Parser *vor* Einbau der semantischen Aktionen zum Füllen der Symboltabelle:

```
private static void Type() {
  check(ident);
}


private static void MethodDecl() {
  if (sym == ident) Type();
  else if (sym == voidK) scan();
  else synError("function type or void expected");
  check(ident);
  check(lpar);
  if (sym == ident) FormPars();
  check(rpar);
  ...
}


private static void VarDecl() {
  // erkennt nur Variablen vom Typ Array
  Type();
  check(ident);
  check(lbrack);
  check(rbrack);
  ...
  check(semicolon);
}
```

Parser *nach* Einbau der semantischen Aktionen zum Füllen der Symboltabelle:

```
private static Struct Type() {
  check(ident);
  Obj o = Tab.find(t.string);
  // Objektknoten muss ein bereits existierender Typ sein.
  // Wenn find nichts findet -> noObj -> kein Typ
  if (o.kind != Obj.Type) semError("type expected");
  return o.typ;
}


private static void MethodDecl() {
  Struct type = Tab.noTyp;
  if (sym == ident) type = Type();
  else if (sym == voidK) scan();
  else synError("function type or void expected");
  check(ident);
  Obj meth = Tab.insert(Obj.Meth, t.string, type);
  Tab.openScope();
  check(lpar);
  if (sym == ident)
    meth.level = FormPars();  // Anzahl Parameter
  check(rpar);
  ...
  Tab.closeScope();
}


private static void VarDecl() {
  // erkennt nur Variablen vom Typ Array
  Struct type = Type();
  check(ident);
  String varName = t.string;
  check(lbrack);
  check(rbrack);
  Struct a = new Struct();
  a.kind = Struct.Arr;
  a.elemType = type;
  Tab.insert(Obj.Var, varName, a);
  ...
  check(semicolon);
}
```