1 Goal

The goal of the project is to compile an arbritrary Eiffel program to Java Byte Code. The generated program should run on a standart Java Virtual Machine. A code generator is added to the ISE Eiffel Compiler to do the Job. A Java programer should be able to use this generated classes in his code in a straight forward way. In particular he should be able to inherit from such generated classes and use them as a client. The generated code should also not loose the information about the subtyping structure of the Eiffel program.

2 Overview

Multiple inheritance can be modeled in Java with the help of interfaces. The interfaces only reflect the subtyping relation of Eiffel classes. For every Interface a class is generated which implements that interface and contains the translated code from the corresponding Eiffel class. Since multible subclassing is not possible in Java subclassing is replaced by delegation. An implementing class holds references to other implementing classes. This classes are superclasses in Eiffel.

3 Example

The very simple Eiffel program:

```
1  class
2          CLASS_A
3
4  feature
5
6          f is do σ0 end
7  end
8
9  class
10         CLASS_B
11
12 feature
13
14         g is do σ1 end
15 end
16
17 class
18         CLASS_C
19
20 inherit
21         CLASS_A
22
23         CLASS_B
24
25 end
26
27         local
28                 class_c: CLASS_C
29         do
30                 create class_c
```

```
31                  class_c.f -- yields σ0
32                  class_c.g -- yields σ1
33          end
```

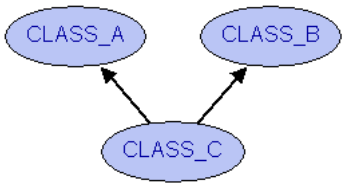

Fig. 1: Multiple inheritance in Eiffel.

can be translated to Java like this:

```
1  public interface ClassA {
2
3      public void f();
4  }
5
6  public interface ClassB {
7
8      public void g();
9  }
10
11 public interface ClassC extends ClassB, ClassA {
12 }
13
14 public class ClassAImpl implements ClassA {
15
16     public void f() {σ0}
17 }
18
19 public class ClassBImpl implements ClassB {
20
21     public void g() {σ1}
22 }
23
24 public class ClassCImpl implements ClassC {
25
26     private ClassAImpl classADelegate = new ClassAImpl();
27     private ClassBImpl classBDelegate = new ClassBImpl();
28
29     public void g() {
30         classBDelegate.g();
31     }
32
33     public void f() {
34         classADelegate.f();
35     }
36 }
```

```
37
38      ClassC classC = new ClassCImpl();
39      classC.f();
40      classC.g();
```

Now the object referenced by classC is of type: ClassCImpl, ClassC, ClassA, ClassB and Object and a call to f() respectively g() will execute the correct code sequences.

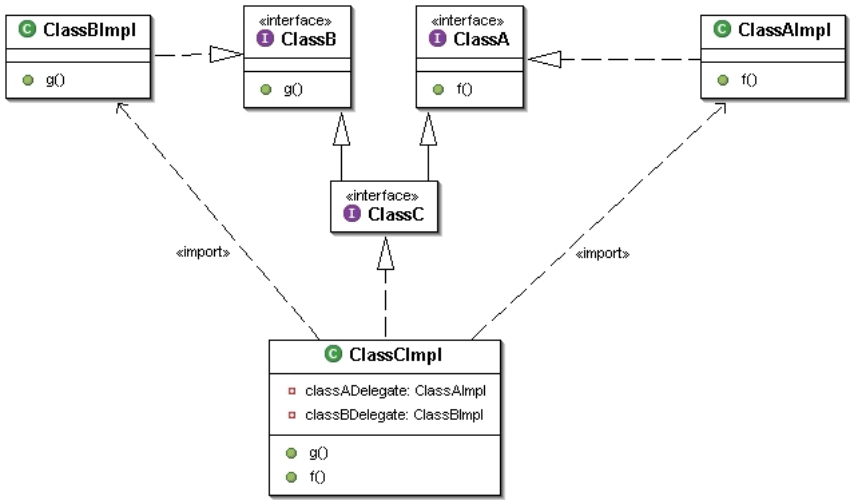

Fig. 2: Multiple inheritance in Java.