POGLAVLJE 2

UML – objedinjeni jezik za modelovanje

Pregled

U ovom poglavlju

U ovom poglavlju ukratko objašnjavamo objedinjeni jezik za modelovanje (engl. *Unified Modeling Language, UML*), koji se koristi za modelovanje objektno orijentisanih sistema. Ako niste upoznati sa UML-om, ovo poglavlje će vam pružiti minimum potreban za razumevanje dijagrama u knjizi.

U ovom poglavlju:

- opisujem šta je UML i zašto se koristi;
- razmatram UML dijagrame koji su neophodni za ovu knjigu:
  - dijagram klasa
  - dijagram interakcija

Šta je UML?

UML nudi razne dijagrame za modelovanje

UML je vizuelni jezik (koristi crteže kao semantičke oznake) koji opisuje modele programa. Pod modelima programa, podrazumevam predstavljanje programa pomoću dijagrama u kojima se vide veze između objekata u kodu.

UML nudi nekoliko vrsta dijagrama – neke za analizu, druge za projektovanje i ostale za realizovanje (tačnije, za raspoređivanje, distribuciju koda). (U tabeli 2-1 ukratko sam predstavio najvažnije dijagrame). Svaki dijagram prikazuje veze između različitih skupova entiteta, zavisno od namene dijagrama.

Tabela 2-1 UML dijagrami i njihove namene

Kada ste...	Koristite...
U fazi analize	• **Dijagrame slučajeva korišćenja (engl. *use case diagrams*),** koji sadrži entitete u interakciji sa sistemom (tj. korisnicima i drugim sistemima) i prikaz funkcija koje treba da realizujete. • **Dijagrame aktivnosti (engl. *activity diagrams*),** koji se usredsređuju na tok u aktivnosti domenu problema (prostor u kome ljudi i ostali činioci funkcionišu, oblast subjekata u programu) više nego na logički tok programa. *Napomena*: pošto je ova knjiga usredsređena pre svega na projektovanje, neću objašnjavati ni dijagrame slučajeva korišćenja, ni dijagrame aktivnosti.
Pri ispitivanju *interakcija* između objekata	• **Dijagrame interakcija (engl. *interaction diagrams*)**, koji pokazuju međusbne interakcije objekata. Od koristi su i pri proveri zahteva i pri proveri projekata, s obzirom na to da su usmereni na posebne slučajeve, a ne na opštu situaciju. Najpopularnija vrsta ovakvih dijagrama je **dijagram sekvence (engl. *sequence diagrams*)**.
U fazi projektovanja	• **Dijagrame klasa (engl. *class diagrams*)**, koji podrobno opisuju veze između klasa.
Pri ispitivanju *ponašanja* objekta koje je uslovljeno njegovim stanjem	• **Dijagrame stanja (engl. *state diagrams*)**, koji podrobno opisuju stanja objekta, kao i prelaze između tih stanja.
U fazi raspoređivanja	• **Dijagrame raspoređivanja (engl. *deployment diagrams*)**, koji pokazuju raspored modula. U ovoj knjizi ne opisujem te dijagrame.

Zašto se koristi UML?

Prvenstveno za komunikaciju

UML se koristi prvenstveno za komunikaciju – sa samim sobom, članovima tima i klijentima. Jedan od stalnih problema u razvoju softvera jesu neprecizni zahtevi (bilo nekompletni, bilo netačni). UML nudi alatke za bolje prikupljanje zahteva.

Za jasnoću

UML omogućava da odredim da li se moje shvatanje sistema podudara sa shvatanjima drugih ljudi. Pošto su sistemi složeni i prenose različite vrste informacija, UML omogućava da se različiti dijagrami koriste s različitim vrstama informacija.

Za preciznost

Da biste uvideli vrednost UML-a, prisetite se poslednjih pregleda projekata. Ako ste ikada prisustvovali pregledu na kome neko opisuje kôd ne upotrebljavajući jezik za modelovanje, kao što je UML, njegovo

razmatranje je verovatno bilo i zbunjujuće i predugačko. UML nije samo bolji način opisivanja objektno orijentisanih projekata već i primorava programera da preispita svoj pristup (pošto mora da zapiše podatke).

Dijagram klasa

Osnovni dijagrami modelovanja

Najosnovniji UML dijagram je dijagram klasa. On opisuje klase i prikazuje veze između njih. Tipovi mogućih veza su:

- Kada je jedna klasa „vrsta“ druge klase: veza „*jeste*“
- Kada postoje veze između dve klase
  - Jedna klasa „sadrži“ drugu klasu: veza „*ima*“
  - Jedna klasa „koristi“ drugu klasu

Postoje varijacije navedenih pojmova. Recimo, pojam sadržavanja može da označava:

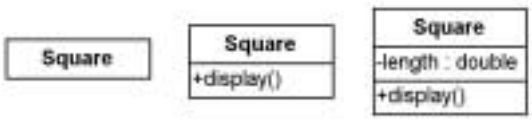
- Sadržani elemenat je deo spoljnog elementa (kao motor u kolima).
- Postoji skup samostalnih stvari (kao avioni na aerodromu).

Prvi primer se naziva *kompozicija* (engl. *composition*), a drugi *agregacija* (engl. *aggregation*).[1]

Različiti načini prikazivanja informacija u klasi

Slika 2-1 prikazuje nekoliko bitnih stvari. Prvo, svaki pravougaonik predstavlja klasu. U UML-u, u klasi predstavljam:

- Ime klase
- Podatke članove klase
- Metode (funkcije) klase



Slika 2-1 Dijagram klasa – tri varijante

1. Gamma, Helm, Johnson i Vlissides („Velika četvorka“) nazvali su prvo „agregacijom“, a drugo „kompozicijom“ – obrnuto od UML-a. Međutim, knjiga „Velike četvorke“ napisana je pre nego što je dovršen UML. Sadašnja definicija je u skladu sa UML-ovom. To objašnjava i neke od razloga za nastanak UML-a; pre nego što je nastao UML, postojalo je nekoliko različitih jezika za modelovanje, svaki od njih sa sopstvenim oznakama i pojmovima.

Postoje tri načina prikazivanja klasa:

- *Levi pravougaonik* prikazuje samo ime klase. Taj tip predstavljanja klasa koristim kada detaljnije informacije nisu potrebne.
- *Srednji pravougaonik* prikazuje i ime i metode klase. U ovom slučaju, klasa `Square`[2] sadrži metodu *display*. Znak plus (+) ispred *display* (naziv metode) označava da je metoda javna – objekti druge klase mogu je pozivati.
- *Desni pravougaonik* prikazuje isto što i prethodna dva (naziv i metode klase) uz podatake članove klase. U ovom slučaju, znak minus (-) pre atributa *length* (tipa double) pokazuje da je vrednost atributa privatna, to jest, dostupna samo objektu kome pripada. [3]

Oznake za dostupnost

Dostupnost atributa i metoda klasa može se zadati. Pomoću UML-a možete označiti dostupnost svakog člana. Većina objektno orijentisanih jezika podržava tri tipa dostupnosti:

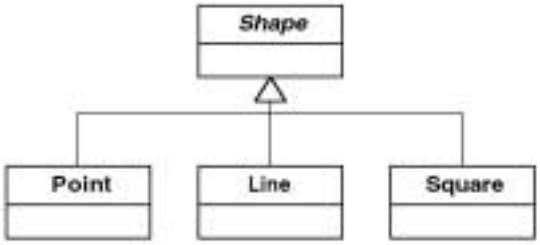
- **Javno**: označeno znakom plus (+).
  Svi objekti mogu pristupiti podatku ili metodi.
- **Zaštićeno**: označeno znakom tarabe (#).
  Samo objekti te klase i svih njenih izvedenih klasa (uključujući i klase izvedene iz tih klasa) mogu pristupiti tom podatku ili metodi.
- **Privatno**: označeno znakom minus (-).
  Samo metode te klase mogu pristupiti tom podatku ili metodi. (Napomena: neki jezici dopuštaju pristup samo određenom objektu)

Dijagram klasa prikazuje i veze
Prikazuje vezu „jeste“

Dijagrami klasa prikazuju i veze između različitih klasa. Slika 2-2 prikazuje vezu između klase **`Shape`** i nekoliko klasa izvedenih iz nje.

Slika 2-2 prikazuje nekoliko stvari. Prvo, vrh strelice ispod klase **`Shape`** označava da se klase koje pokazuju na **`Shape`** izvode iz klase **`Shape`**. Dalje, *iskošeno* ime klase **`Shape`** označava da je to apstraktna klasa. Apstraktna klasa služi da definiše interfejs za klase izvedene iz nje.

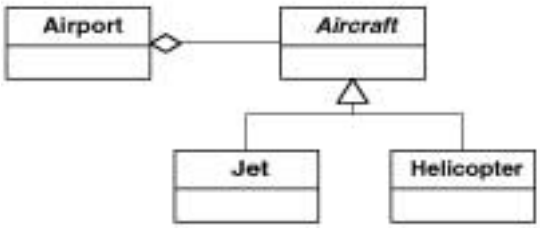
2. Ime klase uvek prikazujemo podebljano, kao u ovom slučaju.
3. U nekim jezicima, objekti mogu da pristupaju privatnim podacima drugih objekata iste klase.



Slika 2-2 Dijagram klasa prikazuje veze „jeste“.

Prikazuje vezu „ima“

Postoje dve vrste veza „*ima*“. Objekat može posedovati drugi objekat, a da sadržani objekat jeste ili nije deo tog objekta. Na slici 2-3 pokazujem da **`Airport (aerodrom)`** „ima“ klasu **`Aircraft (letelica)`**. **`Aircraft`** nije deo klase **`Airport`**, ali i dalje mogu reći da je **`Airport`** ima. Ovaj tip veze se naziva *agregacija.*



Slika 2-3 Dijagram klasa prikazuje vezu „ima“.

Agregacija

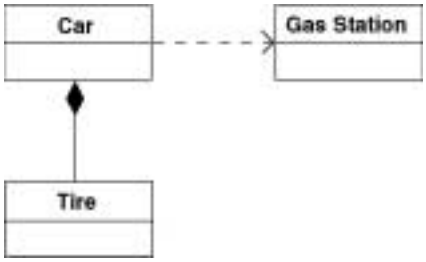
U ovom dijagramu pokazujem da je **`Aircraft (letelica)`** ili **`Jet (mlaznjak)`** ili **`Helicopter (helikopter)`**. Vidi se da je **`Aircraft`** apstraktna klasa pošto je njen naziv prikazan iskošeno. To znači da će **`Airport`** imati ili **`Jet`** ili **`Helicopter`**, ali će ih isto koristiti (kao objekat tipa **`Aircraft`**). Prazni (neobojeni) romb s desne strane klase **`Airport`** ukazuje na agregaciju.

Kompozicija

Drugi tip veze „*ima*“ ostvaruje se kada je jedan objekat deo drugog. Taj tip veze naziva se *kompozicija.*

Kompozicija i veza „koristi“

Slika 2-4 pokazuje da **`Car (automobil)`** ima deo klase **`Tire (guma)`**. **`Car`** je sačinjen od objekata klase **`Tire`** i ostalih delova. Ovaj tip veze „*ima*“, kompozicija, prikazan je punim rombom. Dijagram takođe pokazuje da **`Car`** koristi klasu **`GasStation (benzinska pumpa)`**. Veza „*koristi*“ je označena isprekidanom linijom sa strelicom. Ona se često naziva i veza zavisnosti (engl. *dependency relationship*).



Slika 2-4 Dijagram klasa prikazuje kompoziciju i vezu „koristi".

Kompozicija nasuprot agregacije

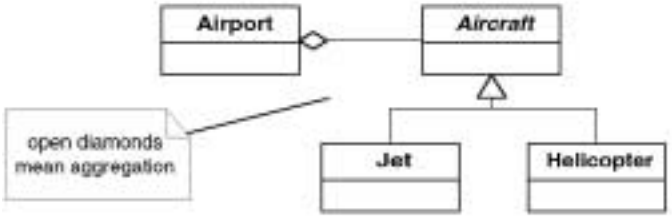
I kompozicija i agregacija podrazumevaju da objekat sadrži jedan ili više objekata. Ipak, kompozicija označava da je jedan objekat deo drugog objekta, dok agregacija označava da su sadržani objekti više skup nezavisnih stvari. Kompoziciju tumačimo kao nedeljivu asocijaciju, pri čemu životni vek unutrašnjeg objekta zavisi od spoljašnjeg objekta. Primena konstruktora i destruktora olakšava stvaranje i uništavanje objekta.

Napomene u UML-u

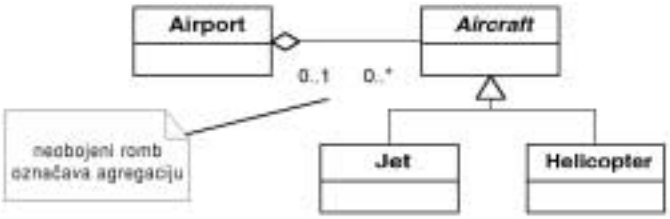
Na slici 2-5, prikazan je nov simbol: napomena. Okvir koji sadrži poruku „neobojeni rombovi označavaju agregaciju" predstavlja napomenu. Napomene izgledaju kao papir sa savijenim gornjim desnim uglom. Često ih linija povezuje sa određenom klasom, što znači da se napomene odnose baš na tu klasu.

Pokazuje broj stvari koje ima drugi objekat.

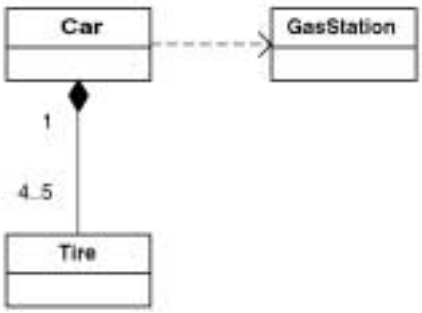
Dijagrami klasa prikazuju veze između klasa. Međutim, kompozicija i agregacija više se odnose na objekte nego na cele klase. Na primer, istina je da klasa `Airport` ima klasu `Aircraft`, ali je preciznije reći da određeni aerodromi imaju određene letelice. Može se postaviti pitanje: „Koliko letelica ima aerodrom?". To se naziva *kardinalnost* (engl. *cardinality*) veze. Prikazana je na slikama 2-6 i 2-7.



Slika 2-5 Dijagram klasa s napomenom.



Slika 2-6 Kardinalnost veze `Airport-Aircraft`.



Slika 2-7 Kardinalnost veze `Car-Tire`.

Kardinalnost

Slika 2-6 govori da jedan objekat tipa **`Airport`** ima od 0 do bilo kog broja (ovde prikazano zvezdicom, ali ponekad se prikazuje slovom „n“) objekata tipa **`Aircraft`**. Kardinalnost „0..1“ na strani objekta tipa **`Airport`** označava da svaki objekat tipa **`Aircraft`** može da bude u 0 ili 1 objekata tipa **`Airport`** (letelica može biti u vazduhu).

Nastavak o kardinalnosti

Slika 2-7 govori da objekat tipa **`Car`** ima 4 ili 5 objekata tipa **`Tire`** (može, ali ne mora imati rezervnu gumu). Gume su na tačno jednim kolima. Ako se izostavi specifikacija kardinalnosti, neki ljudi pretpostavljaju da postoji jedan objekat. To nije tačno. Ako kardinalnost nije zadata, ne treba da pretpostavljate koliko objekata postoji.

Isprekidane linije prikazuju zavisnost

Kao i ranije, isprekidana linija između klase **`Car`** i klase **`GasStation`** na slici 2-7 pokazuje da postoji zavisnost između njih. UML koristi isprekidanu strelicu da bi prikazao semantičku vezu (značenja) između dva elementa.

Dijagrami interakcija

Dijagram sekvence

Dijagrami klasa prikazuju statičke veze između klasa. Drugim rečima, oni ne prikazuju aktivnosti. Iako su vrlo korisni, ponekad je neophodno prikazati kako objekti klasa sarađuju.

UML dijagrami koji pokazuju kako objekti međusobno sarađuju nazivaju se *dijagrami interakcija*. Uobičajen tip dijagrama interakcija je dijagram sekvence, koji je prikazan na slici 2-8.

Kako čitati dijagram sekvence

Dijagrami sekvence se čitaju od vrha ka dnu.

- Svaki pravougaonik na vrhu predstavlja određeni objekat. Iako većina pravougaonika sadrži ime klase, zapazite da se ispred njega nalazi znak dve tačke. Neki pravougaonici sadrže dva imena, na primer, **`shape1 : Square`**
- Okviri na vrhu sadrže ime klase (desno od dve tačke) i opciono, ime objekta (ispred dve tačke).
- Uspravne linije predstavljaju životni vek objekata. Nažalost, većina programa za crtanje UML dijagrama iscrtava linije od vrha do dna, tako da je nejasno kada objekat zapravo nastaje.
- Vodoravnim linijama pokazujem kako objekti međusobno komuniciraju.
- Rezultati funkcija (vrednosti i/ili objekti) ponekad su eksplicitno prikazani, a nekad su rezultati sami po sebi razumljivi.

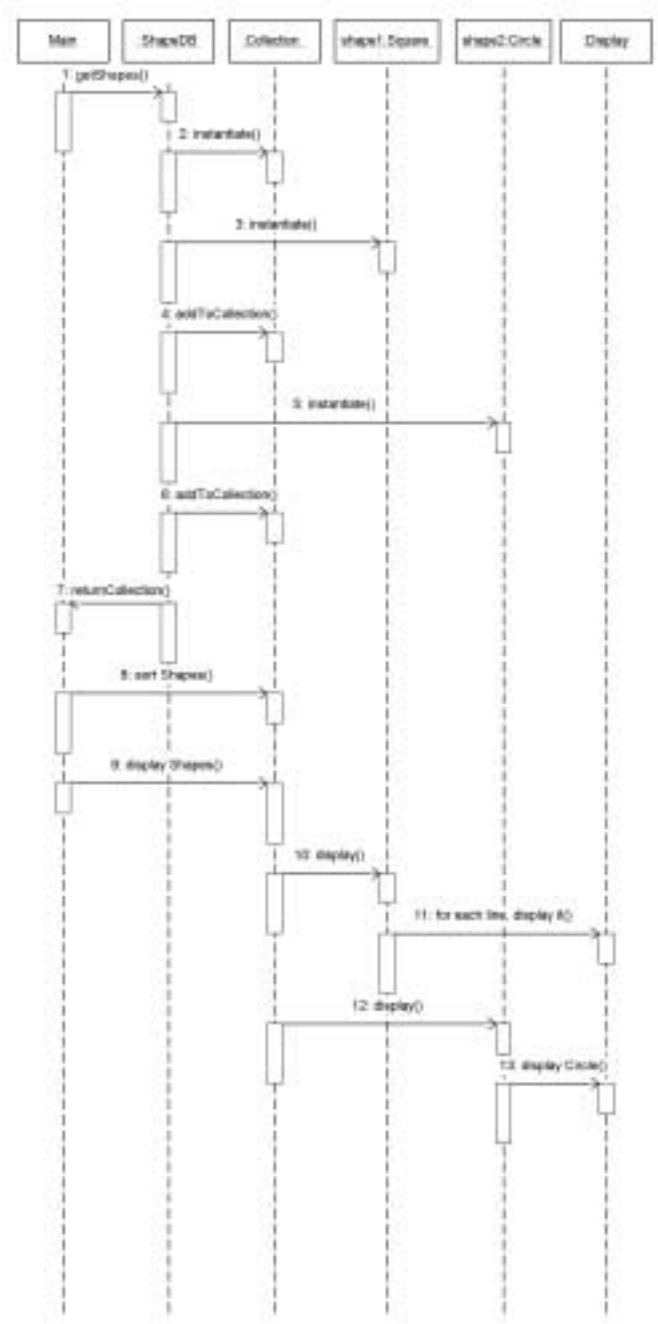
Na primer, na slici 2-8:

- Na vrhu, vidim da klasa **`Main`** šalje poruku **`getShapes`** („čitaj oblike“) objektu **`ShapeDB`** (koji nije imenovan). Kada primi poruku „čitaj oblike“, objekat tipa **`ShapeDB`**:
  - Instancira skup
  - Instancira kvadrat
  - Dodaje kvadrat skupu
  - Instancira krug
  - Dodaje krug skupu
  - Vraća skup pozivajućoj rutini (iz klase Main)

I ostatak dijagrama čitam od vrha ka dnu da bih video ostale akcije. Taj dijagram se naziva dijagram sekvence zato što opisuje sekvence operacija.

Zapis `objekat: klasa`

Na nekim UML dijagramima treba precizno označiti specifičan objekat neke klase. To se postiže povezivanjem imena objekta i imena klase znakom dve tačke. Na slici 2-8, oznaka **`shape1: Square`** odnosi se na objekat **`shape1`** klase **`Square`**.



Slika 2-8 Dijagram sekvence za oblike.

Sažetak

U ovom poglavlju

Svrha UML-a je i da se opišu projekti i da se objasne drugima. Ne brinite previše o izradi dijagrama na „pravi" način. Uvek razmišljajte o najboljem načinu da saradnicima i/ili klijentima predstavite koncept svog projekta. Drugim rečima:

- Ukoliko smatrate da nešto treba reći, koristite napomene.
- Ako niste sigurni šta označava određena slika ili simbol, pa značenje morate da tražite u udžbenicima, napišite napomenu jer će taj simbol i drugima biti nejasan.
- Trudite se da dijagram bude što jasniji.

Naravno, to znači da ne bi trebalo da koristite UML na nestandardan način jer niko drugi neće razumeti dijagrame. Dok crtate dijagrame mislite o tome šta pokušavate da prenesete čitaocima.