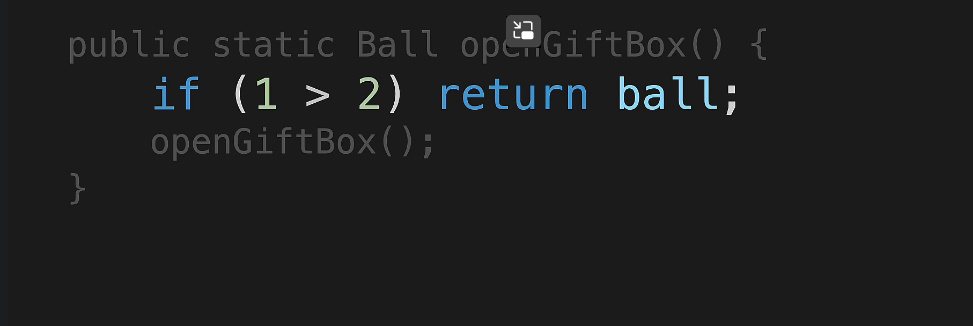
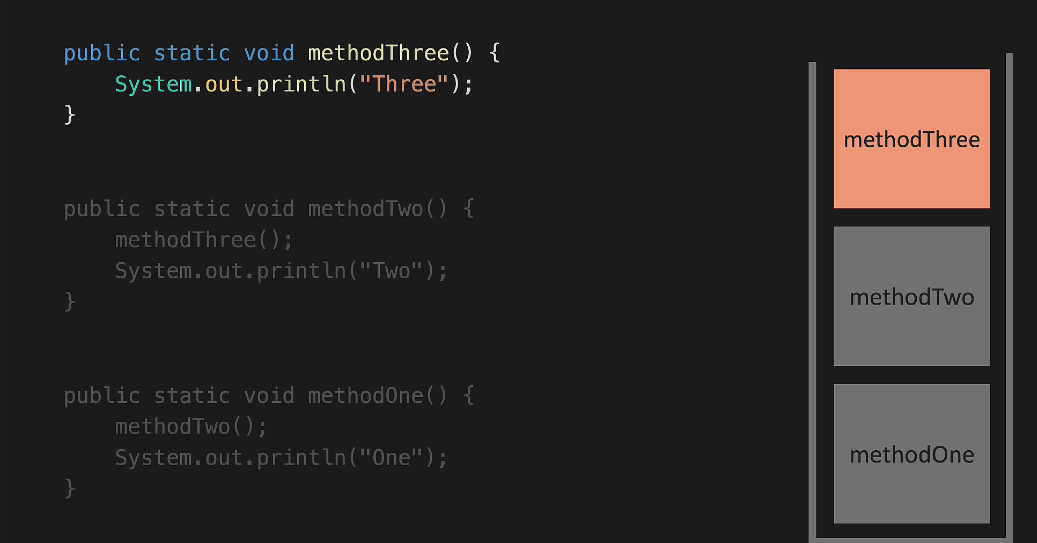
* Cand o metoda se apeleaza de prea multe ori, adica recursivitatea e infinita sau de prea multe ori, apare o exceptie **StackOverFlow**

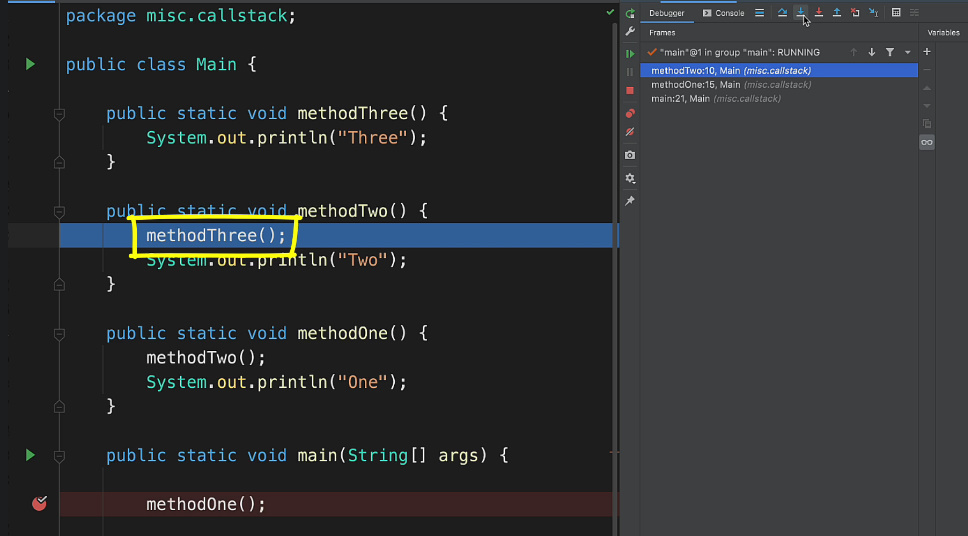


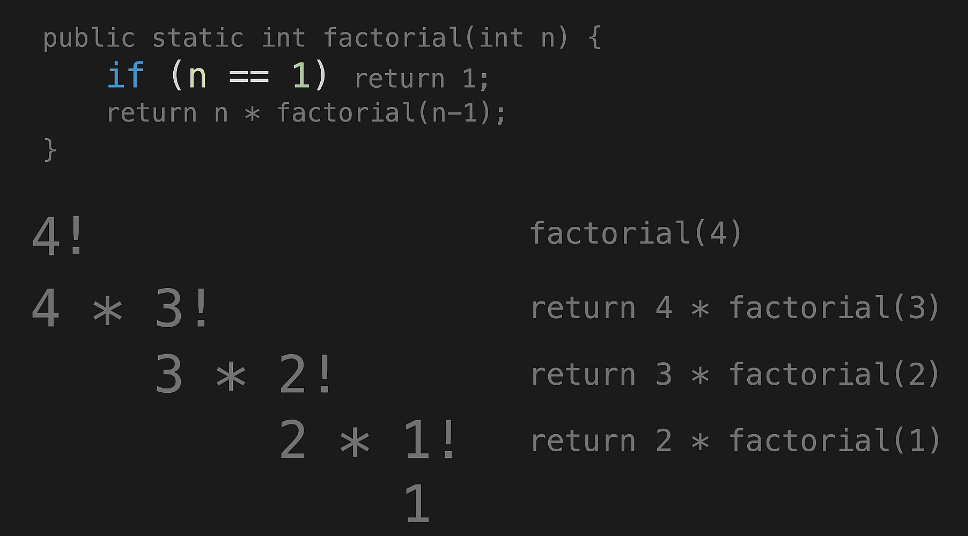
**Call Stack**

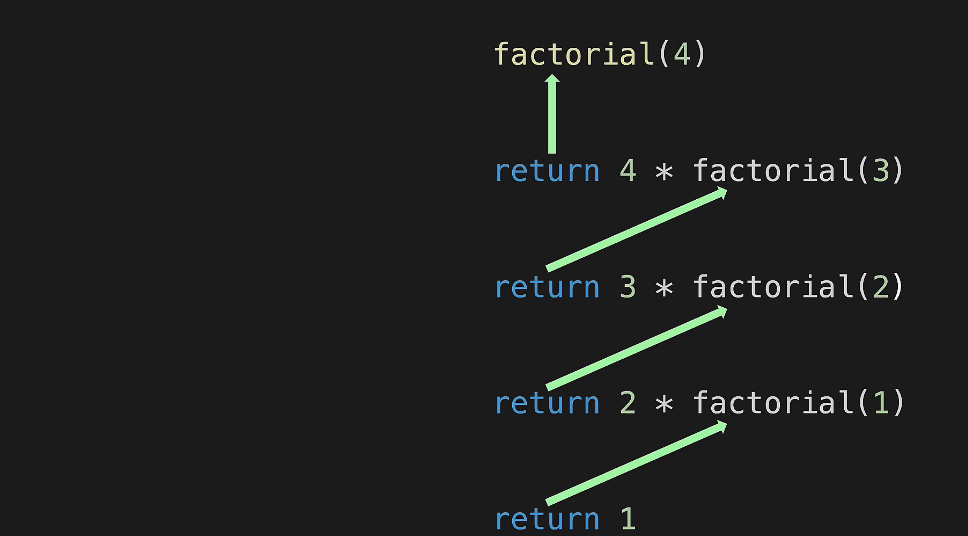
* Apelurile de metoda sunt pastrate in Stack
* Deci, logic ca intai e executata metoda care se afla in varf, adica prima, apoi urmatoarea si tot asa.
* De ex:



daca vom da methodOne(), intai se va executa methodThree, apoi methodTwo si abea apoi methodOne. Exact asa au si fost puse in stack.

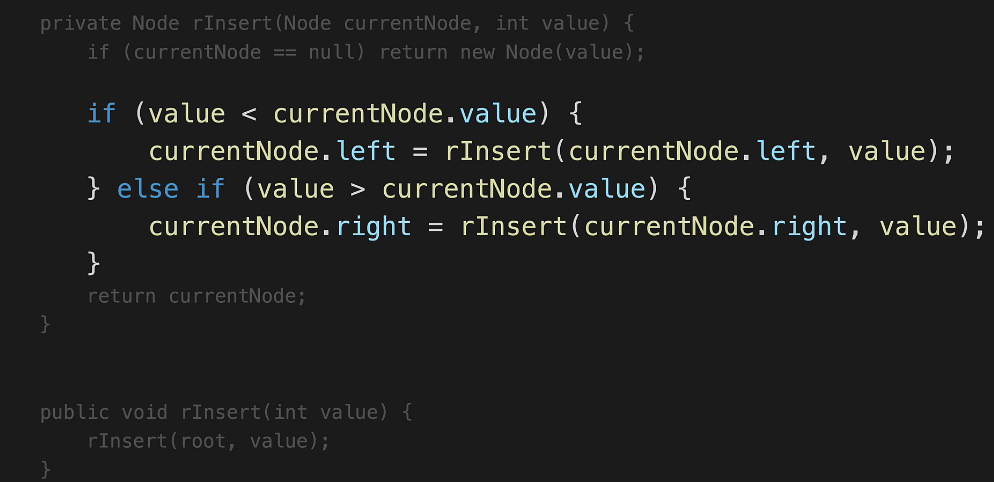


* Cand o metoda se apeleaza tot pe ea, adica la recursie, apelul ei tot e pus in stack
* 

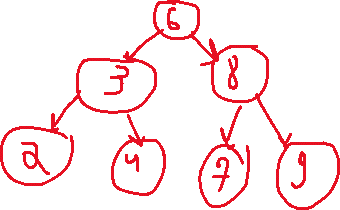
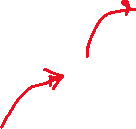


**recursive BST**

* contains()
* public boolean contains(int value){  
   return contains(root,value);  
  }  
  private boolean contains(Node temp,int value){  
   if(temp == null)  
   return false;  
   if(temp.value == value)  
   return true;  
   else if(value > temp.value)  
   return contains(temp.right,value);  
   else  
   return contains(temp.left,value);  
  }
* insert(value)



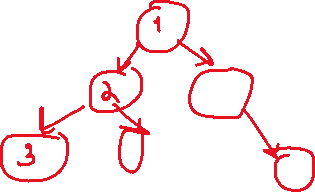
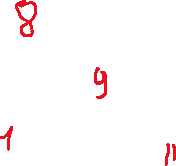
insert(1)



* **delete()**

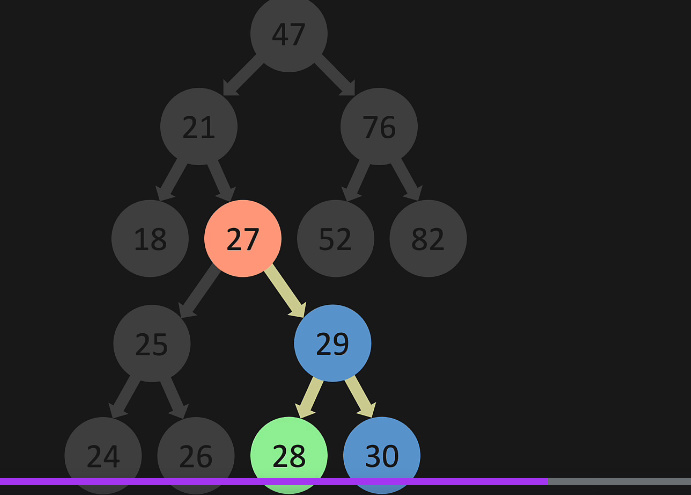
Este cam dificil de a sterge un nod cand acesta are atat un copil stang cat si drept.

De ex, daca are doar un copil, e usor:



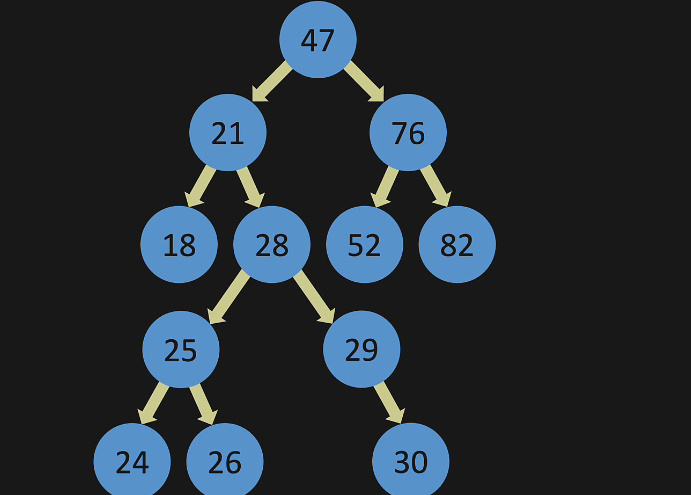
sa zicem ca stergem nodul 9. In asa caz, valoarea nodului 11 pur si simplu va fi copiata in nodul 9 si nodul 9, care acum are valoarea nodului 11 in el, va pierde referinta catre nodul 11. Dar, daca am sterge nodul 2, pai aici deja cam e problema, caci avem si nodul 3 si 1.

Fie arborele binar:



Vrem sa il stergem pe 27, dar raman 25,26 si 24

Solutia este urmatoarea:





Deci, 27 a fost eliminat, dar el are copii drept si stang. Asa dar, solutia este simpla:

1. Trebuie sa punem vreun copil de al lui 27 in locul lui 27.
2. Cea mai buna varianta este de a lua valoarea celei mai din stanga frunze a copilului din dreapta si de a o inlocui cu valoarea nodului 27. Asa, valorea nodului 28 din copil e pusa in fostul parinte al parintelui sau. Daca copilul din dreapta are copiii si inca nepoti, ei bine luam cea mai din stanga frunza a celui mai din stanga copil. Acea frunza la sigur e mai mica ca copilul drept a lui 27, adica parintele acestui nod din dreapta, dar si la sigur mai mare ca orice noduri a copilului din stanga a lui 27. Am putea lua si cea mai din dreapta frunza a copilului din stanga.

Atentie! Nu mutam nodul 28! Pur si simplu copiem valoarea lui 28 in 27, apoi stergem referinta parintelui lui 28 catre el.

**Varianta liniara:**

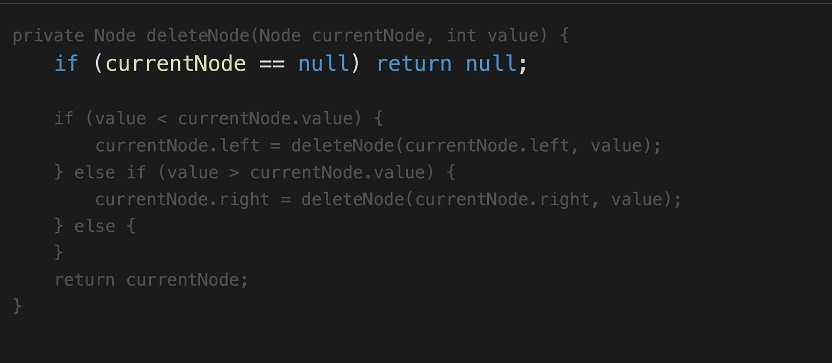
public boolean remove(int value){  
 if(root==null)  
 return false;  
 Node temp = root;  
 Node parent = root; //va fi nodul parinte celui ce trebuie sters  
 while(temp!=null){ //ajungem la nodul de sters  
 if(value < temp.value){  
 if(temp.left != null){  
 parent = temp;  
 temp = temp.left;  
 }  
 else return false;  
 }  
 else if(value > temp.value){  
 if(temp.right != null){  
 parent = temp;  
 temp = temp.right;  
 }  
 else return false;  
 }  
 else{  
 if(temp.right==null && temp.left == null) { //daca nodul de sters nu are copii, e frunza adica  
 if(parent.left==temp)  
 parent.left = null;  
 else if(parent.right==temp)  
 parent.right= null;  
 }  
 if(temp.right!=null && temp.left==null){ //daca nodul de sters are un copil drept  
 if(parent.left==temp) //verificam daca nodul de sters e copil stang al parintelui sau, ca sa nu iteram iar prin arbore  
 parent.left = temp.right;  
 else if(parent.right ==temp) //verificam daca nodul de sters e copil drept al parintelui sau, ca sa nu iteram iar prin arbore  
 parent.right = temp.right;  
 temp.right = null;  
 return true;  
 }  
 if(temp.right==null && temp.left!=null){ //daca nodul de sters are un copil stang  
 if(parent.left==temp) //verificam daca nodul de sters e copil stang al parintelui sau, ca sa nu iteram iar prin arbore  
 parent.left = temp.left;  
 else if(parent.right==temp) //verificam daca nodul de sters e copil drept al parintelui sau, ca sa nu iteram iar prin arbore  
 parent.right = temp.left;  
 temp.left = null;  
 return true;  
 }  
 if(temp.right!=null && temp.left!=null){ // daca nodul de sters are 2 copii, si stang si drept  
 Node leaf = temp.right;  
 Node parentLeaf = leaf; //parintele referinta la nodul cel mai din stanga al copilului drept  
  
 if(leaf.left==null){ //daca copilul drept al nodului de sters nu are copil stang, insemna ca nu avem ce mai cauta  
 leaf.left = temp.left; //deci,noul nod trebuie sa aiba referinta la copilul stang al nodului sters, sa nu o pierdem  
 parent.right = temp.right;  
 return true;  
 }  
  
 while(leaf.left!=null){ //daca copilul drept al nodului de sters are si el un copil stang, ajungem la cel mai din stanga, adica la frunza  
 parentLeaf = leaf;  
 leaf = leaf.left;  
 }  
 parentLeaf.left = null;  
 temp.value = leaf.value;  
  
 }  
 return false;  
 }  
 }  
 return false;  
}

Deci, avem 4 cazuri de baza cand ajungem la nodul ce trebuie sters:

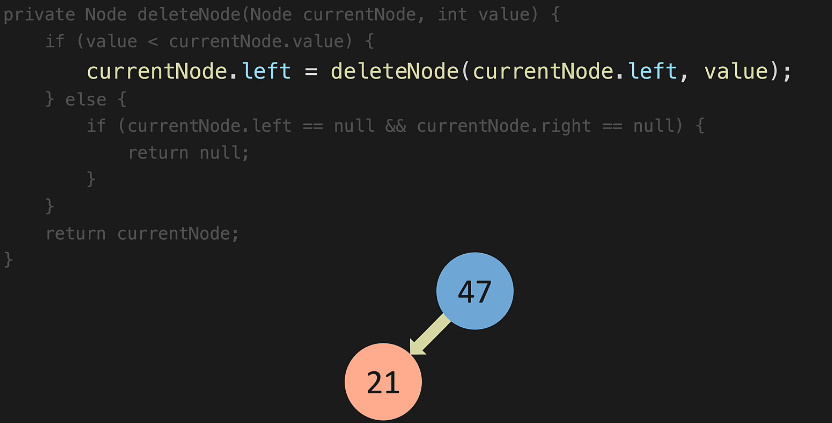
1. Nodul de sters nu are nici copil stang, nici drept. In asa caz, nodul sau parinte pur si simplu va sterge referinta la el.
2. Nodul de sters are doar copil stang. In asa caz, nodul parinte al nodului de sters va avea o referinta la copilul stang al nodului de sters(vom decide inca odata daca nodul de sters e copil drept sau stang al parintelui sau, sa nu mai iteram iar, de asta si pastram o variabila cu referinta la parintele nodului de sters)
3. Nodul de sters are doar copil drept. In asa caz, nodul parinte al nodului de sters va avea o referinta la copilul drept al nodului de sters(vom decide inca odata daca nodul de sters e copil drept sau stang al parintelui sau, sa nu mai iteram iar, de asta si pastram o variabila cu referinta la parintele nodului de sters)
4. Nodul de sters are si copil drept si stang. Deci, vom cauta cel mai din stanga nod al copilului nodului drept al nodului de sters. Daca nu are, in asa caz nodul parinte al nodului de sters va avea drept copil drept copilul drept al nodului de sters, si in plus copilul drept adaugat va avea drept copil stang copilul stang al nodului sters.Daca totusi copilul drept al nodului de sters are copil stang, nodul de sters va copia doar valoarea(nu modificam copilul nodului parinte al nodului de sters referinta!!!!) nodului gasit, si apoi parintele nodului gasit va sterge referinta la el, de asta aici avem nevoie si de o referinta la nodul copil al frunzei gasite.

**Recursive METHOD**

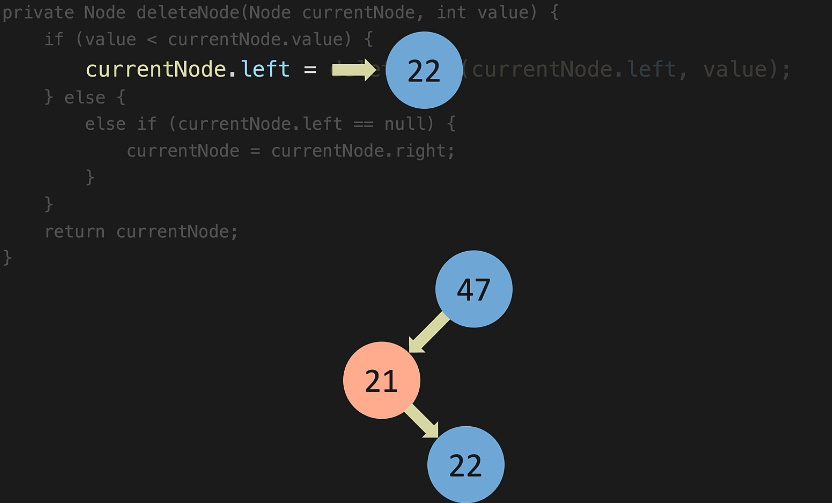




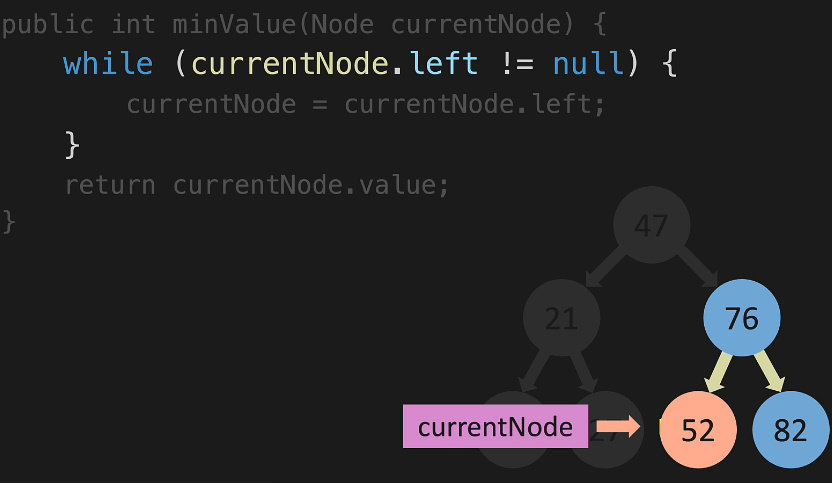
Situatia cand nodul de sters nu are copii



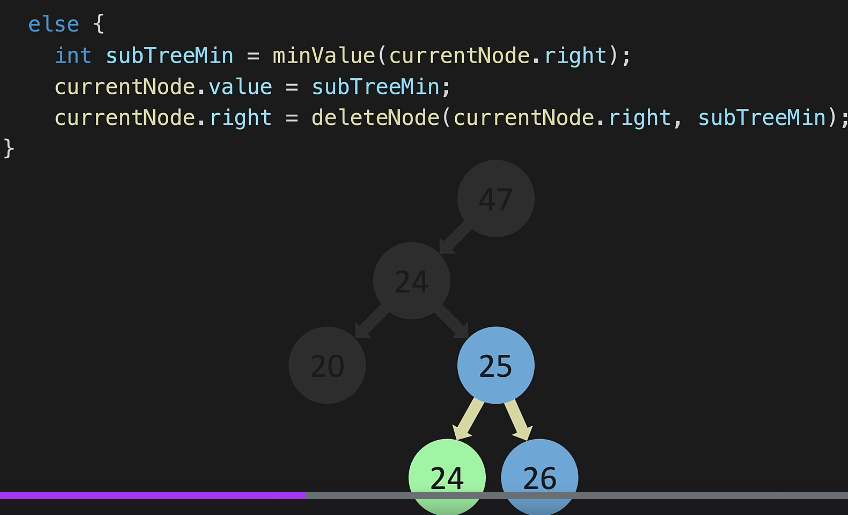
Situatia cand nodul de sters are un copil stang



determinarea valorii minime intr-un subarbore:



Cand nodul de sters are copil stang si drept



Total code:

