

Projektovanje softvera

Stanje



Stanje (1)

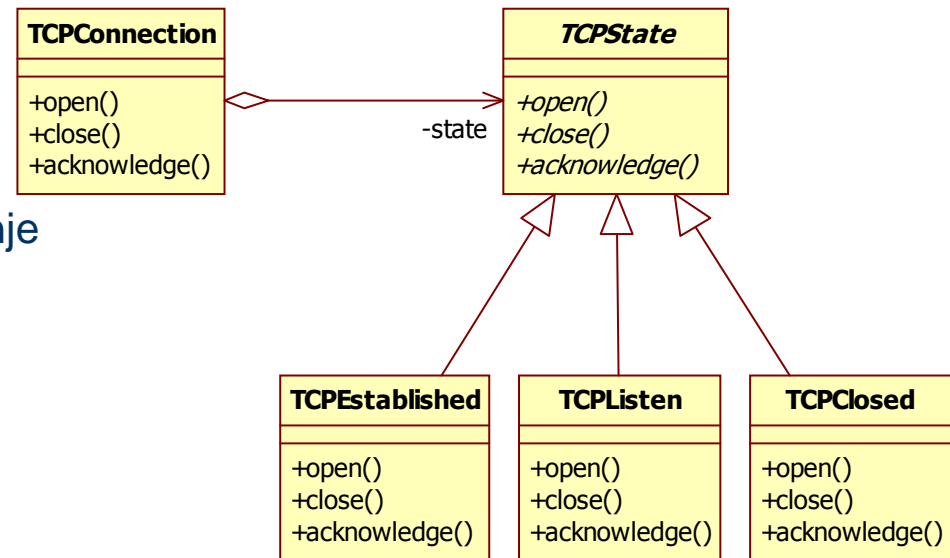
- Ime i klasifikacija:
 - Stanje (engl. *State*) – objektni uzorak ponašanja
- Namena:
 - omogućava objektu da pouzdano menja svoje ponašanje kada se menja njegovo unutrašnje stanje
 - izgleda kao da objekat menja svoju klasu
- Drugo ime:
 - Objekti za stanja (engl. *Objects for States*)

Stanje (2)

- Motivacija - primer vezan za mrežni TCP protokol
 - vezu opisuje `TCPConnection` klasa
 - moguća stanja veze: *Established*, *Listening*, *Closed*
 - `TCPConnection` objekat odgovara na zahteve u zavisnosti od stanja
 - npr. efekat `open()` zahteva se razlikuje u stanju *Closed* i *Established*
 - uzorak *Stanje*
 - opisuje kako ponašanje `TCPConnection` objekta zavisi od stanja
 - ključna ideja: uvođenje apstraktne klase `TCPState`
 - klasa predstavlja stanja veze
 - klasa `TCPState` deklariše zajednički interfejs za klase stanja
 - potklase `TCPState` realizuju specifično ponašanje pojedinih stanja

Stanje (3)

- Motivacija (nastavak):
 - objekat `TCPConnection`
 - sadrži objekat stanja (objekat potklase `TCPState`) koji reprezentuje tekuće stanje veze
 - prosleđuje objektu stanja zahteve koji su zavisni od tekućeg stanja
 - kada konekcija menja stanje zamenjuje objekat stanja koji koristi

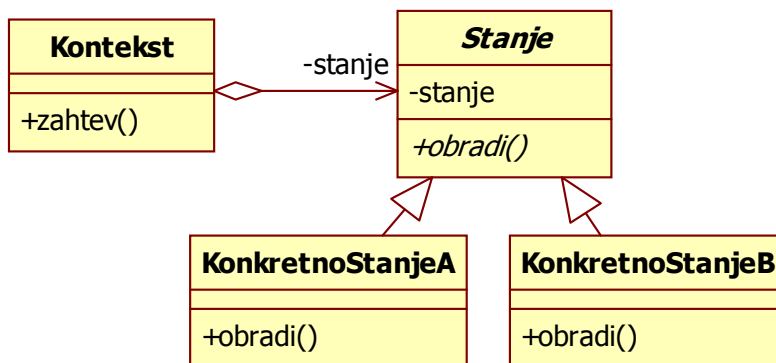


Stanje (4)

- Primenljivost:
 - kada ponašanje objekta zavisi od stanja i mora se menjati u vreme izvršavanja
 - kada operacije imaju velike uslovne naredbe sa više grana, čije izvršenje zavisi od stanja objekta
 - često više operacija sadrži istu uslovnu naredbu
 - uzorak *Stanje* pravi od svake grane posebnu klasu koja određuje ponašanje operacije u datom stanju

Stanje (5)

- **Struktura:**



- **Saradnja:**

- kontekst prosleđuje zahteve koji su zavisni od stanja tekućem objektu konkretnog stanja
- kontekst može da prosledi konkretnom objektu stanja referencu na sebe, što omogućava objektu stanja da pristupi kontekstu (npr. da zahteva promenu stanja)
- kontekst je primarni interfejs za klijente, klijenti ga mogu konfigurisati objektima stanja; nakon toga klijenti ne moraju da interaguju direktno sa objektima stanja

- **Učesnici:**

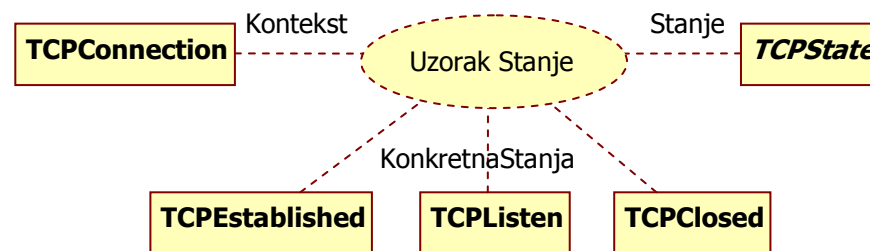
- Kontekst (klasa `TCPConnection`)
 - definiše interfejs od interesa za klijente
 - održava primerak `KonkretnoStanjeX`
- Stanje (klasa `TCPState`)
 - definiše interfejs za pojedina stanja
 - kapsulacija ponašanja pridruženog stanju objekta klase `Kontekst`
- `KonkretnoStanjeX` (klasa `TCPEstablished`, `TCPListen`)
 - implementira interfejs stanja

Stanje (6)

- Posledice:
 - dobra kapsulacija ponašanja specifičnog za stanje
 - jasno razdvajanje ponašanja u različitim stanjima
 - jednostavno dodavanje novih stanja definisanjem novih potklasa stanja
 - nefleksibilna alternativa su uslovni iskazi rasuti svuda po kodu konteksta
 - robusno rešenje - prelazi između stanja su eksplicitni i atomični
 - kada kontekst samostalno definiše svoje stanje vrednostima atributa
 - prelazi između stanja nemaju eksplicitnu reprezentaciju
 - predstavljaju se dodelama vrednosti atributima
 - objekti stanja štite objekat konteksta od nekonzistentnih stanja
 - prelazi su atomični iz perspektive konteksta
 - prelaz se svodi na prevezivanje jednog pokazivača
 - objekti stanja mogu biti deljeni
 - ako objekti stanja nemaju attribute, konteksti mogu da dele objekat stanja
 - stanje koje objekti stanja reprezentuju je kodirano tipom objekta
 - tada su objekti stanja “muve” bez unutrašnjeg stanja, imaju samo ponašanje

Stanje (7)

- UML notacija:



- Povezani uzorci:
 - *Muva* – objašnjava kada se objekti stanja mogu deliti
 - Objekti stanja se često realizuju kao *Unikati*