1
12    bipush  15
14    if_icmple 25
17    getstatic #2 <Field java/lang/System.out Ljava/io/PrintStream;>
20    ldc #3 <String "i is greater than ten">
22    invokevirtual #4 <Method java/io/PrintStream.println (Ljava/lang/String;)V>
25    return
```

- Sun: 14 instructions, Lamby: 10 instructions
- Simple rules to transform code

Register colouring

```
public class Example2 {
    public static void main(String args[]) {
        int a = 0;
        int b = 0;

        a = 1;
        b = 1;

        int c = a + 1;
    }
}
```

Register colouring

```
public class Example2 {
    public static void main(String args[]) {
        int a = 0;
        int b = 0;

        a = 1;
        b = 1;

        int c = a + 1;
    }
}
```

- Variables b and c and are never live at the same time → can share register
- Sun's `javac` uses 3 registers, I use only 2

Detecting errors - input

```
public class Example3 {
    public static void main(String args[]) {
        int i = false + 2 + 3;
        byte b = (int) 12;
        if ("String") { }
        boolean bool;
        bool++;
        int j;
        System.out.println(j);
    }
}
```

Detecting errors - output

```
Example3.java:3:16: operator + cannot be applied to boolean,int
        int i = false + 2 + 3;
                ^

Example3.java:4:8: possible loss of precision
        byte b = (int) 12;
        ^

Example3.java:5:12: incompatible types
found   : java.lang.String
required: boolean
        if ("String") { }
            ^

Example3.java:7:8: operator ++ cannot be applied to boolean
        bool++;
        ^

Example3.java:9:27: variable i might not have been initialized
        System.out.println(j);
                           ^
```

Linting

Detects semantic errors in syntacticly valid code.

```
public class Example4
{
    public static void main(String args[])
    {
        for (int i = 0; i > 10; i++)
        {
            System.out.println(i);
        }
    }
}
```

Linting

Detects semantic errors in syntacticly valid code.

```
public class Example4
{
    public static void main(String args[])
    {
        for (int i = 0; i > 10; i++)
        {
            System.out.println(i);
        }
    }
}
```

Outputs (as a compiler warning):

```
warning:Example4.java:6:24: possible wrong comparison used
        for (int i = 0; i > 10; i++)
                          ^
```

Compiling Lisp

```
(defun factorial (n)
    (if (<= n 1)
        1
        (* n (factorial (- n 1)))))

(print (factorial 3))
```

- **defun** definitions ⇔ **public static** methods
- Type checker ensures correct scoping

Compiling Lisp - bytecode output

```
Method public static  factorial (int) -> int
0      iload_0
1      iconst_1
2      if_icmpgt 9
5      iconst_1
6      goto 17
9      iload_0
10     dup
11     iconst_1
12     isub
13     invokestatic #6 <Method Factorial.factorial (I)I>
16     imul
17     ireturn


Method public static  main (java/lang/String []) -> void
0      iconst_3
1      invokestatic #6 <Method Factorial.factorial (I)I>
4      dup
5      getstatic #14 <Field java/lang/System.out Ljava/io/PrintStream;>
8      swap
9      invokevirtual #20 <Method java/io/PrintStream.println (I)V>
12     return
```

Compiling BF

```
 ++++++++++[>+++++++>++++++++++++
>+++>+<<<<-]>++++.>---.--.+.>++.<
<---------.>------.+.+++++.------
------.---.+++++++++++++.>.<<+.>.
-----------------.++++++++.+++++.
--------.+ +     +++
++++++++++.     --
----------------.++
++++++.>.<<+
+++++.>----
------.++.+
+++++++.---
---.+++++++
++++++.[-]+
+++++++++.
```

b a n g !

Compiling BF

```
 ++++++++++[>+++++++>++++++++++++
>+++>+<<<<-]>++++.>---.--.+.>++.<
<---------.>------.+.+++++.------
------.---.+++++++++++++.>.<<+.>.
-----------------.++++++++.+++++.
--------.+ +     +++
++++++++++.     --
----------------.++
++++++.>.<<+
+++++.>----
------.++.+
+++++++.---
---.+++++++
++++++.[-]+
+++++++++.
```

b a n g !

Outputs: “Just Another Brainf*** Hacker”.

Abusing the `--lang` switch

```
public class Multi{public static void main(String args[//-
    ]) {
        // ++++++++++[>+++++++>+++
        // +++++++>+++>+<<<<-]>++
        System. // >+
        out. // +++++++.
        println("Hello World!"); // .+++.>++.<<+++
        // ++++++++++++.>.+++.------.--------.>+.>.
    }}
```

Abusing the `--lang` switch

```
public class Multi{public static void main(String args[//-
    ]) {
        // ++++++++++[>+++++++>+++
        // +++++++>+++>+<<<<-]>++
        System. // >+
        out. // +++++++.
        println("Hello World!"); // .+++.>++.<<+++
        // ++++++++++++.>.+++.------.--------.>+.>.
    }}
```

- Outputs “Hello World!” when called with `--lang=bf` or `--lang=Java` :)