Klasse *Label*

```
class Label {
    boolean defined;    // true, if label has been defined
    int adr;            // if (defined) adr == position of label in code
                        // else adr == position of prev. fixup label

    Label ();           // creates a new, undefined label

    // inserts offset to label at current pc
    void put ();
    // defines label to be at current pc
    void here ();
    // defines this label to be at position of dest
    void setTo (Label dest);

    // links the other's forward jumps with this's
    //NOT needed for MicroJava-Programs
    void add (Label other);
}
```

Klasse *Item* - Erweiterung für Sprünge

```
class Item {
    static final int              // item kinds
        Con=0, Local=1, Static=2, Stack=3, Fld=4, Elem=5, Meth=6,
          Cond = 7;

    int kind;
    Struct type;          // Typ des Operanden
    Obj obj;              // Meth: Methodenobjekt aus Symbolliste
    int adr;              // Con: Wert; Local, Static, Fld, Meth: Adresse
                          // Cond: Operator (eq=0,ne=1,...)
    Label   tLabel,       // Cond: true jump
            fLabel;       // Cond: false jump
}
```

Klasse *Code* - neue Methoden für Sprünge

```
class Code {
    // generates unconditional jump instruction to lab
    void jump (Label lab);
    // generates conditional jump instruction for true jump
    // x represents the condition
    void tJump (Item x);
    // generates conditional jump instruction for false jump
    // x represents the condition
    void fJump (Item x);
}
```

Klasse *Label* - Methode *put*

```
// inserts offset to label at current pc
void put () {
    int pc = Code.pc;
    if (defined) Code.put2(adr - (pc-1));
    else { Code.put2(adr); adr = pc; }
}
```

Klasse *Label* - Methode *here*

```
// defines label to be at current pc
void here () {
  if (defined) throw new Error("label already defined");

  // fixup
  int next = adr;
  while (next != 0) {
     int pos = next;
     next = Code.get2(pos);
     Code.put2(pos, Code.pc - (pos-1));
  }
  defined = true; adr = Code.pc;
}
```

Übersetzung einer do-while-Anweisung

```
do                      ┌─▶ top:
    Statement           │       code for Statement
while                   │       code for Condition
    Condition;          └────── tJump to top
...                             ...
```

```
DoStatement =              (. Item x; Label top; .)
    "do"                   (. top = new Label(); top.here(); .)
    Statement
    "while"
    "(" Condition↑x        (. x.tLabel.setTo(top);
                              Code.tJump(x); .)
    ")"                    (. x.fLabel.here(); .)
    ";"
```

Klasse *Label* - Methode *setTo*

```
// defines this label to be at position of dest
void setTo (Label dest) {
    if (defined) throw new Error("label already defined");
    if (!dest.defined) throw new Error("destination undefined");

    // fixup
    int next = adr;
    while (next != 0) {
        int pos = next;
        next = Code.get2(next);
        Code.put2(pos, dest.adr - (pos-1));
    }
    defined = true; adr = dest.adr;
}
```