

Glava 4: Apstrakcija

- ❖ Modularnost i enkapsulacija
- ❖ Sa proceduralnog na OOP jezik
- ❖ Klasa kao realizacija apstraktnog tipa podataka
- ❖ Klasa kao realizacija apstraktne strukture podataka
- ❖ Klasa kao realizacija apstrakcije
- ❖ Klasa kao realizacija softverske mašine



Modularnost i enkapsulacija

- ❖ Rešavamo sledeći problem: za dati niz znakova koji sadrži samo otvorene i zatvorene zagrade, potrebno je utvrditi da li su zagrade propisno uparene i ugneždene i ako jesu, koja otvorena zagrada odgovara kojoj zatvorenoj zagradi
- ❖ Rešenje se zasniva na korišćenju *steka* (*stack*):
 - stek je inicijalno prazan; obrađujemo znakove u ulaznom nizu redom, jedan po jedan;
 - po nailasku na '(', na vrh steka smeštamo poziciju te otvorene zagrade (operacija *push*);
 - po nailasku na ')', sa vrha steka skidamo poziciju njoj odgovarajuće otvorene zagrade (operacija *pop*);
 - ako je stek prazan pri nailasku na zatvorenu zgradu ili ako je ostao neprazan nakon kraja ulaznog niza, zagrade nisu dobro uparene
- ❖ Potreban nam je stek:
 - linearna struktura elemenata (svaki element ima najviše po jednog prethodnika i sledbenika)
 - operacije *push* i *pop* sa LIFO (*last-in-first-out*) protokolom
- ❖ Složen program čiji je ovo samo mali deo *dekomponujemo* na logičke celine - *module*; u proceduralnom programiranju, ovakav stek bismo implementirali u jednom modulu
- ❖ *Dekompozicija* (*decomposition*) je jedno od osnovnih oruđa (pored apstrakcije) koje čovek koristi u rešavanju kompleksnosti (softvera): podela složenog problema / sistema na delove i odvojeno napadanje i rešavanje tih delova