CS179E: *Phase Two*

Contributors: David H. and Jesus S.

Requirements and Specifications

Specifications:

For this phase in the project, we were required to create a vapor code generator, which takes a correctly type checked MiniJava program (done in phase 1) and converts it's source code into Vapor code. The two programs should behave the same and the vapor program should also check if an array being accessed has gone out of bounds.

Requirements:

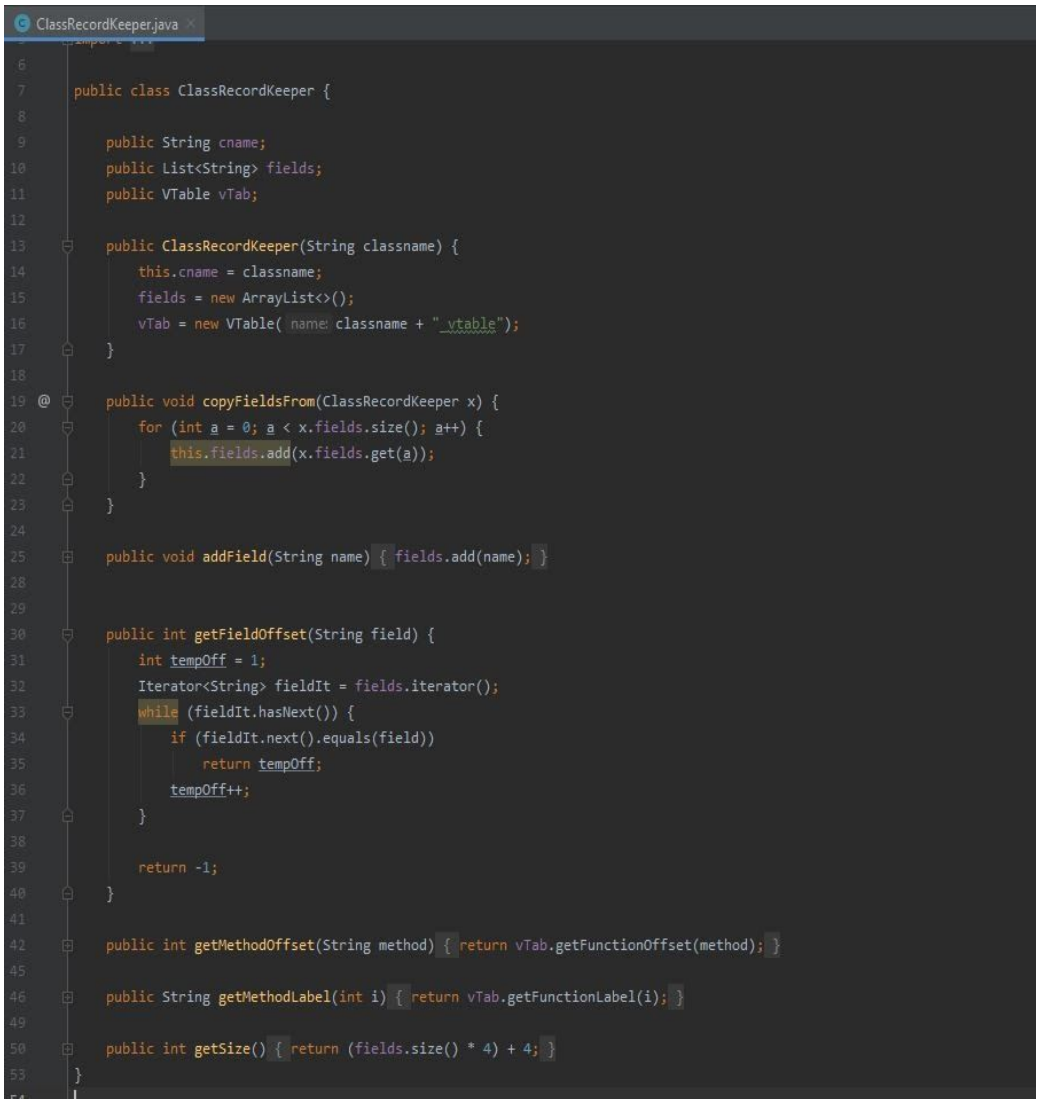
- Software:
    - JavaCC
    - Intellij (or any IDE)
    - Java Tree Builder
    - Vapor
    - Phase Two Test Cases

Design

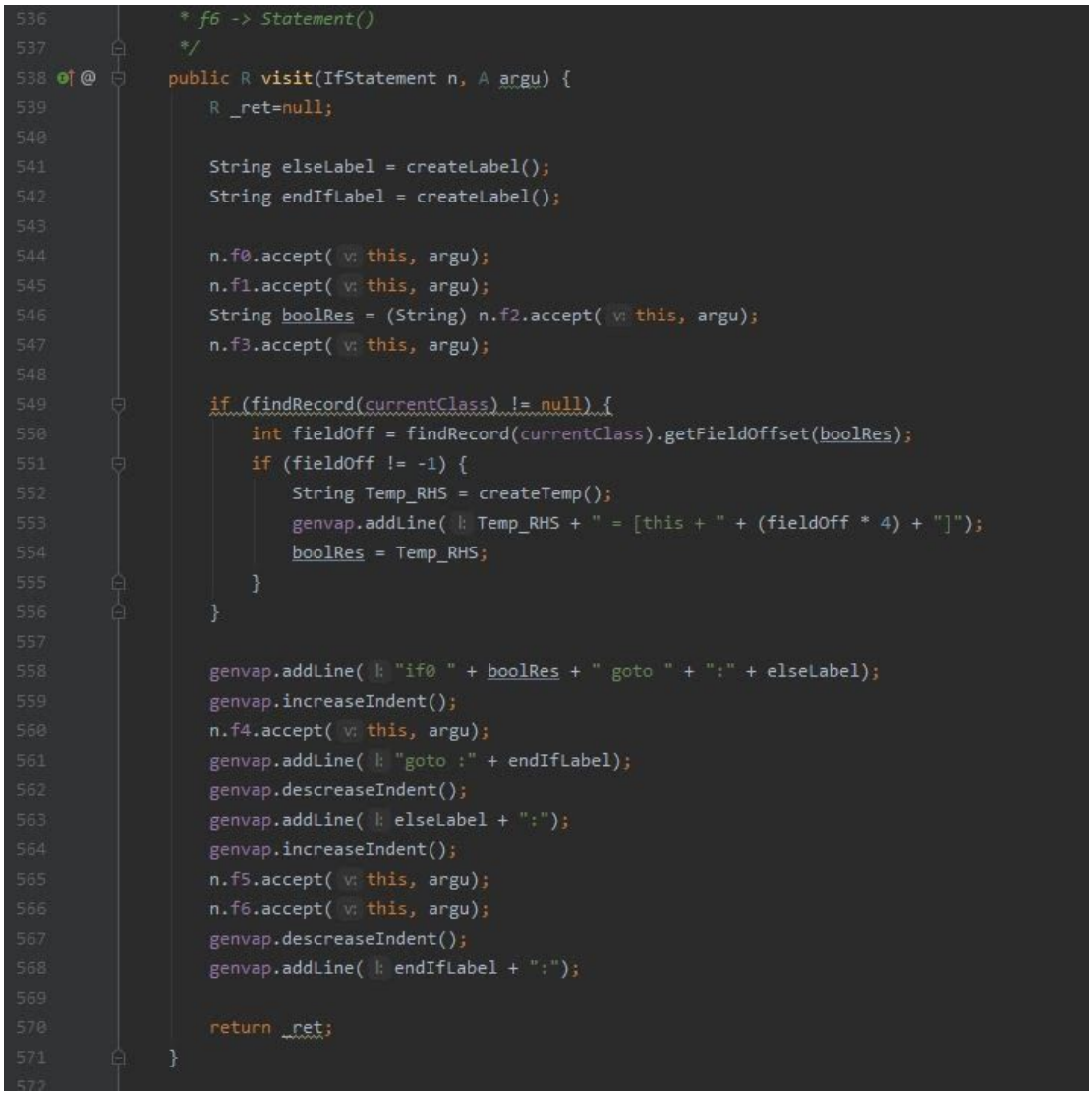
For the design of phase 2, a few new classes were added to implement the conversion from the miniJava input file to a vapor output program. The main classes that were added are called ClassRecordKeeper, VaporGenVisitor, and VTable. However the symbol table from phase 1 is still needed to collect the required information. After the symbol table is completed, a visitor is used to collect the data to fill in the vTable. Next after completing both tables another visitor will have enough information when going through the syntax tree to correctly convert the programs language with the help of the information that's stored in the vTable and symbol table. The creation of the symbol table that I'm using for this phase is explained in my phase 1 documentation, but a vTable serves a different purpose. A vTable stores all of the functions which are members of the class. Also each class contained in the input program will get its own vTable and every time a class instance is made, the vTable is copied to the new class instance. Now if a class extends another class (which is fine in java), then the extending class gets a copy of the information in the parent's vTable. The vTable in vapor is the data structure const since it will be available in the main program, and all of the extended classes will also be const and copy some information from other vTables. The offsets you'll see under testing/verification are calculated for each function label in the program, and done by using my ClassRecordKeeper class. Now in comparison the ClassRecordKeeper is a smaller class then VaporGenVisitor and VTable but is responsible for giving the visitor important information like offsets for function labels/fields and size of the class structure. Size of the class is found by multiplying the number of fields in a class by four, then adding four to the whole sum.

Code Snippets & Explanation

In this section I briefly wanted to show examples of the three main classes that I used to implement phase 2 as described above. The first example I'll show how exactly calculating the offsets/size was implemented in methods such as getFieldOffset, and getSize. Also the data fields in ClassRecordKeeper are used to assist with other functions which help the visitor get the required information for the conversion to vapor. For clarification the variable "cname" is short for class name and "vTab" is short for vTable, while fields is named fine but represented using a list of strings.



In the second example I'll show how VaporGenVistior implements GJVisitor and does the work of traversing the syntax tree and generating the correct vapor code based on the type of expression being visited. Below you'll see how I generate the vapor code needed for an If Statement. The first thing I did was create two labels for the else and endif locations and then create a string called "boolRes" to hold the conditional part of the If Statement. Next I need to check to see if the current class record is null because if it is then the field offset will have to be recalculated. After that check is completed, my "genvap" object is called to add the correct vapor code for the if statement and also adjust the indentation which is in charge of entry and exit of the current scope. The vapor code being generated uses goto: label to handle the logic transitions of the if statement.



```
536         * f6 -> Statement()
537         */
538 @↑@     public R visit(IfStatement n, A argu) {
539             R _ret=null;
540
541             String elseLabel = createLabel();
542             String endIfLabel = createLabel();
543
544             n.f0.accept( v: this, argu);
545             n.f1.accept( v: this, argu);
546             String boolRes = (String) n.f2.accept( v: this, argu);
547             n.f3.accept( v: this, argu);
548
549             if (findRecord(currentClass) != null) {
550                 int fieldOff = findRecord(currentClass).getFieldOffset(boolRes);
551                 if (fieldOff != -1) {
552                     String Temp_RHS = createTemp();
553                     genvap.addLine( l: Temp_RHS + " = [this + " + (fieldOff * 4) + "]");
554                     boolRes = Temp_RHS;
555                 }
556             }
557
558             genvap.addLine( l: "if0 " + boolRes + " goto " + ":" + elseLabel);
559             genvap.increaseIndent();
560             n.f4.accept( v: this, argu);
561             genvap.addLine( l: "goto :" + endIfLabel);
562             genvap.descreaseIndent();
563             genvap.addLine( l: elseLabel + ":");
564             genvap.increaseIndent();
565             n.f5.accept( v: this, argu);
566             n.f6.accept( v: this, argu);
567             genvap.descreaseIndent();
568             genvap.addLine( l: endIfLabel + ":");
569
570             return _ret;
571         }
572
```

Now in my last example I'll show the simple implementation of vTable and a few helping methods which are responsible for adding a function, getting a function's offset, and any function's label. The only data members of vTable are "Vname" which holds the Vtable's name and a list of strings to represent the functions in the vTable.

```java
package vapor_code_gen;

import ...

public class VTable {

    String Vname;
    public List<String> functions;

    public VTable(String Vname) {
        this.Vname = Vname;
        functions = new ArrayList<>();
    }

    public void addFunction(String label) { functions.add(label); }

    public int getFunctionOffset(String key) {
        int counterTemp = 0;
        Iterator<String> x = functions.iterator();
        while(x.hasNext()) {
            String currentTemp = x.next();

            int subTemp = currentTemp.indexOf("_" + key);
            if (subTemp != -1)
            if (currentTemp.substring(subTemp).equals("_" + key))
                return counterTemp;

            counterTemp++;
        }
        return -1;
    }

    public String getFunctionLabel(int i) { return functions.get(i); }
}

```

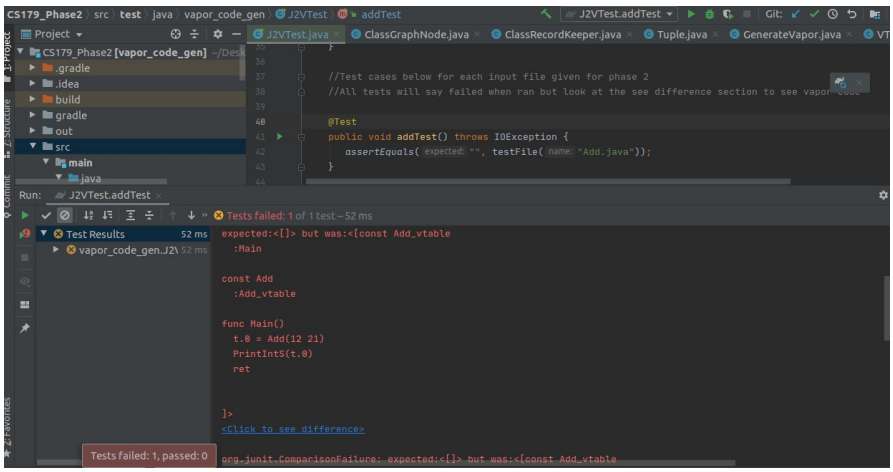
Testing and Verification

In order to verify that our vapor code generator was working correctly, we used the following test cases and the following results appeared:

Test:	Result:
Add.java	pass
BinaryTree.java	pass
BubbleSort.java	pass
Call.java	pass
Factorial.java	pass
LinearSearch.java	pass
LinkenList.java	pass
Morethan4.java	pass
OutOfBounds.error	pass
PrintLiteral.java	pass
QuickSort.java	pass
TreeVistor.java	pass
Vars.java	pass

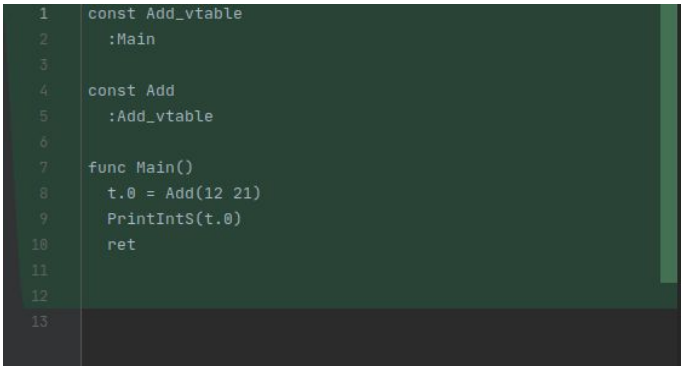
*I continue to have the same problem when it comes to compiling my source code through the terminal, so instead I set up a similar testing system as phase 1 using gradle and Junit with my Intellij IDE. I'm hoping I'll figure out a fix to my problem soon but this project has been a lot of work, so I've been more focused on finishing my code on my local machine and writing the documentation. My test file can be found by navigating to /src/test/java/vapor_code_gen and clicking on the file called J2VTest.java. Once there you can click on the green arrow on line 12 which will run all of the tests and input programs at once, but doing it that way creates a lot of output code to go through.

So instead I would suggest running each test individually by clicking on the green arrows starting at line 41 - 101, this way the output is more readable. Each test will say it “failed” after running but this is because the expected parameter of assertEquals is set to “ “ or nothing, since the output is generated vapor code. To get the best look at the output source code scroll up on the TestResults section until you see “<Click to see difference” then click there to see the real expected output. I’ve attached screenshots below of how to view the results of add.java in intellij, and there will also be screenshots of the vapor source code output for each of the phase 2 test files. However some of the output files are long, so some of the output screenshots will show the beginning and end of those files. Finally to check whether the input and output program behave the same way, I had to run the output program using the Vapor interpreter “vapor.jar”, which was given to use in the instructions. -David



*Pictures of output (vapor source code) for each input file is below

Add.java



BinaryTree.java

```
const Tree_vtable
  :Tree_GetHas_Right
  :Tree_SetHas_Right
  :Tree_Init
  :Tree_RecPrint
  :Tree_Compare
  :Tree_Insert
  :Tree_RemoveLeft
  :Tree_Delete
  :Tree_SetKey
  :Tree_GetHas_Left
  :Tree_GetKey
  :Tree_SetHas_Left
  :Tree_Search
  :Tree_RemoveRight
  :Tree_SetLeft
  :Tree_Remove
  :Tree_GetLeft
  :Tree_Print
  :Tree_GetRight
  :Tree_SetRight

const Tree
  :Tree_vtable

const BinaryTree_vtable
  :Main

const BinaryTree
  :BinaryTree_vtable

```

```
31
32  const BT_vtable
33    :BT_Start
34
35  const BT
36    :BT_vtable
37
38  func Main()
39    t.0 = HeapAllocZ(4)
40    [t.0] = :BT
41    t.1 = [t.0]
42    t.1 = [t.1]
43    t.1 = [t.1 + 0]
44    t.2 = call t.1(t.0)
45    PrintIntS(t.2)
46    ret
47
48  func BT_Start(this)
49    t.3 = HeapAllocZ(28)
50    [t.3] = :Tree
51    root = t.3
52    vt = [root]
53    vt = [vt]
54    f = [vt + 8]
55    t.5 = call f(root 16 )
56    ntb = t.5
57    vt = [root]
58    vt = [vt]
59    f = [vt + 68]
60    t.7 = call f(root)
61    ntb = t.7
62    PrintIntS(100000000)
```

```
637       ntb = t.179
638       goto :l.53
639     l.52:
640       ntb = 1
641     l.53:
642     vt = [node]
643     vt = [vt]
644     f = [vt + 40]
645     t.182 = call f(node)
646     PrintIntS(t.182)
647     vt = [node]
648     vt = [vt]
649     f = [vt + 0]
650     t.184 = call f(node)
651     if0 t.184 goto :l.54
652       vt = [node]
653       vt = [vt]
654       f = [vt + 72]
655       t.186 = call f(node)
656       t.189 = [this]
657       t.189 = [t.189]
658       t.189 = [t.189 + 12]
659       t.188 = call t.189(this t.186 )
660       ntb = t.188
661       goto :l.55
662     l.54:
663       ntb = 1
664     l.55:
665     ret 1
```

BubbleSort.java

```
const BubbleSort_vtable
  :Main

const BubbleSort
  :BubbleSort_vtable

const BBS_vtable
  :BBS_Init
  :BBS_Start
  :BBS_Sort
  :BBS_Print

const BBS
  :BBS_vtable

func Main()
  t.0 = HeapAllocZ(12)
  [t.0] = :BBS
  t.1 = [t.0]
  t.1 = [t.1]
  t.1 = [t.1 + 4]
  t.2 = call t.1(t.0 10 )
  PrintIntS(t.2)
  ret

func BBS_Start(this sz)
  t.5 = [this]
  t.5 = [t.5]
  t.5 = [t.5 + 0]
  t.4 = call t.5(this sz )
  aux01 = t.4
  t.8 = [this]
```

```
  t.73 = MulS(5 4)
  t.74 = Add(t.72 t.73)
  t.74 = Add(t.74 4)
  [t.74] = 11
  t.75 = Add(this 8)
  t.75 = [t.75]
  t.76 = MulS(6 4)
  t.77 = Add(t.75 t.76)
  t.77 = Add(t.77 4)
  [t.77] = 6
  t.78 = Add(this 8)
  t.78 = [t.78]
  t.79 = MulS(7 4)
  t.80 = Add(t.78 t.79)
  t.80 = Add(t.80 4)
  [t.80] = 9
  t.81 = Add(this 8)
  t.81 = [t.81]
  t.82 = MulS(8 4)
  t.83 = Add(t.81 t.82)
  t.83 = Add(t.83 4)
  [t.83] = 19
  t.84 = Add(this 8)
  t.84 = [t.84]
  t.85 = MulS(9 4)
  t.86 = Add(t.84 t.85)
  t.86 = Add(t.86 4)
  [t.86] = 5
  ret 0
```

Call.java

```
const A_vtable
  :A_run

const A
  :A_vtable

const Call_vtable
  :Main

const Call
  :Call_vtable

func Main()
  t.0 = HeapAllocZ(4)
  [t.0] = :A
  t.1 = [t.0]
  t.1 = [t.1]
  t.1 = [t.1 + 0]
  t.2 = call t.1(t.0)
  PrintIntS(t.2)
  ret

func A_run(this)
  PrintIntS(42)
  ret 99


```

Factorial.java

```
const Factorial_vtable
  :Main

const Factorial
  :Factorial_vtable

const Fac_vtable
  :Fac_ComputeFac

const Fac
  :Fac_vtable

func Main()
  t.0 = HeapAllocZ(4)
  [t.0] = :Fac
  t.1 = [t.0]
  t.1 = [t.1]
  t.1 = [t.1 + 0]
  t.2 = call t.1(t.0 10 )
  PrintIntS(t.2)
  ret

func Fac_ComputeFac(this num)
  t.3 = LtS(num 1)
  if0 t.3 goto :l.0
    num_aux = 1
    goto :l.1
  l.0:
    t.4 = Sub(num 1)
    t.7 = [this]
    t.7 = [t.7]
    t.7 = [t.7 + 0]
```

```
func Main()
  t.0 = HeapAllocZ(4)
  [t.0] = :Fac
  t.1 = [t.0]
  t.1 = [t.1]
  t.1 = [t.1 + 0]
  t.2 = call t.1(t.0 10 )
  PrintIntS(t.2)
  ret

func Fac_ComputeFac(this num)
  t.3 = LtS(num 1)
  if0 t.3 goto :l.0
    num_aux = 1
    goto :l.1
  l.0:
    t.4 = Sub(num 1)
    t.7 = [this]
    t.7 = [t.7]
    t.7 = [t.7 + 0]
    t.6 = call t.7(this t.4 )
    t.8 = MulS(num t.6)
    num_aux = t.8
  l.1:
  ret num_aux
```

LinearSearch.java

```
const LS_vtable
  :LS_Init
  :LS_Start
  :LS_Search
  :LS_Print

const LS
  :LS_vtable

const LinearSearch_vtable
  :Main

const LinearSearch
  :LinearSearch_vtable

func Main()
  t.0 = HeapAllocZ(12)
  [t.0] = :LS
  t.1 = [t.0]
  t.1 = [t.1]
  t.1 = [t.1 + 4]
  t.2 = call t.1(t.0 10 )
  PrintIntS(t.2)
  ret

func LS_Start(this sz)
  t.5 = [this]
  t.5 = [t.5]
  t.5 = [t.5 + 0]
  t.4 = call t.5(this sz )
  aux01 = t.4
  t.8 = [this]
```

```
  t.39 = HeapAllocZ(t.40)
  [t.39] = sz
  [this + 8] = t.39
  j = 1
  t.41 = [this + 4]
  t.42 = Add(t.41 1)
  k = t.42
  l.14:
    t.43 = [this + 4]
    t.44 = LtS(j t.43)
    if0 t.44 goto :l.15
    t.45 = MulS(2 j)
    aux01 = t.45
    t.46 = Sub(k 3)
    aux02 = t.46
    t.47 = Add(aux01 aux02)
    t.48 = Add(this 8)
    t.48 = [t.48]
    t.49 = MulS(j 4)
    t.50 = Add(t.48 t.49)
    t.50 = Add(t.50 4)
    [t.50] = t.47
    t.51 = Add(j 1)
    j = t.51
    t.52 = Sub(k 1)
    k = t.52
  goto :l.14
  l.15:
  ret 0
```

LinkedList.java

```
const LL_vtable
  :LL_Start

const LL
  :LL_vtable

const List_vtable
  :List_GetNext
  :List_Init
  :List_SetNext
  :List_Insert
  :List_Delete
  :List_InitNew
  :List_Search
  :List_GetEnd
  :List_GetElem
  :List_Print

const List
  :List_vtable

const Element_vtable
  :Element_GetSalary
  :Element_Init
  :Element_Compare
  :Element_GetMarried
  :Element_Equal
  :Element_GetAge

const Element
  :Element_vtable
```

```
    t.119 = call f(head)
    aux01 = t.119
    PrintIntS(2220000)
    vt = [head]
    vt = [vt]
    f = [vt + 16]
    t.121 = call f(head el02 )
    head = t.121
    vt = [head]
    vt = [vt]
    f = [vt + 36]
    t.123 = call f(head)
    aux01 = t.123
    PrintIntS(33300000)
    vt = [head]
    vt = [vt]
    f = [vt + 16]
    t.125 = call f(head el01 )
    head = t.125
    vt = [head]
    vt = [vt]
    f = [vt + 36]
    t.127 = call f(head)
    aux01 = t.127
    PrintIntS(44440000)
    ret 0
```

MoreThan4.java

```
const MoreThan4_vtable
  :Main

const MoreThan4
  :MoreThan4_vtable

const MT4_vtable
  :MT4_Start
  :MT4_Change

const MT4
  :MT4_vtable

func Main()
  t.0 = HeapAllocZ(4)
  [t.0] = :MT4
  t.1 = [t.0]
  t.1 = [t.1]
  t.1 = [t.1 + 0]
  t.2 = call t.1(t.0 1 2 3 4 5 6 )
  PrintIntS(t.2)
  ret

func MT4_Start(this p1 p2 p3 p4 p5 p6)
  PrintIntS(p1)
  PrintIntS(p2)
  PrintIntS(p3)
  PrintIntS(p4)
  PrintIntS(p5)
  PrintIntS(p6)
  t.5 = [this]
  t.5 = [t.5]
  t.5 = [t.5 + 4]
  t.4 = call t.5(this p6 p5 p4 p3 p2 p1 )
  aux = t.4
  ret aux

func MT4_Change(this p1 p2 p3 p4 p5 p6)
  PrintIntS(p1)
  PrintIntS(p2)
  PrintIntS(p3)
  PrintIntS(p4)
  PrintIntS(p5)
  PrintIntS(p6)
  ret 0
```

OutOfBounds.error

```
const OutOfBounds_vtable
  :Main

const OutOfBounds
  :OutOfBounds_vtable

const A_vtable
  :A_run

const A
  :A_vtable

func Main()
  t.0 = HeapAllocZ(4)
  [t.0] = :A
  t.1 = [t.0]
  t.1 = [t.1]
  t.1 = [t.1 + 0]
  t.2 = call t.1(t.0)
  PrintIntS(t.2)
  ret

func A_run(this)
  t.4 = MulS(20 4)
  t.4 = Add(t.4 4)
  t.3 = HeapAllocZ(t.4)
  [t.3] = 20
  a = t.3
  t.6 = 10
  t.7 = [a]
  ok = LtS(t.6 t.7)
```

```
  if ok goto :l.0
  Error("array index out of bounds")
  l.0:
  ok = LtS(-1 10)
  if ok goto :l.1
  Error("array index out of bounds")
  l.1:
  t.6 = MulS(10 4)
  t.6 = Add(a t.6)
  t.6 = Add(t.6 4)
  t.5 = [t.6]
  PrintIntS(t.5)
  t.9 = 40
  t.10 = [a]
  ok = LtS(t.9 t.10)
  if ok goto :l.2
  Error("array index out of bounds")
  l.2:
  ok = LtS(-1 40)
  if ok goto :l.3
  Error("array index out of bounds")
  l.3:
  t.9 = MulS(40 4)
  t.9 = Add(a t.9)
  t.9 = Add(t.9 4)
  t.8 = [t.9]
  ret t.8
```

PrintLiteral.java

```
const PrintLiteral_vtable
  :Main

const PrintLiteral
  :PrintLiteral_vtable

func Main()
  PrintIntS(12)
  ret

```

QuickSort.java

```
const QS_vtable
  :QS_Init
  :QS_Start
  :QS_Sort
  :QS_Print

const QS
  :QS_vtable

const QuickSort_vtable
  :Main

const QuickSort
  :QuickSort_vtable

func Main()
  t.0 = HeapAllocZ(12)
  [t.0] = :QS
  t.1 = [t.0]
  t.1 = [t.1]
  t.1 = [t.1 + 4]
  t.2 = call t.1(t.0 10 )
  PrintIntS(t.2)
  ret

func QS_Start(this sz)
  t.5 = [this]
  t.5 = [t.5]
  t.5 = [t.5 + 0]
  t.4 = call t.5(this sz )
  aux01 = t.4
```

```
342     t.104 = MulS(5 4)
343     t.105 = Add(t.103 t.104)
344     t.105 = Add(t.105 4)
345     [t.105] = 11
346     t.106 = Add(this 8)
347     t.106 = [t.106]
348     t.107 = MulS(6 4)
349     t.108 = Add(t.106 t.107)
350     t.108 = Add(t.108 4)
351     [t.108] = 6
352     t.109 = Add(this 8)
353     t.109 = [t.109]
354     t.110 = MulS(7 4)
355     t.111 = Add(t.109 t.110)
356     t.111 = Add(t.111 4)
357     [t.111] = 9
358     t.112 = Add(this 8)
359     t.112 = [t.112]
360     t.113 = MulS(8 4)
361     t.114 = Add(t.112 t.113)
362     t.114 = Add(t.114 4)
363     [t.114] = 19
364     t.115 = Add(this 8)
365     t.115 = [t.115]
366     t.116 = MulS(9 4)
367     t.117 = Add(t.115 t.116)
368     t.117 = Add(t.117 4)
369     [t.117] = 5
370     ret 0
```

TreeVisitor.java

```
1   const TV_vtable
2     :TV_Start
3
4   const TV
5     :TV_vtable
6
7   const TreeVisitor_vtable
8     :Main
9
10  const TreeVisitor
11    :TreeVisitor_vtable
12
13  const Tree_vtable
14    :Tree_GetHas_Right
15    :Tree_SetHas_Right
16    :Tree_Init
17    :Tree_RecPrint
18    :Tree_Compare
19    :Tree_Insert
20    :Tree_RemoveLeft
21    :Tree_Delete
22    :Tree_SetKey
23    :Tree_GetHas_Left
24    :Tree_GetKey
25    :Tree_SetHas_Left
26    :Tree_Search
27    :Tree_RemoveRight
28    :Tree_SetLeft
29    :Tree_Remove
30    :Tree_GetLeft
31    :Tree_Print
```

```
    :Tree_accept
    :Tree_GetRight
    :Tree_SetRight

const Tree
    :Tree_vtable

const Visitor_vtable
    :Visitor_visit

const Visitor
    :Visitor_vtable

const MyVisitor_vtable
    :MyVisitor_visit

const MyVisitor
    :MyVisitor_vtable

func Main()
    t.0 = HeapAllocZ(4)
    [t.0] = :TV
    t.1 = [t.0]
    t.1 = [t.1]
    t.1 = [t.1 + 0]
    t.2 = call t.1(t.0)
    PrintIntS(t.2)
    ret

func TV_Start(this)
```

```
    nti = 0
  l.61:
  vt = [n]
  vt = [vt]
  f = [vt + 40]
  t.215 = call f(n)
  PrintIntS(t.215)
  vt = [n]
  vt = [vt]
  f = [vt + 36]
  t.217 = call f(n)
  if0 t.217 goto :l.62
    vt = [n]
    vt = [vt]
    f = [vt + 64]
    t.219 = call f(n)
    [this + 4] = t.219
    t.222 = [this + 4]
    vt = [t.222]
    vt = [vt]
    f = [vt + 72]
    t.221 = call f(t.222 this )
    nti = t.221
    goto :l.63
  l.62:
    nti = 0
  l.63:
  ret 0
```

Vars.java

```
const A_vtable
  :A_run
  :A_helper

const A
  :A_vtable

const Vars_vtable
  :Main

const Vars
  :Vars_vtable

func Main()
  t.0 = HeapAllocZ(4)
  [t.0] = :A
  t.1 = [t.0]
  t.1 = [t.1]
  t.1 = [t.1 + 0]
  t.2 = call t.1(t.0)
  PrintIntS(t.2)
  ret

func A_run(this)
  t.5 = [this]
  t.5 = [t.5]
  t.5 = [t.5 + 4]
  t.4 = call t.5(this 12 )
  a = t.4
  t.8 = [this]
  t.8 = [t.8]
```

```
  t.8 = [t.8 + 4]
  t.7 = call t.8(this 15 )
  b = t.7
  t.9 = Add(a b)
  ret t.9

func A_helper(this param)
  x = param
  t.10 = Add(param 1)
  param = t.10
  PrintIntS(x)
  ret x

```