Comparison of Code Execution between Eolang and Java

1. This program sums array elements and outputs to the console (Test cases attached)

Java

```
class Main {
    public static void main(String[] args) {
        int numbers[] = {1, 2, 3, 4…}; // Array of 2700 elements
        System.out.println(getSum(numbers));
    }

    public static int getSum(int[] elems){
        int sum = 0;
        int i=0;
          while(i<elems.length){
              sum += elems[i];
              i++;
          }
          return sum;
      }

  }
```

Eolang (while loop)

```
+package sandbox
+alias stdout org.eolang.io.stdout
+alias sprintf org.eolang.txt.sprintf

[args...] > app
  memory > sum
  memory > count
  seq > @
    count.write 0
      sum.write 0
      while.
        count.less ($.args.length)
        [i]
          seq > @
            ^.sum.write (^.sum.add (^.args.get i).toInt)
            ^.count.write (^.count.add 1)
      stdout
        sprintf "Sum is %d\n" sum
```

	Java	**Eolang**	
Input	Array of 2700 elements	Array of 2700 elements	
Execution Time	0.2086576s	3.2857814s	
Output	136350	136350	

Time in seconds

Eolang runs much slower than Java based on the above results.

This is the generated source code by the current Eolang compiler for the above Eo code:

```
 package sandbox;

import org.eolang.*;
import org.eolang.phi.*;

public final class EOapp extends PhDefault {
  public EOapp() {
    this(new PhEta());
  }
  public EOapp(final Phi parent) {
    super(parent);
    this.add("args", new AtVararg(/* default */));
    this.add("sum", new AtBound(new AtOnce(new AtLambda(this, self -> {
      Phi ret = new org.eolang.EOmemory(self);
      return ret;
    }))));
    this.add("count", new AtBound(new AtOnce(new AtLambda(this, self -> {
      Phi ret = new org.eolang.EOmemory(self);
      return ret;
    }))));
    this.add("φ", new AtBound(new AtOnce(new AtLambda(this, self -> {
      Phi ret = new org.eolang.EOseq(self);
      ret = new PhCopy(ret);
        Phi ret_1_base = new PhMethod(self, "count");
        Phi ret_1 = new PhMethod(ret_1_base, "write"); // why instantiate
        ret_1 = new PhCopy(ret_1); // make a copy right after
          Phi ret_1_1 = new org.eolang.EOint(self);
          ret_1_1 = new PhCopy(ret_1_1);
            ret_1_1 = new PhWith(ret_1_1, "Δ", new Data.Value<Long>(0L));
          ret_1 = new PhWith(ret_1, 0, ret_1_1);
        Phi ret_2_base = new PhMethod(self, "sum");
        Phi ret_2 = new PhMethod(ret_2_base, "write"); //why instantiate
        ret_2 = new PhCopy(ret_2); // and make a copy right after
          Phi ret_2_1 = new org.eolang.EOint(self);
          ret_2_1 = new PhCopy(ret_2_1);
            ret_2_1 = new PhWith(ret_2_1, "Δ", new Data.Value<Long>(0L));
          ret_2 = new PhWith(ret_2, 0, ret_2_1);
        Phi ret_3_base_base = new PhMethod(self, "count");
        Phi ret_3_base = new PhMethod(ret_3_base_base, "less"); // why instantiate
        ret_3_base = new PhCopy(ret_3_base); // and make a copy after
          Phi ret_3_base_1_base_base = self;
          Phi ret_3_base_1_base = new PhMethod(ret_3_base_1_base_base, "args");
          Phi ret_3_base_1 = new PhMethod(ret_3_base_1_base, "length");
          ret_3_base = new PhWith(ret_3_base, 0, ret_3_base_1);
        Phi ret_3 = new PhMethod(ret_3_base, "while");
        ret_3 = new PhCopy(ret_3); // Redundant
          Phi ret_3_1 = new EOapp$EO3$EO2$EOα1(self);
          ret_3_1 = new PhCopy(ret_3_1); // Redundant
          ret_3 = new PhWith(ret_3, 0, ret_3_1);
        Phi ret_4 = new org.eolang.io.EOstdout(self);
        ret_4 = new PhCopy(ret_4); // Redundant
          Phi ret_4_1 = new org.eolang.txt.EOsprintf(self);
          ret_4_1 = new PhCopy(ret_4_1); // Redundant
            Phi ret_4_1_1 = new org.eolang.EOstring(self);
            ret_4_1_1 = new PhCopy(ret_4_1_1); // Redundant
              ret_4_1_1 = new PhWith(ret_4_1_1, "Δ", new Data.Value<String>("Sum is
%d\n"));
            Phi ret_4_1_2 = new PhMethod(self, "sum");
            ret_4_1 = new PhWith(ret_4_1, 0, ret_4_1_1);
            ret_4_1 = new PhWith(ret_4_1, 1, ret_4_1_2);
          ret_4 = new PhWith(ret_4, 0, ret_4_1);
        ret = new PhWith(ret, 0, ret_1); // redundant
        ret = new PhWith(ret, 1, ret_2); // redundant
        ret = new PhWith(ret, 2, ret_3); // redundant
        ret = new PhWith(ret, 3, ret_4); // only one is relevant
      return ret;
    }))));
  }
}
```

There are several lines with “new PhCopy(***)” which logically seem redundant.

Also the “new PhWith(***)” has been redundantly used in the last few lines.

After removing the code that generates the “new PhCopy(***)” lines from the compiler, the behavior of the program after recompiling does not negatively change.

The same test case was applied to this newly generated code (without the “new PhCopy(***)”) and the new execution time was recorded as 2.8761593s, which is better than the previous.

Eolang (Map reduce function)

```
+package sandbox
+alias stdout org.eolang.io.stdout
+alias sprintf org.eolang.txt.sprintf

[args...] > app
  [accumulator current] > getSum
    add. > @
      accumulator
      current.toInt

   reduce. > sum
     args
     0
     getSum

   stdout > @
     sprintf
       "Sum is %d\n"
       sum
```

	Java	**Eolang (Map Reduce Function)**	
	Array of 600 elements	Array of 600 elements	
Execution Time	0.2303133	8.3125863	
Output	30300	30300	
Input	Array of 650 elements	Array of 650 elements	
Execution Time	0.3013884	1.0132759	
Output	31575	StackOverFlowError	

The map reduce object in Eolang could not handle larger test cases (where number of inputs is 650 and above).

2. This program prints some random numbers (float)

```
+package sandbox
+alias stdout org.eolang.io.stdout
+alias sprintf org.eolang.txt.sprintf

[args...] > app
  stdout > @
    sprintf
      "Random between 0 and 5: %f\nNegative: %f\nPower: %f\nRandom mul: %f\nRandom div: %f\nRandom equality: %b\nRandom <: %b\n"
        (random.mul 5.0).add 1.0
        0.23.neg
        random.pow 9.0
        (100.toFloat).mul random
        10.0.div random
        random.eq random
        1.0.less random
```

The Eolang random object works much like Math.random in Java. Except that Eo random objects cannot be used with integers at all.

The same issues described above exists in the generated source for this code too:

```
package sandbox;

import org.eolang.*;
import org.eolang.phi.*;

public final class EOapp extends PhDefault {
  public EOapp() {
    this(new PhEta());
  }
  public EOapp(final Phi parent) {
    super(parent);
    this.add("args", new AtVararg(/* default */));
    this.add("φ", new AtBound(new AtOnce(new AtLambda(this, self -> {
      Phi ret = new org.eolang.io.EOstdout(self);
      ret = new PhCopy(ret);
        Phi ret_1 = new org.eolang.txt.EOsprintf(self);
        ret_1 = new PhCopy(ret_1);
          Phi ret_1_1 = new org.eolang.EOstring(self);
          ret_1_1 = new PhCopy(ret_1_1);
            ret_1_1 = new PhWith(ret_1_1, "Δ", new Data.Value<String>("Random number between 1 and 5: %f\nNegative: %f\nPower: %f\nRandom mul: %f\nRandom div: %f\nRandom equality: %b\nRandom <: %b\n"));
          Phi ret_1_2_base_base = new org.eolang.EOrandom(self);
          Phi ret_1_2_base = new PhMethod(ret_1_2_base_base, "mul");
          ret_1_2_base = new PhCopy(ret_1_2_base);
            Phi ret_1_2_base_1 = new org.eolang.EOfloat(self);
            ret_1_2_base_1 = new PhCopy(ret_1_2_base_1);
              ret_1_2_base_1 = new PhWith(ret_1_2_base_1, "Δ", new Data.Value<Double>(5.0d));
            ret_1_2_base = new PhWith(ret_1_2_base, 0, ret_1_2_base_1);
          Phi ret_1_2 = new PhMethod(ret_1_2_base, "add");
          ret_1_2 = new PhCopy(ret_1_2);
            Phi ret_1_2_1 = new org.eolang.EOfloat(self);
            ret_1_2_1 = new PhCopy(ret_1_2_1);
              ret_1_2_1 = new PhWith(ret_1_2_1, "Δ", new Data.Value<Double>(1.0d));
            ret_1_2 = new PhWith(ret_1_2, 0, ret_1_2_1);
          Phi ret_1_3_base = new org.eolang.EOfloat(self);
          ret_1_3_base = new PhCopy(ret_1_3_base);
            ret_1_3_base = new PhWith(ret_1_3_base, "Δ", new Data.Value<Double>(0.23d));
          Phi ret_1_3 = new PhMethod(ret_1_3_base, "neg");
          Phi ret_1_4_base = new org.eolang.EOrandom(self);
          Phi ret_1_4 = new PhMethod(ret_1_4_base, "pow");
          ret_1_4 = new PhCopy(ret_1_4);
            Phi ret_1_4_1 = new org.eolang.EOfloat(self);
            ret_1_4_1 = new PhCopy(ret_1_4_1);
              ret_1_4_1 = new PhWith(ret_1_4_1, "Δ", new Data.Value<Double>(9.0d));
            ret_1_4 = new PhWith(ret_1_4, 0, ret_1_4_1);
          Phi ret_1_5_base_base = new org.eolang.EOint(self);
          ret_1_5_base_base = new PhCopy(ret_1_5_base_base);
            ret_1_5_base_base = new PhWith(ret_1_5_base_base, "Δ", new Data.Value<Long>(100L));
          Phi ret_1_5_base = new PhMethod(ret_1_5_base_base, "toFloat");
          Phi ret_1_5 = new PhMethod(ret_1_5_base, "mul");
          ret_1_5 = new PhCopy(ret_1_5);
            Phi ret_1_5_1 = new org.eolang.EOrandom(self);
            ret_1_5 = new PhWith(ret_1_5, 0, ret_1_5_1);
          Phi ret_1_6_base = new org.eolang.EOfloat(self);
          ret_1_6_base = new PhCopy(ret_1_6_base);
            ret_1_6_base = new PhWith(ret_1_6_base, "Δ", new Data.Value<Double>(10.0d));
          Phi ret_1_6 = new PhMethod(ret_1_6_base, "div");
          ret_1_6 = new PhCopy(ret_1_6);
            Phi ret_1_6_1 = new org.eolang.EOrandom(self);
            ret_1_6 = new PhWith(ret_1_6, 0, ret_1_6_1);
          Phi ret_1_7_base = new org.eolang.EOrandom(self);
          Phi ret_1_7 = new PhMethod(ret_1_7_base, "eq");
          ret_1_7 = new PhCopy(ret_1_7);
            Phi ret_1_7_1 = new org.eolang.EOrandom(self);
            ret_1_7 = new PhWith(ret_1_7, 0, ret_1_7_1);
          Phi ret_1_8_base = new org.eolang.EOfloat(self);
          ret_1_8_base = new PhCopy(ret_1_8_base);
            ret_1_8_base = new PhWith(ret_1_8_base, "Δ", new Data.Value<Double>(1.0d));
          Phi ret_1_8 = new PhMethod(ret_1_8_base, "less");
          ret_1_8 = new PhCopy(ret_1_8);
            Phi ret_1_8_1 = new org.eolang.EOrandom(self);
            ret_1_8 = new PhWith(ret_1_8, 0, ret_1_8_1);
          ret_1 = new PhWith(ret_1, 0, ret_1_1);
          ret_1 = new PhWith(ret_1, 1, ret_1_2);
          ret_1 = new PhWith(ret_1, 2, ret_1_3);
          ret_1 = new PhWith(ret_1, 3, ret_1_4);
          ret_1 = new PhWith(ret_1, 4, ret_1_5);
          ret_1 = new PhWith(ret_1, 5, ret_1_6);
          ret_1 = new PhWith(ret_1, 6, ret_1_7);
          ret_1 = new PhWith(ret_1, 7, ret_1_8);
        ret = new PhWith(ret, 0, ret_1);
      return ret;
    })))));
  }
}
```

Many lines of Unnecessary “new PhCopy(***)” and “new PhWith(***)”. It is not clear why these redundancies should exist. It seems alright to remove them from the compiler.