The Java Compiler -- More Limitations

1. Cannot handle boolean operations except a boolean-type method returning a boolean value (true/false).
2. Cannot handle array initialization, such as:

```
final float sideVec[]={0,0,1,0,0,-1,0,-1,0,1,0,0,0,1,0,-1,0,0};
```

This is done in a way that: 1) count the number of initializers, which determines the size of the array to be initialized. 2) Allocate the array; 3) Assign values to the elements of the array.

3. Treat such numbers as 1.1 as `CONSTANT_Float`, not `CONSTANT_Double`. Treat such numbers as 11 as `CONSTANT_Integer`, not `CONSTANT_Long`. Therefore, the following code will not work:

```
double a = 11;
```

Discussion:

In order to make the compiler be able to handle double-precision data, the following should be made:

i) Store double precision numbers into the constant pool, each number will occupy two spots in the constant pool.

ii) Generate functions that can search/add double-precision numbers in the constant pool.

iii) Generate functions that can generate assmbler code for double-precision numbers. At present, the compiler treats all floating-point constants as single-precision numbers while in the SUN compiler they are treated as double-precision ones. For example:

For the code: `float a = 11.1`;

The compiler can generate code as:

```
ldc # <Real 11.100000>
fstore_...
```

However, the correct implementation shuld be:

```
ldc2w # <Double  11.100000>
d2f
fstore_...
```

For the code float a = 11;

The correect implementation is:

```
ldc # <Real 11.000000>
fstore_...
```

iv) Generate assembler instructions for the double-precision operations.

4. The compiler cannot handle exceptions.