8. Tabela simbola

Zadatak 1.

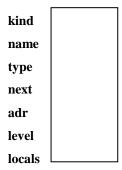
Prikazati izgled tabele simbola u karakterističnim tačkama za priloženi MikroJava program.

```
class P (** 1 **)
final int c = 15;
char r;
(** 2 **)
class Unutra {
  int u[];
(** 3 **)
Unutra k;
{ (** 4 **)
  int m1 (** 5 **) (int w, Unutra d) {
  (** 6 **)
       print(w);
       read(d.u[5]);
       return 0;
  } (** 7 **)
  void main () (** 8 **)
  int a, b; (** 9 **)
       read(a);
       k = new Unutra;
       b = m1(a, k);
  } (** 10 **)
(** 11 **)
```

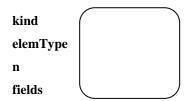
Rešenje:

Tabela simbola za jezik MikroJava će imati 3 vrste čvorova:

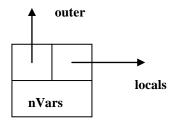
Objektni čvorovi (Object nodes) čuvaju informacije o deklarisanim imenima.



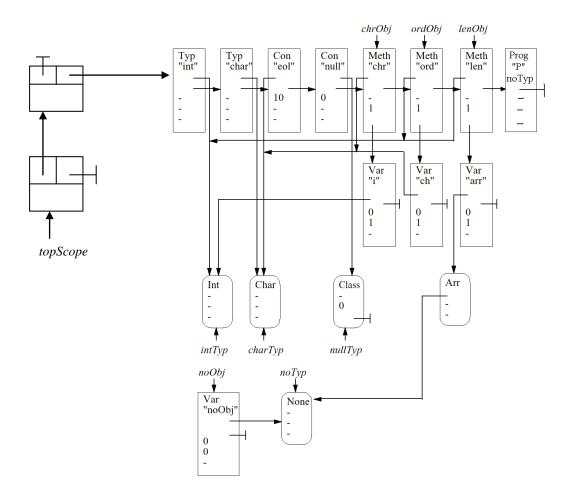
Strukturni čvorovi (Structure nodes) čuvaju informacije o strukturama tipa.



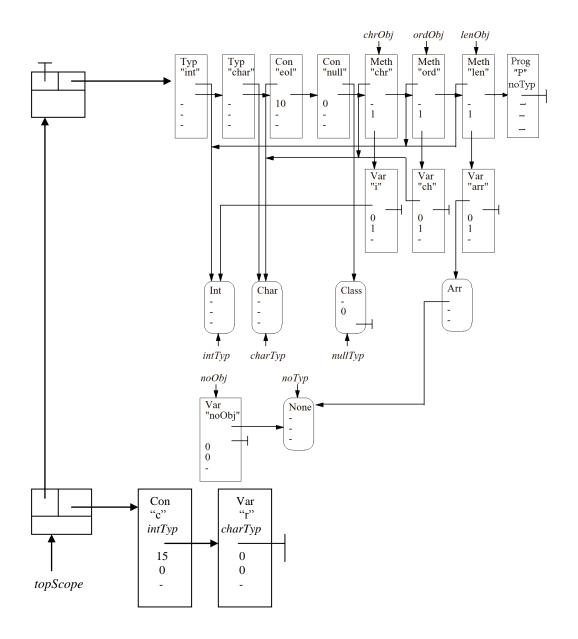
• Čvorovi opsega (Scope nodes) služe za manipulaciju opsezima važenja imena.



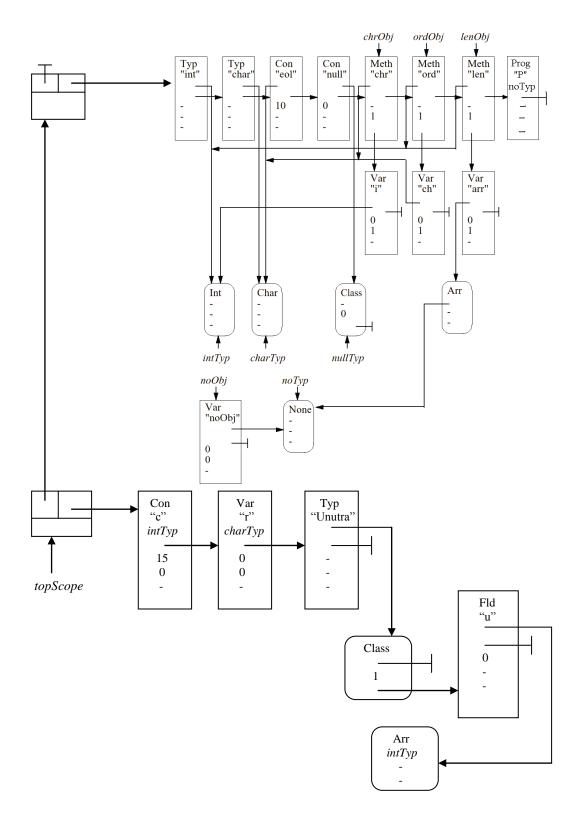
Tačka (** 1 **)



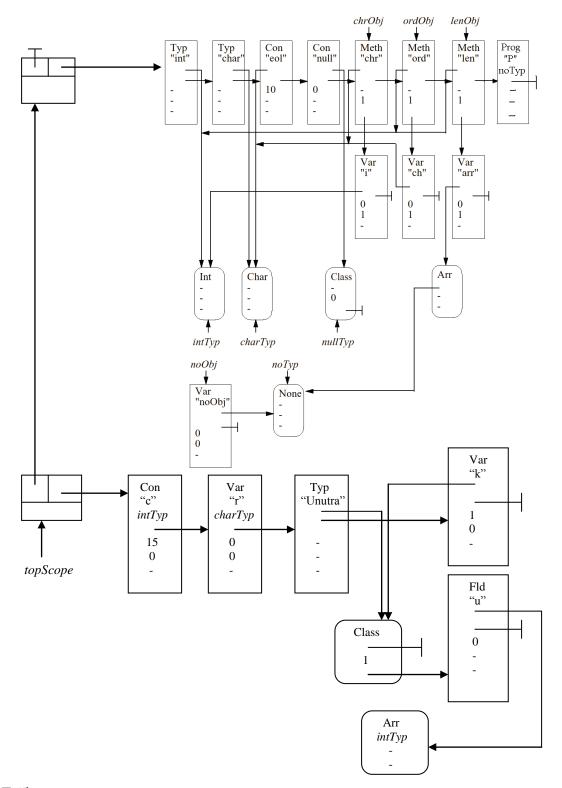
Tačka (** 2 **)



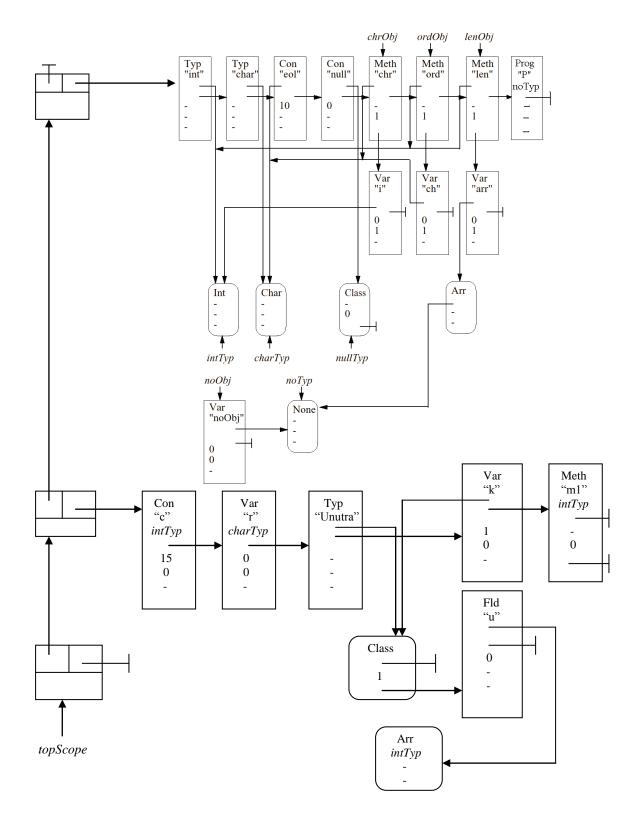
Tačka (** 3 **)



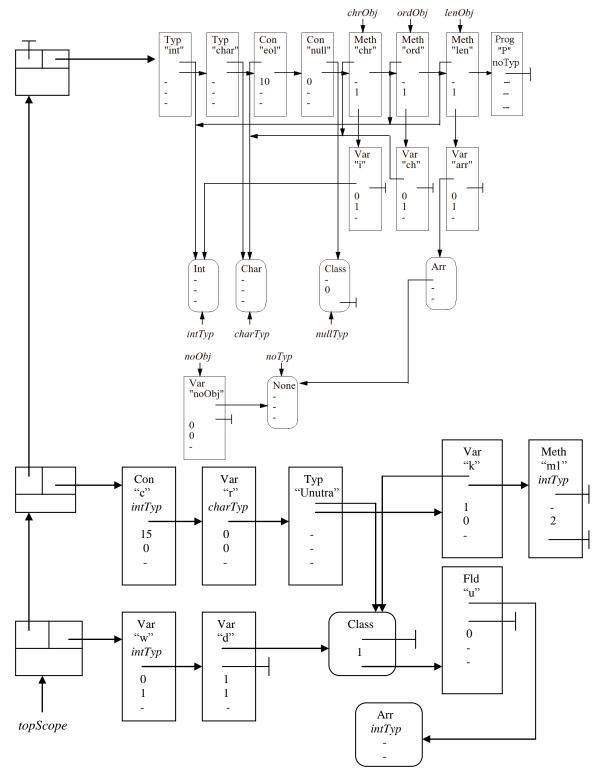
Tačka (** 4 **)



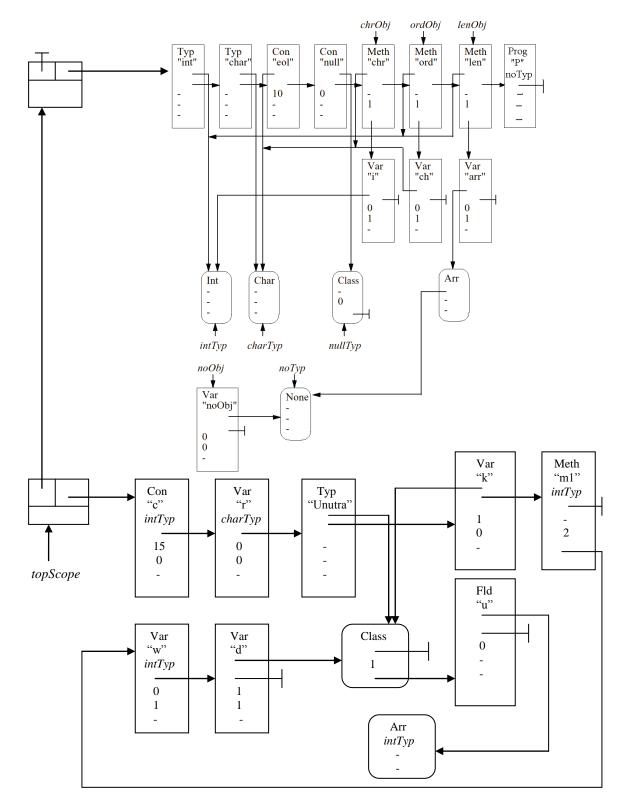
Tačka (** 5 **)



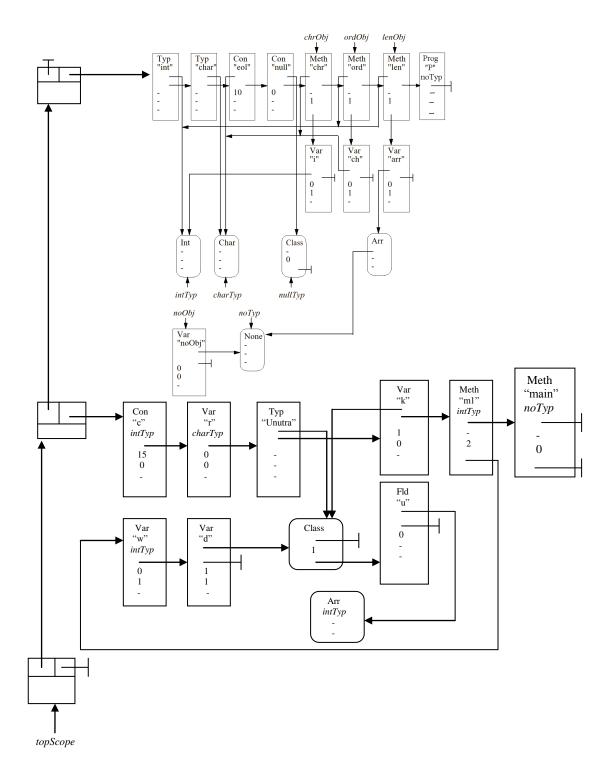
Tačka (** 6 **)



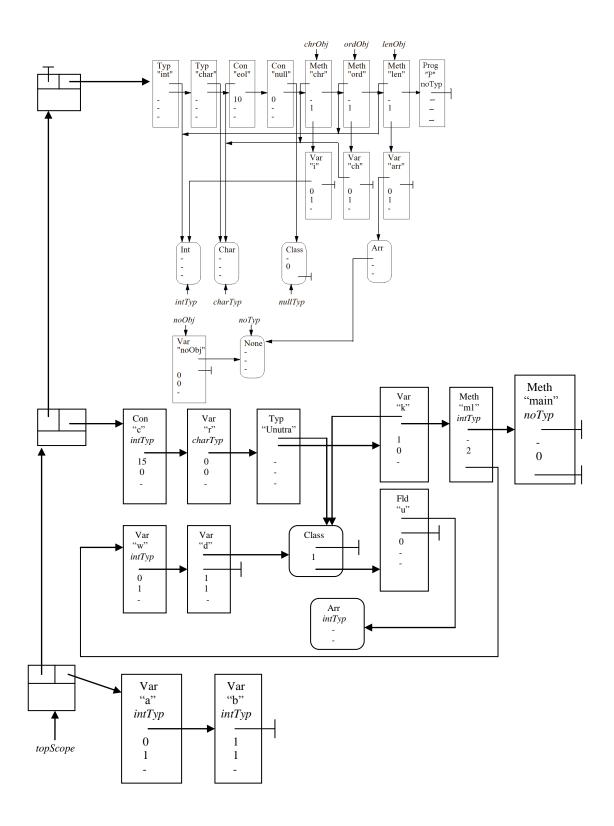
Tačka (** 7 **)



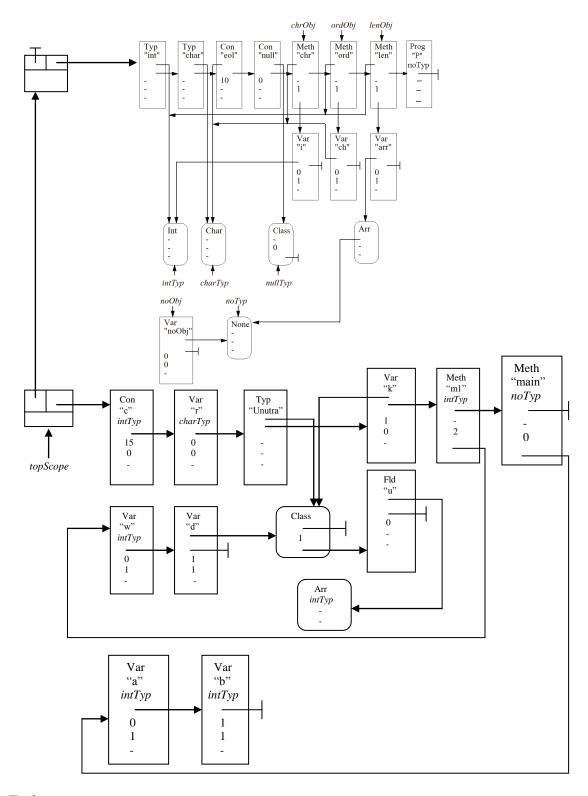
Tačka (** 8 **)



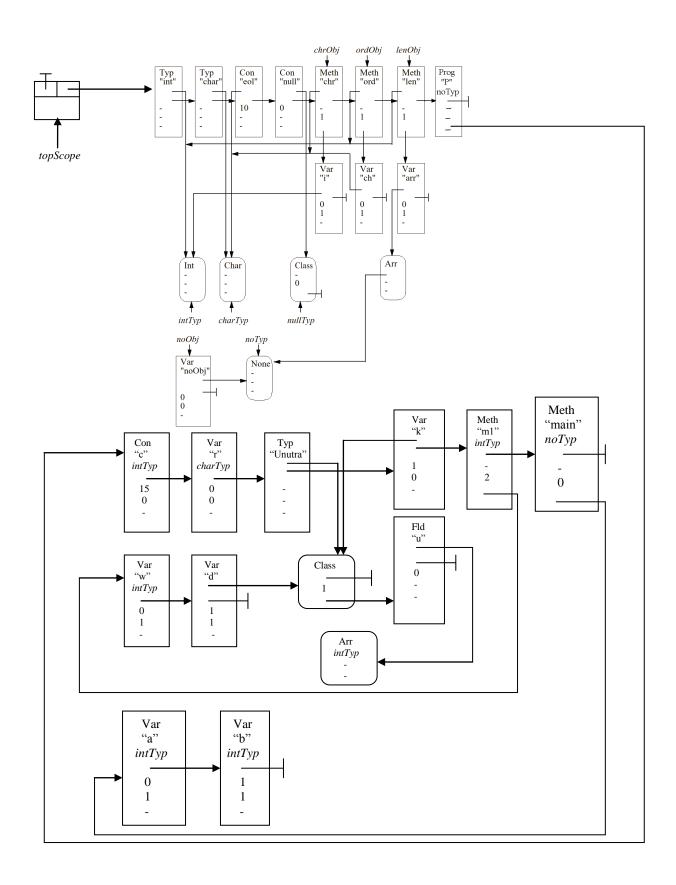
Tačka (** 9 **)



Tačka (** 10 **)



Tačka (** 11 **)



Zadatak 2.

Prikazati izgled tabele simbola u karakterističnim tačkama za priloženi MikroJava program.

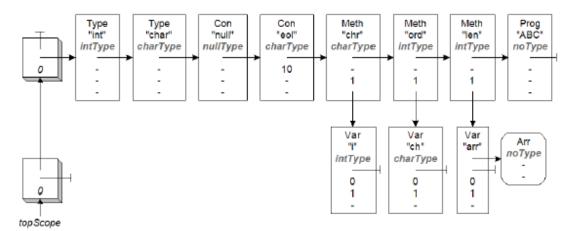
```
class ABC (** 1 **)
char c[];
int max;
char npp;

{
  int put (** 2 **)(int x)
  {
    (** 3 **)
        x++;
        print(x, 5);
        npp = 'C';
        return x;
    }(** 4 **)
}
(** 5 **)
```

Rešenje:

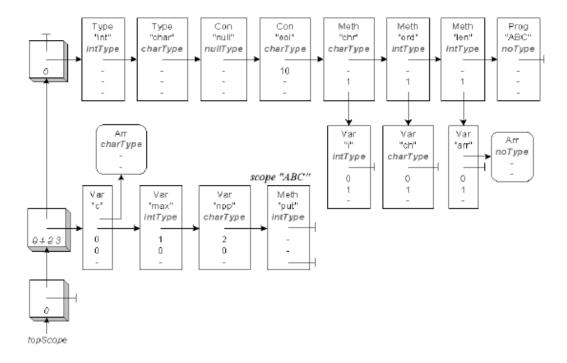
Tačka (** 1 **)

U glavnom opsegu se nalaze predefinisani simboli("int", "char",...), a pošto se tačka 1 nalazi iza deklaracije glavne klase u tabeli simbola se nalazi i ime glavne klase tj. programa.



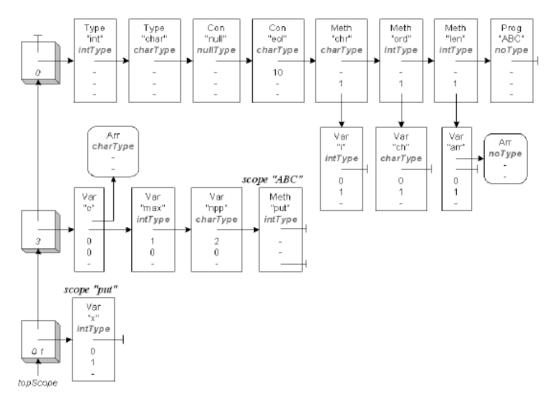
Tačka (** 2 **)

Obrađene su deklaracije globalnih promenljivih (niza karaktera c, celobrojne promenljive max i karaktera npp) i početak definicije metode put. Simboli se ubacuju u opseg koji pripada glavnoj klasi tj. u opseg "ABC".



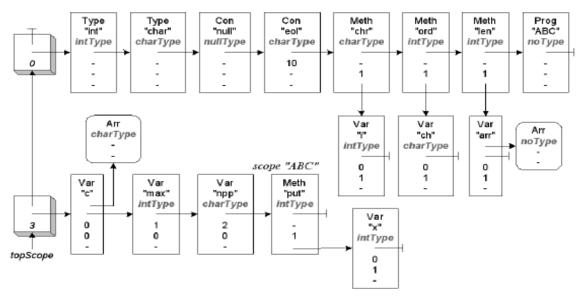
Tačka (** 3 **)

Obrađeni su argumenti metode put i simbol x je unet u tabelu simbola. Pošto je x vidljiv samo u metodi put, simbol x se stavlja u opseg koji pripada metodi put.



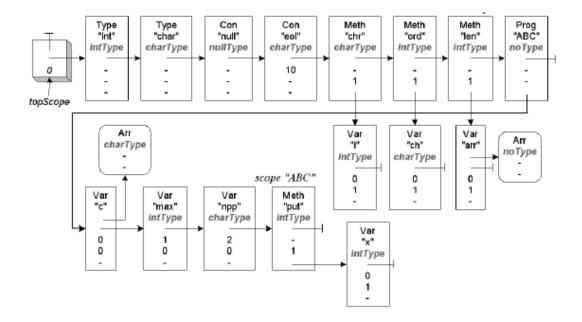
Tačka (** 4 **)

Parser je izašao iz opsega metode put i odgovarajući opseg je skinut sa steka.



Tačka (** 5 **)

Parser je završio sa radom, skinuo je poslednji opseg sa steka i dodelio ga Object čvoru glavne klase.



Zadatak 3

Prodiskutovati prednosti i mane organizovanja tabele simbola

- a) kao jedinstvene tabele
- b) uz postojanje posebne tabele za svaki opseg ako se tabele realizuju uz upotrebu ili heš funkcije ili leksičkog stabla.

Rešenje:

Postoje 4 varijante: jedinstvena tabela - heš funkcija,

jedinstvena tabela - leksičko stablo,

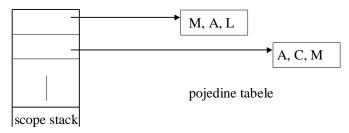
lokalna tabela - heš funkcija,

lokalna tabela - leksičko stablo.

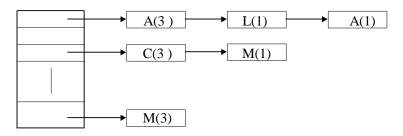
Razmotrićemo pretraživanje tabela (funkcija LookUp), otvaranje i zatvaranje leksičkog opsega funkcije OpenScope i CloseScope na primeru:

```
{ int H, A, L;
...
{float X, Y;
...
}
{char A, C, M;
...
/* trenutna pozicija*/
}
```

b) individualne tabele

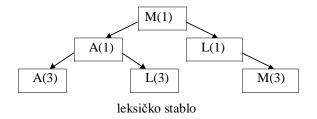


- pri otvaranju opsega na vrh steka ide tabela za taj opseg,
- pretraživanje: nelokalna imena ⇒ više tabela (mana),
- zatvaranje opsega ⇒ skine se tabela sa vrha steka,
- memorijski zahtevi: bolja realizacija sa stablima zbog fiksnog overheada heš tabela
- a) jedinstvena tabela



heš tabela

- OpenScope: dodeli se opsegu jedinstven broj i čuva se uz ime (više memorije),
- LookUp: traži se dok se ne naleti na prvo A (efikasno),
- Insert: stavlja se na početak lanca kolizije
- CloseScope: izbrišu se imena sa početka svih lanaca (efikasno)



- OpenScope: kao gore,
- Insert: uvek se dodaju listovi, pa pri look up-u ne smemo se zadovoljiti prvim A nego treba ići do lista (neefikasno),
- CloseScope: mora se pretraživati celo stablo (neefikasno),
- rezime: pogodnije kombinacije heš jedinstvena tabela i stablo lokalne tabele.

Kod višeprolaznih prevodilaca gotovo isključivo se koriste lokalne tabele jer pri CloseScope ne smemo jednostavno uništiti informacije iz tabele (trebaju nam za druge prolaze).

Nasleđivanje i polimorfizam

Zadatak 4

Dat je sledeći program na programskom jeziku Mikrojava. Prikazati izgled tabele simbola u trenucima prevođenja koji su naznačeni u programu. Mikrojava podržava nasleđivanje klasa i polimorfizam.

```
class OOMJ (* T1 *)
     class TCalc (* T2 *) { // TaxCalculator
          double tax; // [0,1] interval (* T3 *)
                double calcPrice(double cost) { return cost*(1+tax); }
                int check(double v) { return d>0; }
           } (* T4 *)
     (* T5 *)
     class ATCalc (* T6 *) extends TCalc {
          // Tax with amortization
          double a;
           { (* T7 *)
                // @Overriden
                double calcPrice(double c) { return (cost*a)*(1*tax); }
           (* T8 *)
     } (* T9 *)
{
     void main()
          TCalc c1, c2;
          double cost; (* T10 *)
     {
          cost = 1000;
          c1 = new Tcalc; c1.tax = (0.09);
          c2 = new ATCalc; c2.tax = 0.18; c2.a = 0.2;
          print(c1.calcPrice(cost));
          print(c2.calcPrice(cost));
     }
(* T11 *)
```

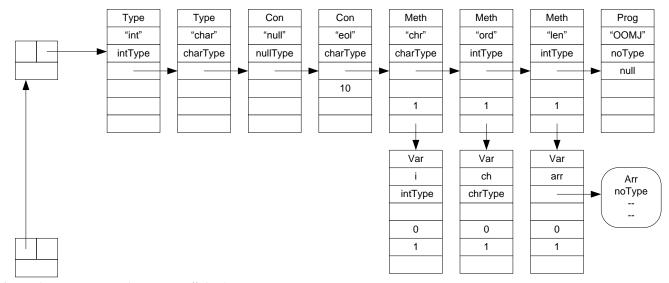


Figure 1 Izgled tabele simbola u tački T1.

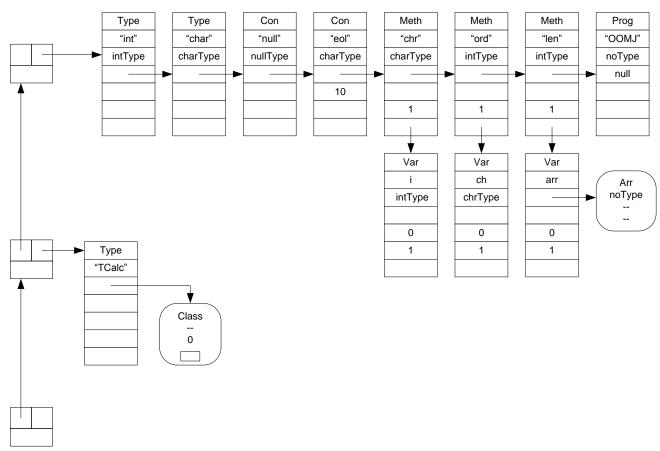


Figure 2 Izgled tabele simbola u tački T2.

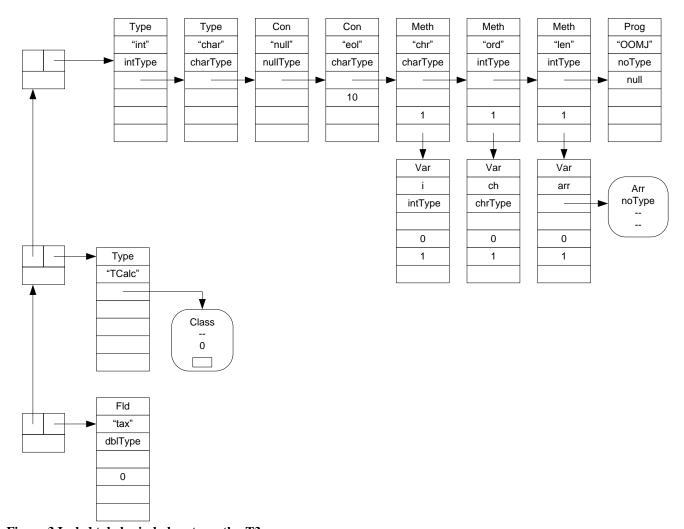


Figure 3 Izgled tabele simbola u trenutku T3.

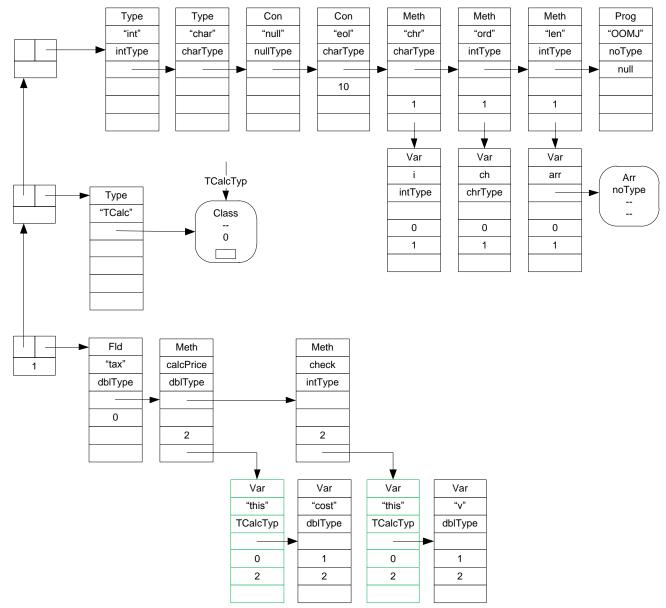


Figure 4 Izgled tabele simbola u tački T4.

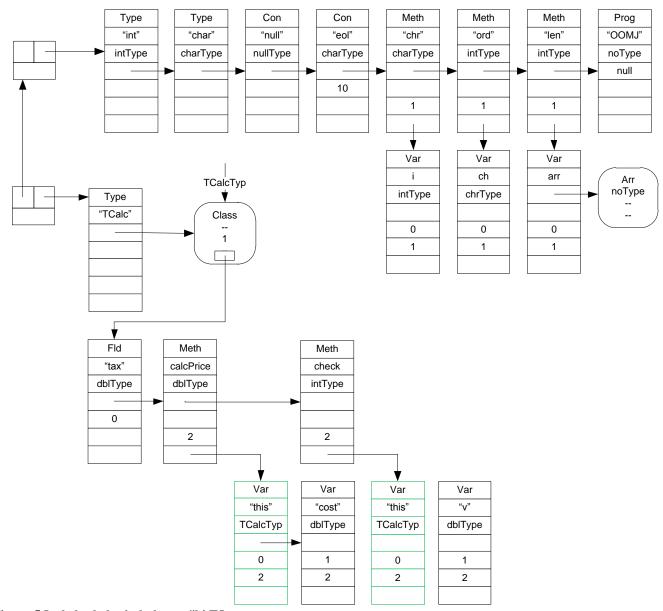
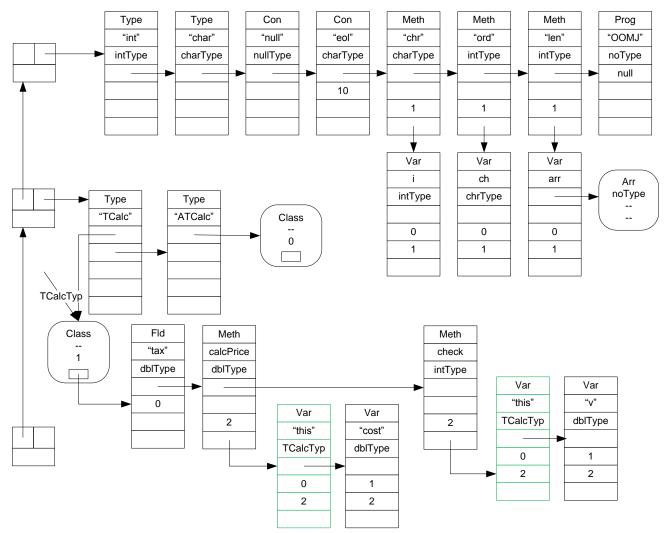


Figure 5 Izgled tabele simbola u tački T5.



Izgled tabele simbola

Figure 6 Izgled tabele simbola u tački T6.

Izgled tabele simbola:

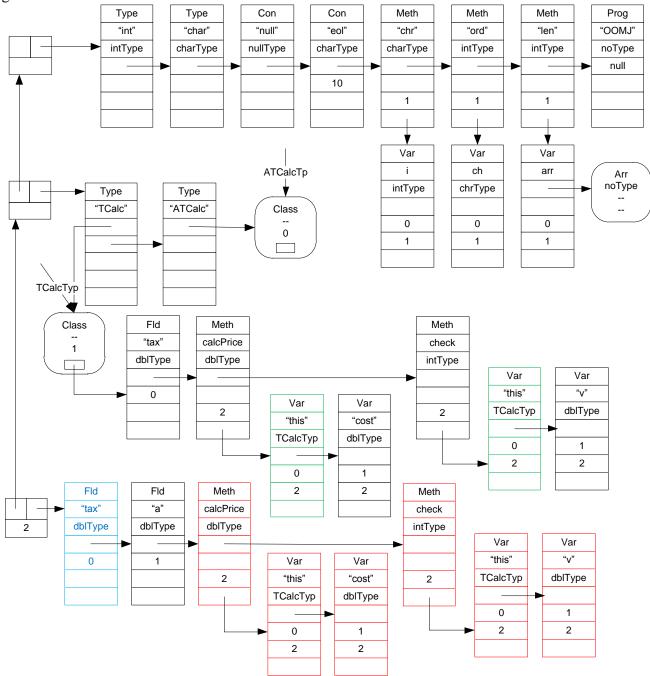


Figure 7 Izgled tabele simbola u tački T7.

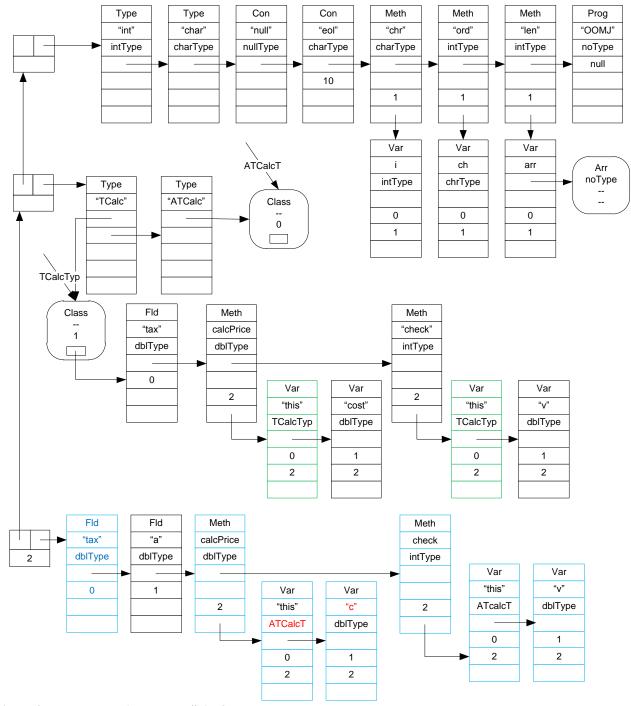


Figure 8 Izgled tabele simbola u tački T8.

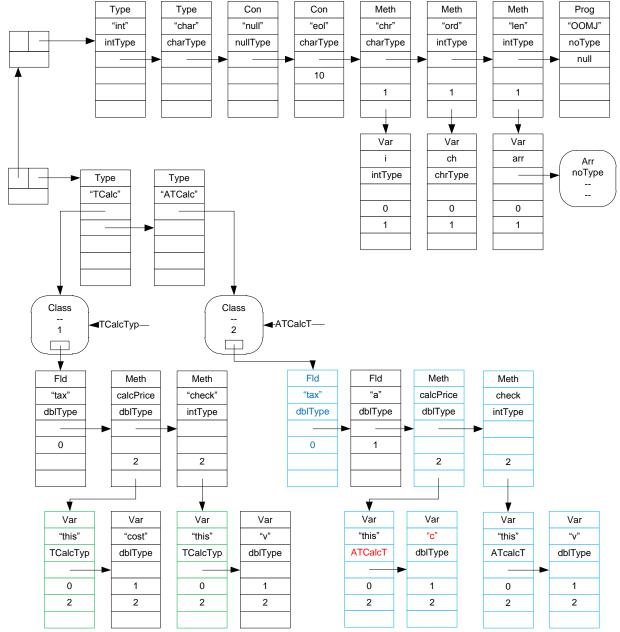


Figure 9 Izgled tabele simbola u tački T9.

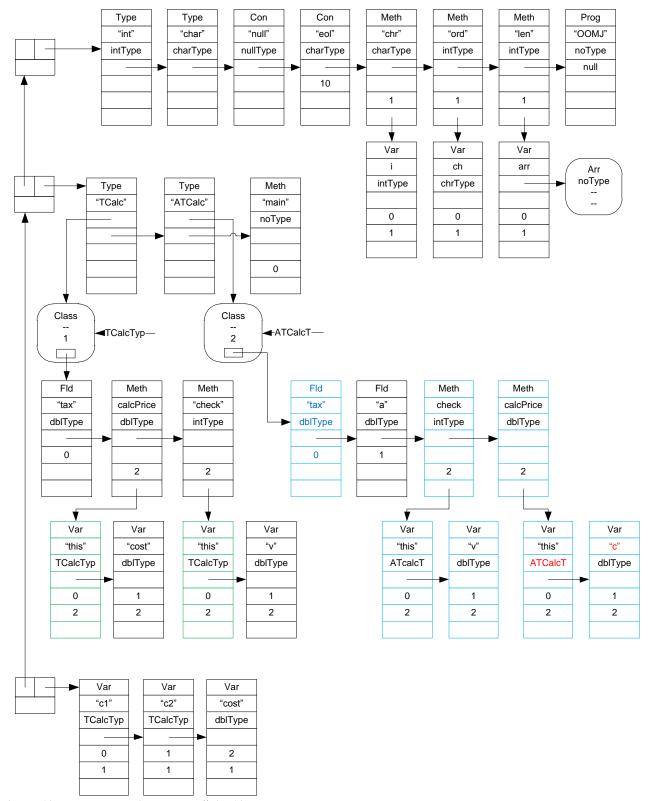


Figure 10 Izgled tabele simbola u tački T10.

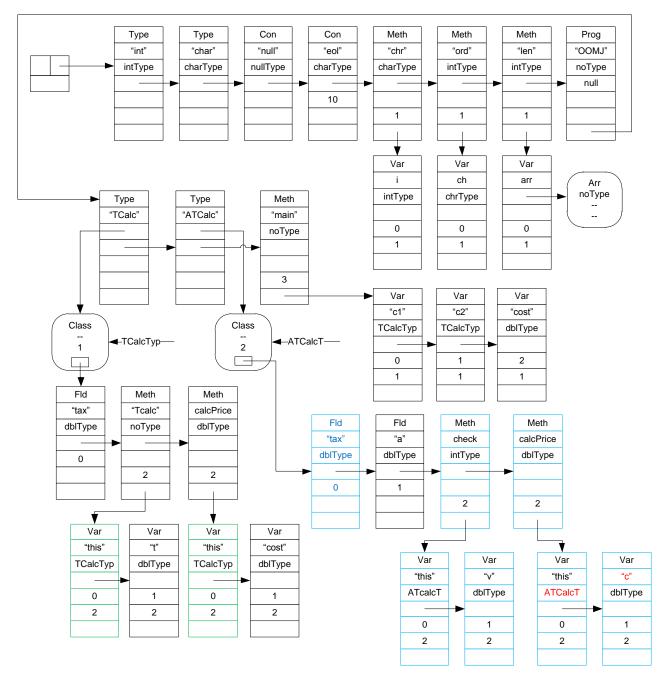


Figure 11 Izgled tabele simbola u tački T11.