Project and Report Ideas:

- Write a translator from a small (non-Java) language to JVM byte code.
- Write a graphical simulator for (some or all of) the JVM.
- Write a byte code compression utility and a corresponding class loader.
- Write a byte code optimizer that tries to make better use of the stack operators.
- Implement a new garbage collection strategy for an existing JVM implementation.
- Survey, compare, and report on several obfuscators or reverse compilers.
- Survey, compare, and report on several whole-program analysis techniques.
- Survey and report on use of formal methods for specifying Java or byte code.
- Do a careful performance comparison on several existing JIT compilers for some architecture.
- Survey, compare, and report on algorithms for JIT compilation.

Small teams are encouraged.

Re-use of publicly-available tools is encouraged.

```
class NonConforming extends Error {};

class MatrixMult {
  static Matrix mmult(Matrix a, Matrix b) {
    int i,j,k;
    if (a.cols() != b.rows())
        throw new NonConforming();
    Matrix c = new Matrix(a.rows(),b.cols());
    for (i = 0; i < a.rows(); i++)
      for (j = 0; j < b.cols(); j++) {
        double d = 0.0;
        for (k = 0; k < a.cols(); k++)
          d += a.get(i,k) * b.get(k,j);
        c.set(i,j,d);
      };
    return c;
  }

  public static void main(String s[])
  {
    Matrix a,b,c;
    ...
    c = mult(a,b);
  }
}
```

```
import java.util.*;
import java.lang.*;

class Matrix {
  private double d[][];
  Matrix(int m, int n) {
    d = new double[m][n];
  };
  double get(int i,int j) {
    return d[i][j];
  };
  void set (int i, int j, double x) {
    d[i][j] = x;
  }
  int rows() {return d.length;}
  int cols() {return d[0].length;}
}
```

```
Method Matrix(int,int)
   0 aload_0
   1 invokespecial #12 <Method java.lang.Object()>
   4 aload_0
   5 iload_1
   6 iload_2
   7 multianewarray #6 dim #2 <Class [[D>
  11 putfield #16 <Field double d[][]>
  14 return

Method double get(int, int)
   0 aload_0
   1 getfield #16 <Field double d[][]>
   4 iload_1
   5 aaload
   6 iload_2
   7 daload
   8 dreturn

Method void set(int, int, double)
   0 aload_0
   1 getfield #16 <Field double d[][]>
   4 iload_1
   5 aaload
   6 iload_2
   7 dload_3
   8 dastore
   9 return
```

```
Method int rows()
   0 aload_0
   1 getfield #16 <Field double d[][]>
   4 arraylength
   5 ireturn

Method int cols()
   0 aload_0
   1 getfield #16 <Field double d[][]>
   4 iconst_0
   5 aaload
   6 arraylength
   7 ireturn
```

```
   55 dload 6
   57 aload_0
   58 iload_2
   59 iload 4
   61 invokevirtual #21 <Method double get(int, int)>
   64 aload_1
   65 iload 4
   67 iload_3
   68 invokevirtual #21 <Method double get(int, int)>
   71 dmul
   72 dadd
   73 dstore 6
   75 iinc 4 1
   78 iload 4
   80 aload_0
   81 getfield #20 <Field int cols>
   84 if_icmplt 55
   87 aload 5
   89 iload_2
   90 iload_3
   91 dload 6
   93 invokevirtual #29 <Method void set(int, int, double)>
   96 iinc 3 1
   99 iload_3
  100 aload_1
  101 getfield #20 <Field int cols>
  104 if_icmplt 46
  107 iinc 2 1
  110 iload_2
  111 aload_0
  112 getfield #28 <Field int rows>
  115 if_icmplt 41
  118 aload 5
  120 areturn
```

```
Method Matrix mmult(Matrix, Matrix)
   0 aload_0
   1 getfield #20 <Field int cols>
   4 aload_1
   5 getfield #28 <Field int rows>
   8 if_icmpeq 19
  11 new #5 <Class NonConforming>
  14 dup
  15 invokespecial #13 <Method NonConforming()>
  18 athrow
  19 new #3 <Class Matrix>
  22 dup
  23 aload_0
  24 getfield #28 <Field int rows>
  27 aload_1
  28 getfield #20 <Field int cols>
  31 invokespecial #16 <Method Matrix(int,int)>
  34 astore 5
  36 iconst_0
  37 istore_2
  38 goto 110
  41 iconst_0
  42 istore_3
  43 goto 99
  46 dconst_0
  47 dstore 6
  49 iconst_0
  50 istore 4
  52 goto 78
```

If `Matrix` is marked `final` and `d` is unmarked `private` and we compile with -O:

```
Method Matrix mmult(Matrix, Matrix)
   0 aload_0
   1 getfield #19 <Field double d[][]>
   4 iconst_0
   5 aaload
   6 arraylength
   7 aload_1
   8 getfield #19 <Field double d[][]>
  11 arraylength
  12 if_icmpeq 23
  15 new #4 <Class NonConforming>
  18 dup
  19 invokespecial #12 <Method NonConforming()>
  22 athrow
  23 new #2 <Class Matrix>
  26 dup
  27 aload_0
  28 getfield #19 <Field double d[][]>
  31 arraylength
  32 aload_1
  33 getfield #19 <Field double d[][]>
  36 iconst_0
  37 aaload
  38 arraylength
  39 invokespecial #15 <Method Matrix(int,int)>
  42 astore 5
  44 iconst_0
  45 istore_2
  46 goto 130
```

```
49 iconst_0
50 istore_3
51 goto 116
54 dconst_0
55 dstore 6
57 iconst_0
58 istore 4
60 goto 90
63 dload 6
65 aload_0
66 getfield #19 <Field double d[][]>
69 iload_2
70 aaload
71 iload 4
73 daload
74 aload_1
75 getfield #19 <Field double d[][]>
78 iload 4
80 aaload
81 iload_3
82 daload
83 dmul
84 dadd
85 dstore 6
87 iinc 4 1
90 iload 4
92 aload_0
93 getfield #19 <Field double d[][]>
96 iconst_0
97 aaload
98 arraylength
99 if_icmplt 63
```

```
102 aload 5
104 getfield #19 <Field double d[][]>
107 iload_2
108 aaload
109 iload_3
110 dload 6
112 dastore
113 iinc 3 1
116 iload_3
117 aload_1
118 getfield #19 <Field double d[][]>
121 iconst_0
122 aaload
123 arraylength
124 if_icmplt 54
127 iinc 2 1
130 iload_2
131 aload_0
132 getfield #19 <Field double d[][]>
135 arraylength
136 if_icmplt 49
139 aload 5
141 areturn
```

Sample times (seconds; 133MHz Pentium, 100 multiplies of a 60x60 double matrix):

Java interpreter	(no inlining)	87.4
	(inlining)	43.0
Kaffe JIT compiler	(no inlining)	39.2
	(inlining)	28.3
C code	(checked)	7.1
	(unchecked)	3.9