Aufgabe 10: Spracherweiterung: Enumerationstypen

Welche Änderungen wären im Compiler nötig, wenn es in MicroJava Enumerationstypen gäbe? Die Deklaration

```
enum Color {RED, BLUE, GREEN}
```

würde einen Enumerationstyp mit drei Werten festlegen, die intern durch die Werte 0, 1 und 2 repräsentiert würden. Zeigen Sie auch mithilfe einer attributierten Grammatik, wie die Deklaration von Enumerationstypen und der Zugriff auf Enumerationswerte (z.B. Color.RED) zu behandeln sind. Es sind Änderungen im Scanner, im Parser, in der Symbolliste und im Codegenerator nötig.

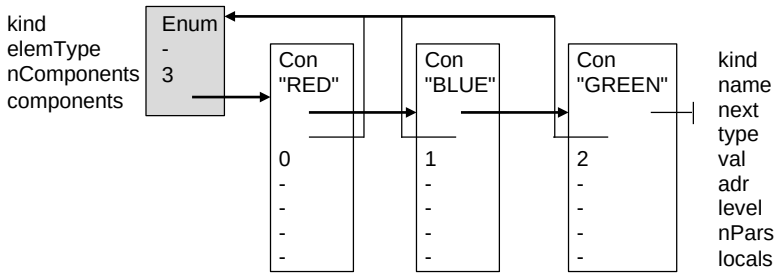
Lösung

Zunächst einmal müssten wir im Scanner und im Parser ein neues Schlüsselwort enum mit einem entsprechenden Tokencode enum_ deklarieren.

Ferner müsste es eine neue Art von Strukturknoten der Art Enum geben. Der Strukturknoten eines Enumerationstyps müsste auf eine Liste von Enumerationskonstanten verweisen. Dazu benennen wir das Feld nFields in nComponents und fields in components um.

```
public class Struct {
   static final int None  = 0, Int   = 1, Char  = 2, Arr   = 3, Class = 4, Enum = 5;
   public int    kind;        // None, Int, Char, Arr, Class, Enum
   public Struct elemType;    // Arr: element type
   public int    nComponents; // Class, Enum: number of fields or constants
   public Obj    components;  // Class, Enum: fields or constants
   ...
}
```

Der Enumerationstyp Color würde dann in der Symbolliste wie folgt dargestellt:



Die Grammatik von MicroJava-Programmen würde wie folgt geändert:

```
Program = "program" ident {ConstDecl | ClassDecl | EnumDecl | | VarDecl} '{' {MethodDecl} '}'.
```

Die Verarbeitung von EnumDecl sähe wie folgt aus:

```
EnumDecl
=  "enum" ident          (. Struct type = new Struct(Struct.Enum);
                            Tab.insert(Obj.Type, t.val, type);
                            Tab.openScope();
                            int val = 0; .)
   "{"
      ident              (. Obj obj = Tab.insert(Obj.Con, t.val, type);
                            obj.val = val; val++; .)
      { "," ident        (. Obj obj = Tab.insert(Obj.Con, t.val, type);
                            obj.val = val; val++; .)
      }
   "}"                   (. type.nComponents = val;
                            type.components = Tab.curScope.locals;
                            Tab.closeScope(); .) .
```

Der Zugriff auf Enumerationskonstanten (z.B. Color.RED) würde in Designator erfolgen:

```
Designator <↑x>
= ident                 (.  Operand x = new Operand(Tab.find(t.val)); .)
  {  "." ident          (.  if (x.kind == Operand.Type) {  // enumeration constant
                              if (x.type.kind != Struct.Enum) error("enumeration type expected");
                              Obj con = Tab.findComponent(t.val, x.type);
                              x.kind = Operand.Con;
                              x.val = con.val;
                            } else if (x.kind == Operand.Local || x.kind == Operand.Static ||
                                       x.kind == Operand.Elem || x.kind == Operand.Fld) {  // field access
                              Code.load(x);
                              if (x.type.kind == Struct.Class) {
                                 Obj comp = Tab.findComponent(t.val, x.type);
                                 x.adr = comp.adr;
                                 x.type = comp.type;
                              } else error("dereferenced object is not a class");
                              x.kind = Operand.Fld;
                            } .)
  | ...
  } .
```

Dazu sind aber noch einige weitere Änderungen nötig. Operand müsste eine neue Operandenart Type haben und der Konstruktor müsste so geändert werden, dass er auch Objekte der Art Obj.Type verarbeiten kann:

```
public class Operand {
   public static final int  Con = 0, Local = 1, Static = 2, Stack = 3, Fld = 4, Elem = 5, Meth = 6, Type = 7, Cond = 8;
   ...
   public Operand (Obj obj) {
      type = obj.type; val = obj.val; adr = obj.adr; kind = Stack; // default
      switch (obj.kind) {
         ...
         case Obj.Type:
            kind = Type; break;
         default:
            Parser.error("wrong kind of identifier"); break;
      }
   }
```

Außerdem müsste man Tab.findField in Tab.findComponent umbenennen, weil ja jetzt nicht nur nach Feldern, sondern auch nach Enumerationskonstanten gesucht wird. Die Funktionsweise dieser Methode bleibt aber gleich.

Bei Typprüfungen müsste man nichts ändern. Bei einer Zuweisung

```
Color c = Color.RED;
```

wäre der Typ der linken und der rechten Seite Color, was durch denselben Strukturknoten ausgedrückt würde.