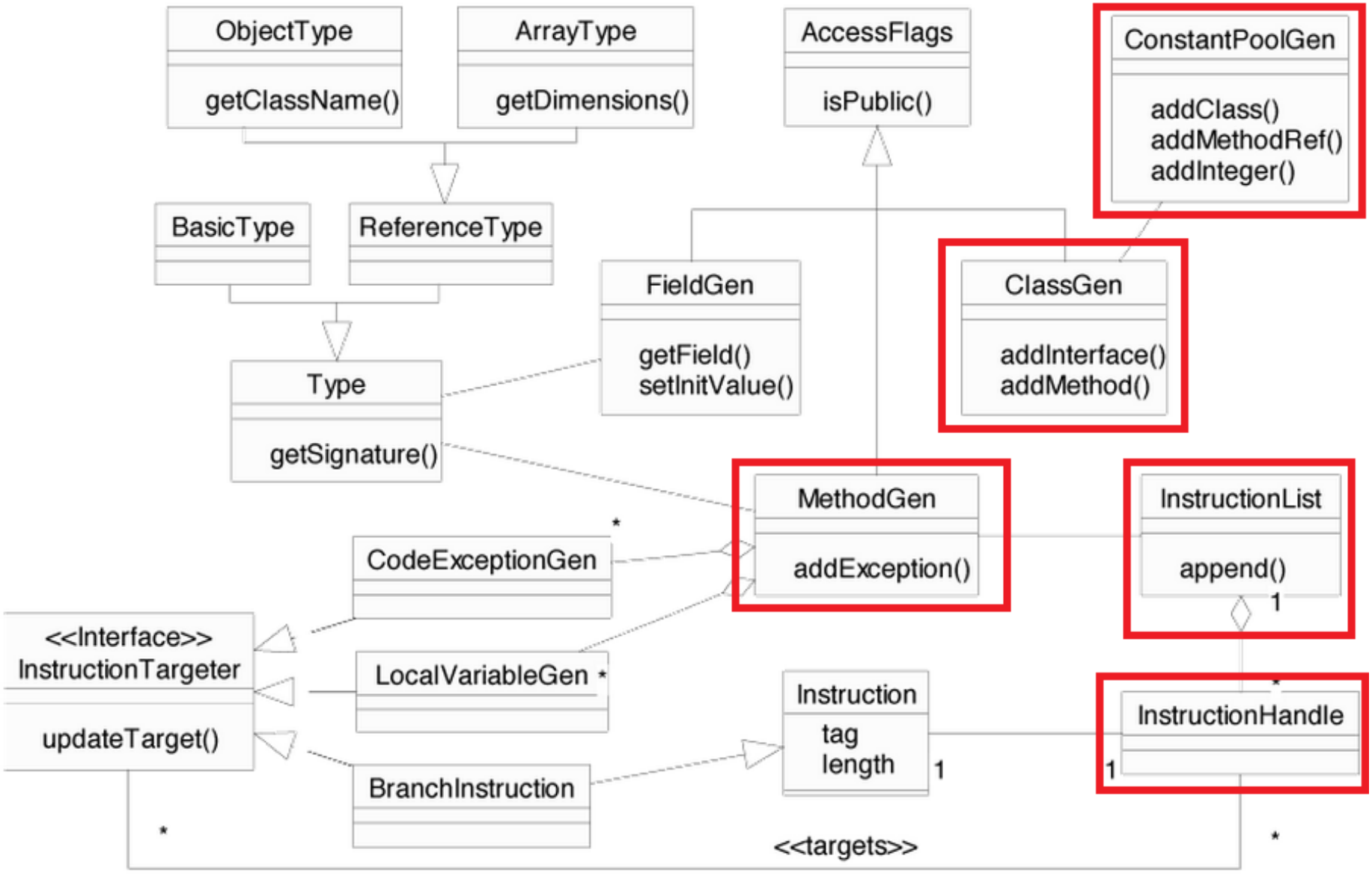
Bytecode-Generierung eines neuartigen Java Compilers

Evelyn Fikus, Franziska Fütterling, Julia Schubert,
Andreas Stadelmeier, Martin Plümicke

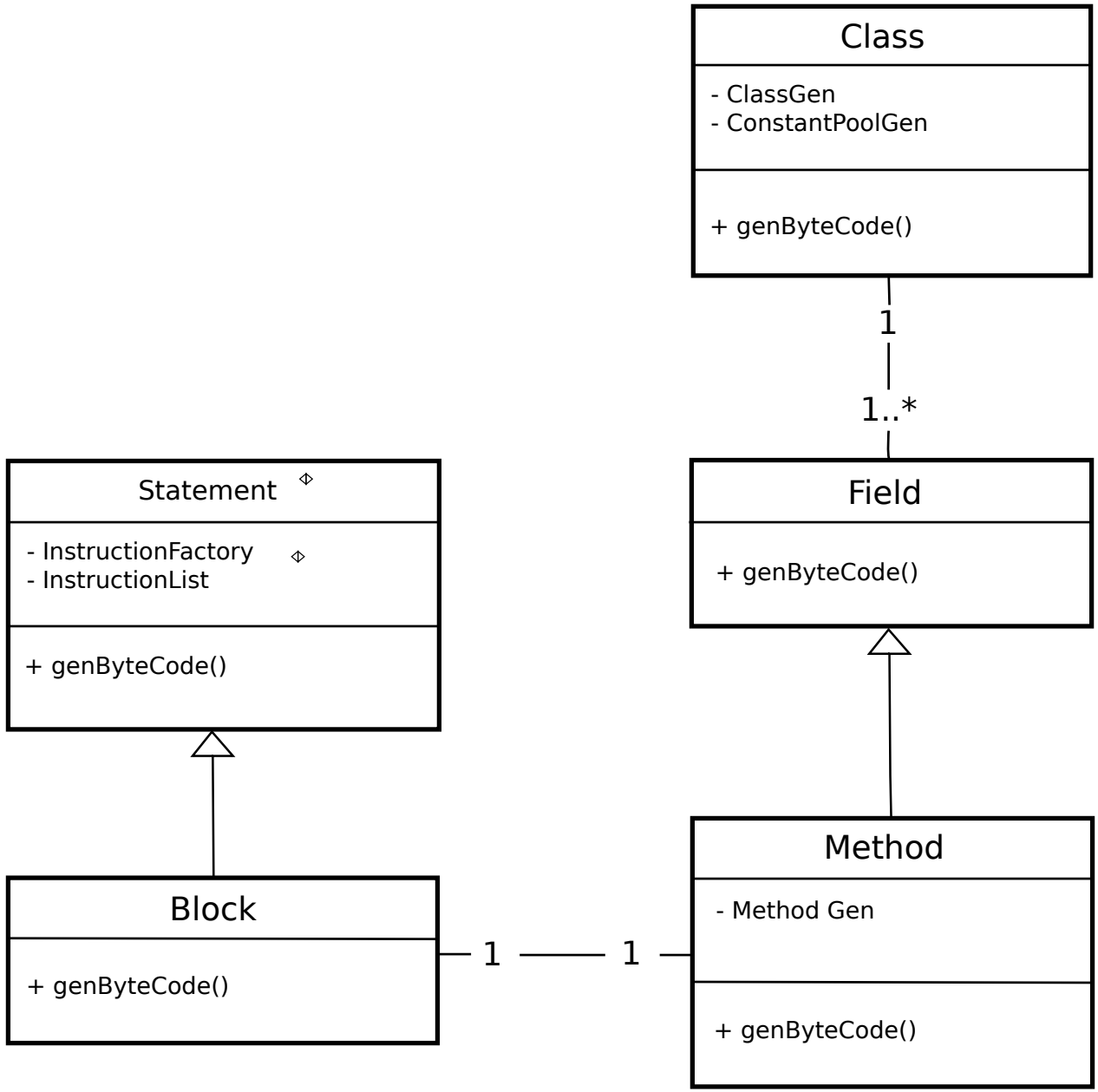
Baden-Wuerttemberg Cooperative State University
Stuttgart/Horb

5. Oktober 2015

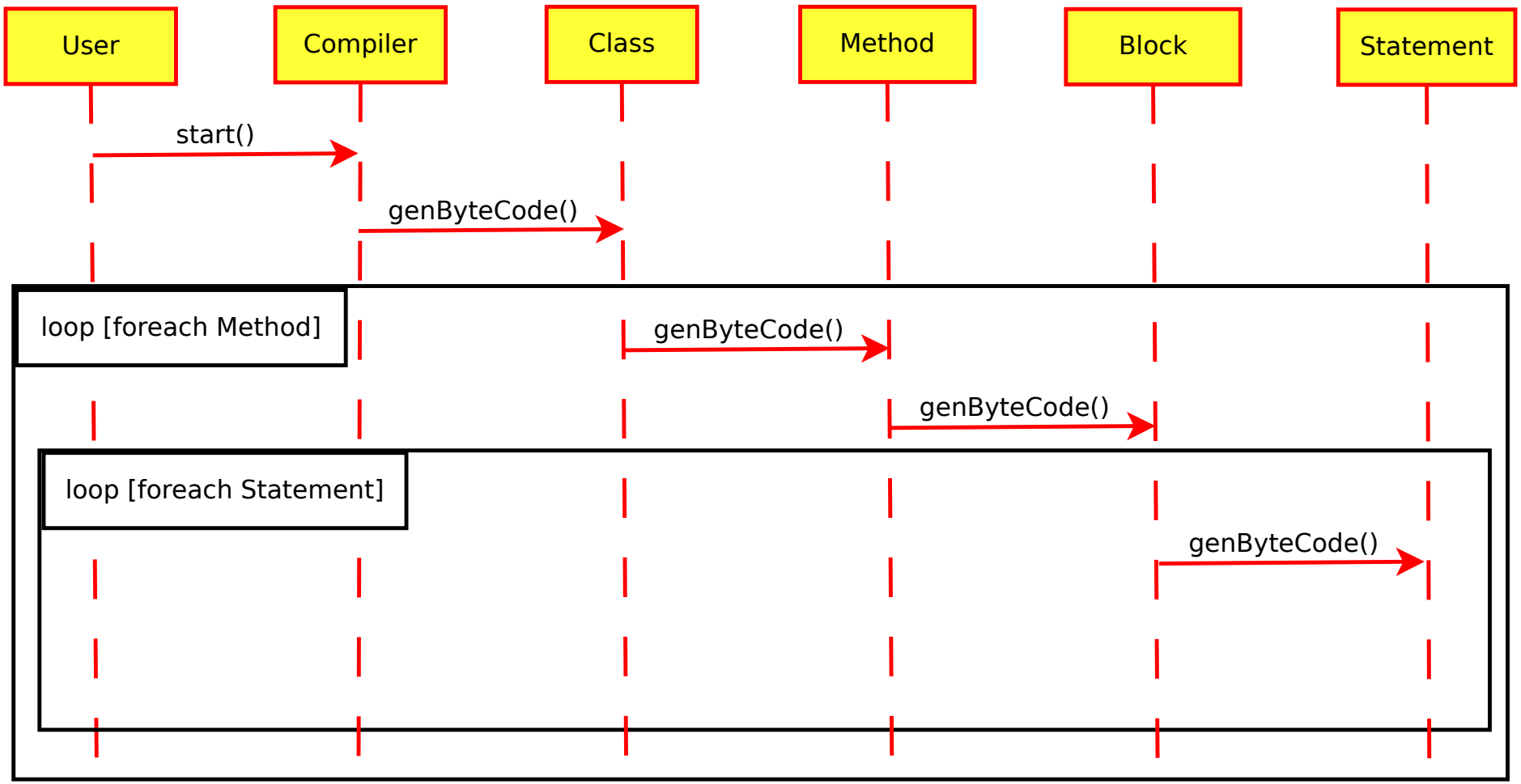
Byte Code Engineering Library (BCEL)



Architektur (Klassendiagramm)



Architektur (Sequenzdiagramm)



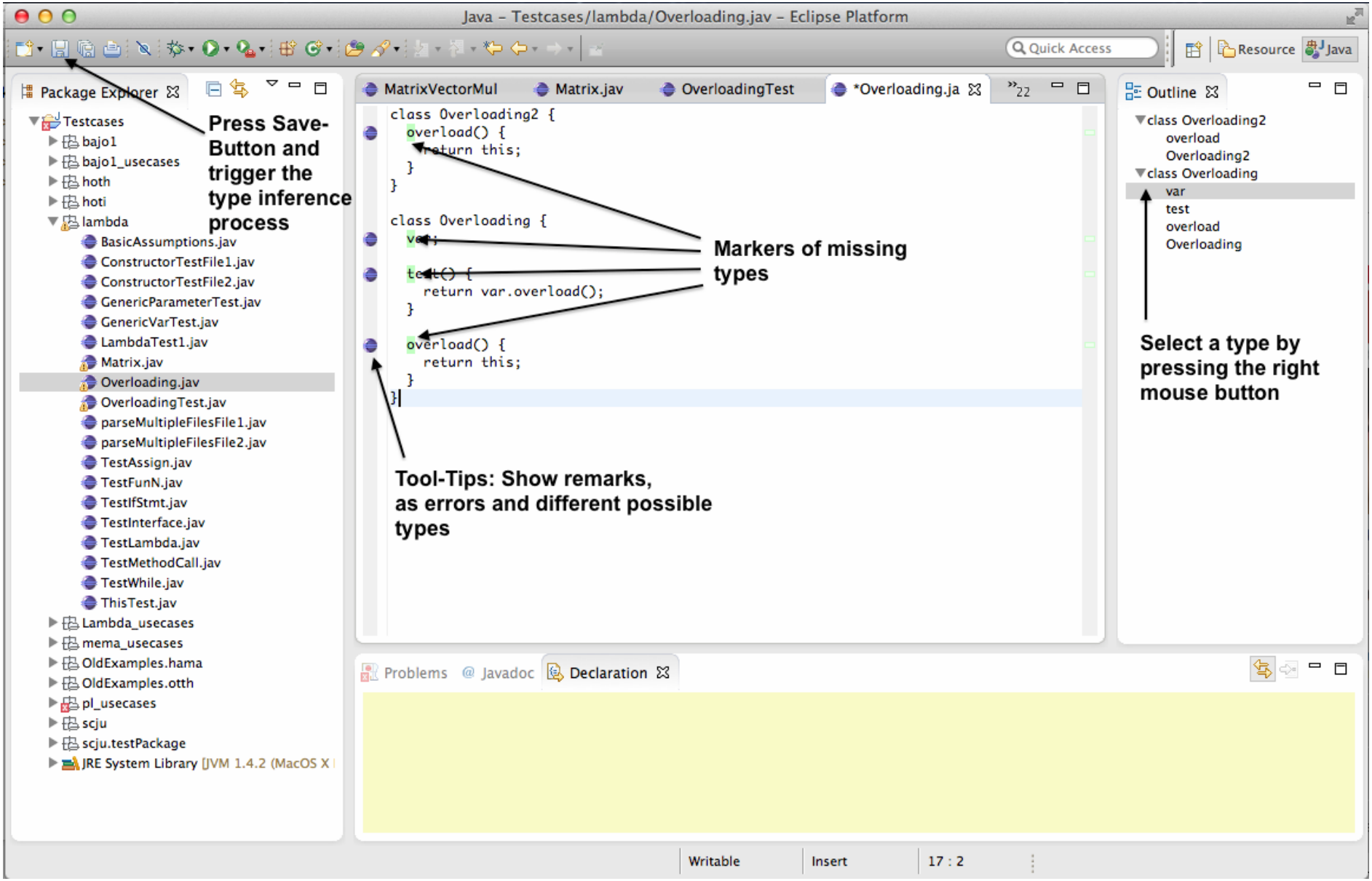
genByteCode in Class.java

```
1  /* genByteCode in Class.java
2  */
3  private ConstantPoolGen     _cp;
4  private ClassGen            _cg;
5
6      public ByteCodeResult genByteCode() throws IOException {
7
8        _cg = new ClassGen(name, superClass.get_Name(), name + ".
             java", Constants.ACC_PUBLIC , new String[] {  });
9        _cp = _cg.getConstantPool();
10
11       for(Field field : this.fielddecl){
12           field.genByteCode(_cg);
13         }
14
15       ByteCodeResult code = new ByteCodeResult(_cg);
16     return code;
17     }
```

genByteCode in Method.java

```
1  public void genByteCode(ClassGen cg) {
2    ConstantPoolGen _cp = cg.getConstantPool();
3    InstructionFactory _factory = new InstructionFactory(cg,_cp);
4    InstructionList il = new InstructionList();
5    Class parentClass = this.getParentClass();
6
7    MethodGen method = new MethodGen(Constants.ACC_PUBLIC, this.
         getType().getBytecodeType(), org.apache.bcel.generic.Type.
         NO_ARGS , new String[] {  }, this.get_Method_Name(),
         parentClass.name, il, _cp);
8
9    Block block = this.get_Block();
10   InstructionList blockInstructions = block.genByteCode(cg);
11   il.append(blockInstructions);
12
13   if (block.get_Statement().size() == 0) { il.append(_factory.
         createReturn(org.apache.bcel.generic.Type.VOID)); }
14   else {
15         if (!(block.get_Statement().lastElement() instanceof
              Return)) { il.append(_factory.createReturn(org.apache
              .bcel.generic.Type.VOID)); }
16     }
17   method.setMaxStack(); //nach dem alle Instructions angehängt wurden
18
19   cg.addMethod(method.getMethod());}
```

Integration ins Eclipse-PlugIn



Vorgehen mit Eclipse

- Wenn nur eine Möglichkeit der Typisierung gegeben ist, wird Bytecode erzeugt.
- Wenn mehrere Typisierungen möglich sind, muss der User auswählen. Erst dann wird Bytecode erzeugt.

Vorgehen mit Eclipse

- Wenn nur eine Möglichkeit der Typisierung gegeben ist, wird Bytecode erzeugt.
- Wenn mehrere Typisierungen möglich sind, muss der User auswählen. Erst dann wird Bytecode erzeugt.

Ziel: Für Intersection–Types wird Bytecode erzeugt.

- Generics in Bytecode
- Fallunterscheidung in überladenen Methoden

Zusammenfassung und Ausblick

Zusammenfassung:

- Bytecodeerzeugung mit BCEL
- Integration in das Eclipse-PlugIn

Ausblick:

- Vollständige Implementierung
- Intelligentere Lösung für die Überladungsauflösung