MRS

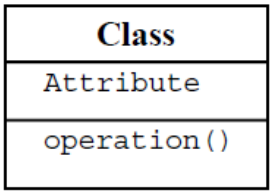
LAB 06

MRSLab06 – Metodologija Razvoja Softvera – Vežba 06

Klasni dijagram

1. Dijagrami klasa

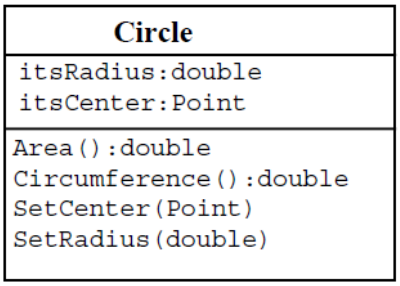
Dijagram klasa je UML dijagram koji sadrži klase, interfejse, pakete i njihove međusobne odnose koji daju logički pogled na ceo softverski sistem, ili na neku njegovu funkcionalnost. Svrha dijagrama klasa je da na pojednostavljen način prikaže klase koje se nalaze u jednoj aplikaciji. Dijagrami klasa prikazuju statičku strukturu sistemu u obliku klasa i relacija između njih. Klasa opisuje skup objekata a relacija opisuje skup veza. U objektno orjentisanim aplikacijama, klase imaju atribute (pripadajuće varijable), metode (pripadajuće funkcije) i relacije sa drugim klasama. UML klasni dijagram može relativno lako prikazati sve ove činioce. Osnovni element klasnog dijagrama je ikonica koja predstavlja klasu. Ova ikonica je predstavljena na sledećoj slici:



Slika broj 1. Predstavljanje klase

Ikona za klasu je jednostavno pravougaonik koje je podeljen u tri dela. Skroz gornji deo sadrži ime klase. Srednji deo sadrži listu atributa (pripadajuće varijable), a donji deo sadrži listu metoda (pripadajuće funkcije). U mnogim dijagramima se donje dve oblasti izostavljaju. Čak i kada su prisutne, one obično ne prikazuju svaki atribut ili metodu. Cilj je da se prikažu samo oni atributi i metode koje su korisne za dati dijagram.

Mogućnost da se skrati ikona je jedno od obeležja UML-a. Svaki dijagram ima svoju svrhu. Ta svrha može biti da se predstavi određeni deo sistema, ili da se uopšteno prikaže ceo sistem. Ikone za klase se u datim dijagramima skraćuju po potrebi. Uglavnom ne postoji potreba da se prikaže svaki atribut i metoda klase u bilo kom dijagramu. Sledeća slika prikazuje tipičan UML opis klase koja predstavlja krug:

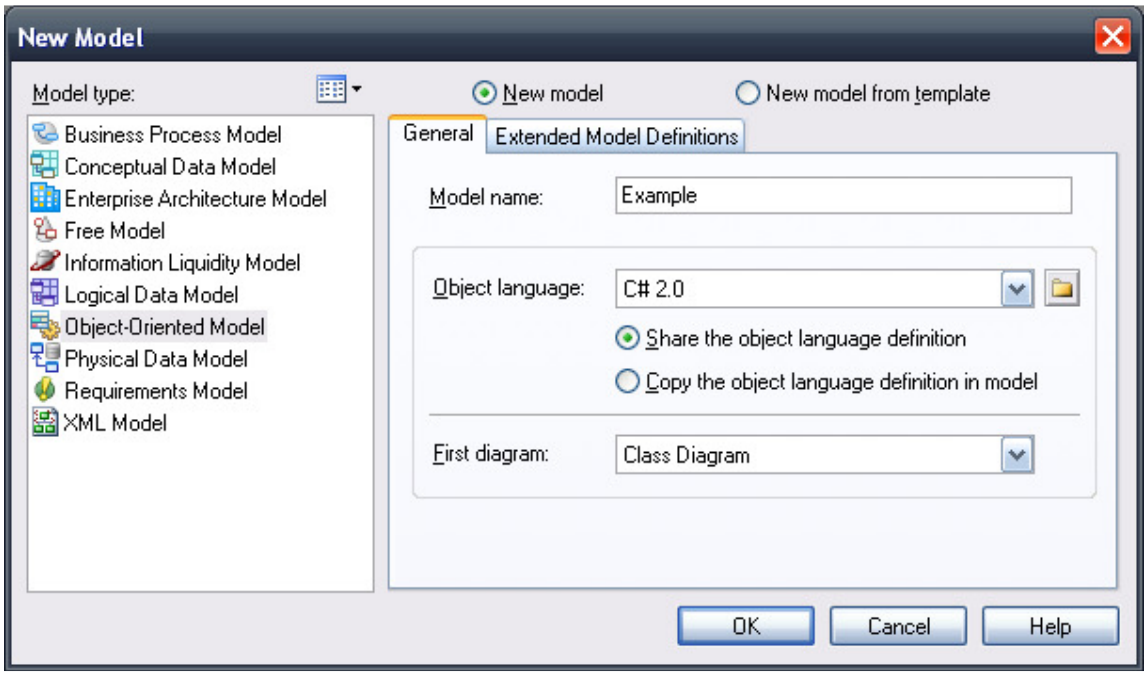


Slika broj 2. Tipičan UML opis klase koja predstavlja krug

Primećuje se da svaka varijabla ima iza sebe tip varijable. Ukoliko je tip suvišan ili nepotreban, može se izostaviti. Takođe su metode predstavljene sa tipom povratne vrednosti koji isto može biti izostavljen.

1.1. Kreiranje dijagrama klasa

Da bi kreirali novi dijagram klasa iz menija u PowerDesigner-u biramo File → New Model da bi se prikazao prozor New Model kao na sledećoj slici.



Slika broj 3. Kreiranje klasnog dijagrama

U listi tipova modela na levoj strani prozora biramo "Object-Oriented Model".

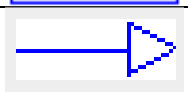
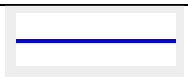
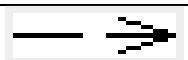
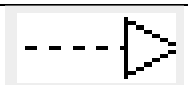
U polju "Model name:" upisujemo ime novog modela koji kreiramo. Kod modela koji može biti korišćen za generisanje koda, je izveden iz ovog imena u skladu sa konvencijama imenovanja modela.

Sledeći korak je odabir objektnog jezika iz liste i odabir jedne od dve ponuđene opcije za vidljivost promena u objektnom jeziku. Prva opcija podrazumeva da se promene u jeziku vide u svim povezanim objektno orjentisanim modelima, dok druga opcija govori da su promene vidljive samo u okviru datog modela.

Nakon ovoga se bira kog tipa će biti prvi dijagram u modelu. Za kreiranje dijagrama klasa biramo "Class Diagram". Potvrđujemo podešavanje klikom na taster "OK".

2. Objekti u dijagramu klasa

U dijagramu klasa se najčešće kreiraju sledeći objekti:

Objekat	Alat	Simbol	Opis
Klasa			Skup objekata koji dele iste atribute, operacije, metode i relacije
Interfejs			Opisuje spolja vidljive operacije klase, objekta ili drugog entiteta bez specifikacije unutrašnje strukture
Generalizacija			Veza između klasa koja pokazuje da klasa potomak deli strukturu i ponašanje definisano u jednoj ili više klasa roditelja
Asocijacija			Strukturalna veza između objekata koji pripadaju različitim klasama
Agregacija			Forma asocijacije koja specificira celina / deo relaciju između klase i agregisane klase (npr. automobil ima motor i točkove)
Kompozicija			Forma agregacije ali sa jakim posedovanjem i istim životnim vekom delova i celine (delovi žive i umiru sa celinom - npr. dostava i putanja dostave)
Zavisnost			Relacija između dva elementa u modelu pri čemu promena jednog elementa ima uticaj na drugi element
Realizacija			Klasa implementira metode navedene u interfejsu

Slika broj 4. Objekti klasnog dijagrama

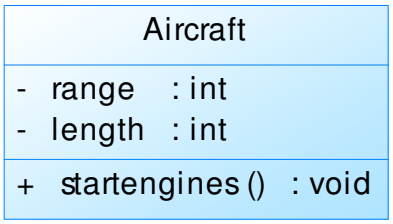
2.1. Klasa

Klasa predstavlja opis skupa objekata koji imaju sličnu strukturu i ponašanje, i dele iste atribute, operacije, relacije i semantiku. Struktura klase je opisana njenim atributima i asocijacijama, a njeno ponašanje je opisano njenim operacijama.

Klase i relacije koje se kreiraju među njima, formiraju osnovnu strukturu objektno orjentisanog modela. Klasa definiše koncept unutar aplikacije koji se modeluje, kao što su:

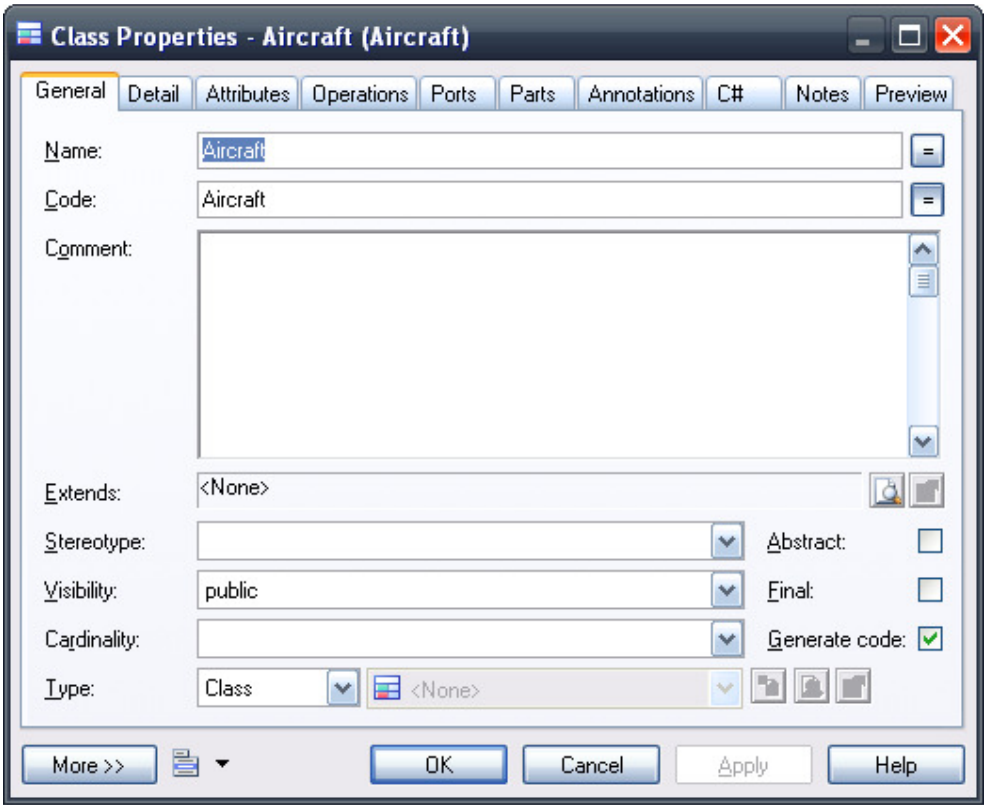
- fizički pojam (npr. automobil)
- poslovni pojam (npr. narudžbina)
- logički pojam (npr. raspored emitovanja)
- ponašanje (npr. zadatak)

Sledeći primer predstavlja klasu *Aircraft* sa atributima *range* i *length* i operacijom *startengines*.



Slika broj 5. Primer klase

Osobine klase se mogu menjati iz njene stranice sa osobinama. Da bi otvorili ovu stranicu dvokliknemo na simbol koji predstavlja klasu na radnoj površini. Nakon ovoga se otvara prozor kao na sledećoj slici:



Slika broj 6. Properties dijalog klase Aircraft

Prozor je izdeljen na tabove pri čemu se najčešće koriste sledeće osobine:

- General
  - Name - određuje ime stavke koje treba da bude jasno i precizno
  - Code - određuje tehničko ime objekta, koje će se koristiti kod generisanja koda i skripti
  - Extends - određuje klasu roditelja (sa kojom je data klasa povezana putem generalizacije)

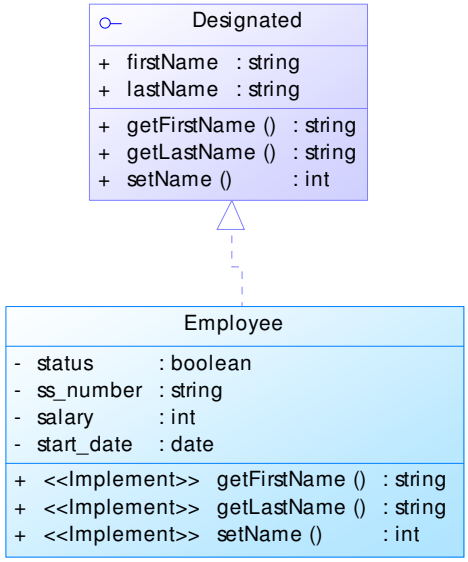
- Visibility - određuje vidljivost objekta. Ponuđene opcije su:
  - Private - vidljiv samo sebi
  - Protected - vidljiv sebi i naslеđenim objektima
  - Package - vidljiv svim objektima u istom paketu
  - Public - vidljiv svim objektima
- Cardinality - određuje broj instanci koje klasa može imati. Može se birati između:
  - 0..1 - nijedna ili jedna
  - 0..* - nijedna ili neograničen broj
  - 1..1 - jedna
  - 1..* - jedna do neograničenog broja
  - * - neograničen broj

- Attributes - služi za dodavanje novih atributa koji pripadaju klasi
- Operations - služi za dodavanje novih metoda koje pripadaju klasi

Atribut je imenovana osobina klase (ili interfejsa) koja opisuje njene karakteristike. Klase (ili interfejsi) mogu imati nula ili više atributa. Svaki objekat u klasi ima iste atribute, ali vrednosti atributa mogu biti različite. Ime atributa unutar klase mora biti jedinstveno. Razlika atributa kod klasa i interfejsa je u tome što interfejsi mogu imati samo konstantne atribute.

2.2. Interfejs

Interfejs je sličan klasi ali se koristi da definiše specifikaciju ponašanja. On ne sadrži implementaciju. Interfejs sadrži nazive metoda. Klasa može implementirati jedan ili više interfejsa. Da bi klasa implementirala jedan interfejs ona mora implementirati sve metode koje su navedene u interfejsu.



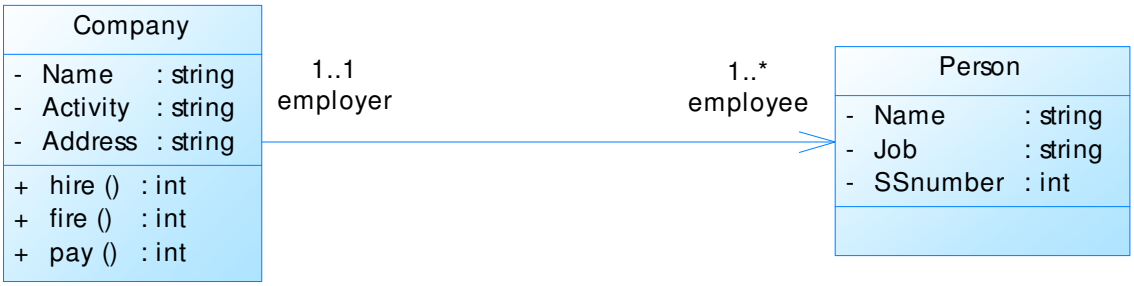
Slika broj 7. Primer klase koja implementira interfejs

Osobine interfejsa se mogu menjati iz njegove stranice sa osobinama. Da bi otvorili ovu stranicu dvokliknemo na simbol koji predstavlja klasu na radnoj površini.

Prozor je izdeljen na tabove pri čemu su najčešće korišćene osobine slične kao i kod klasa.

2.3. Asocijacija

Asocijacija predstavlja relaciju između klasa ili između klasa i interfejsa. Ona se prikazuje kao puna linija između dva objekta. Asocijacija se može imenovati tako što se svakom kraju dodeli ime koje opisuje funkciju klase kako je vidi druga klasa. Npr. osoba posmatra kompaniju u kojoj radi kao poslodavca, dok ta kompanija posmatra osobu kao zaposlenog.



Slika broj 8. Primer asocijacije

Navigacija između instanci u gornjem primeru je moguća u oba smera. Ona omogućava kompaniji da dobije listu svojih radnika, kao i svakom radniku da vidi u kojoj kompaniji radi. Kod asocijacije je suština da jedna klasa ima kao atribut instancu druge klase. Kada je veza između klasa višestruka, onda je tip instance druge klase obično niz instanci klase.

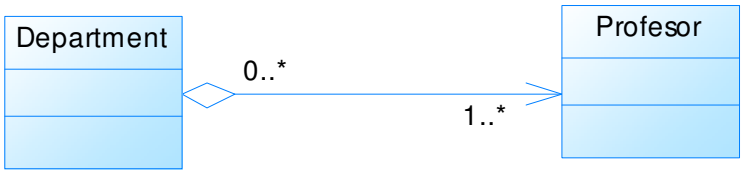
```
public class Company
{
        public String Name;
        public String Activity;
        public String Address;
        public Person[] employee;
}
```

Primer programske realizacije asocijacije

2.4. Agregacija

Ova relacija prikazuje da je agregisana klasa (klasa koju dodiruje beli romb) u nekom smislu "cela", a da je druga klasa u relaciji u nekom smislu "deo" cele klase. Agregacija je vrsta relacija asocijacije koja je više određena u odnosu na asocijaciju. To je asocijacija koja predstavlja celina / deo relaciju. Agregacija ne može uključiti više od dve klase.

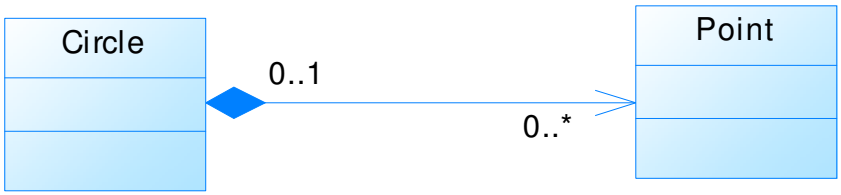
Agregacija se može pojaviti kada je klasa kolekcija ili kontejner druge klase, ali gde sadržana klasa nema jaku zavisnost trajanja životnog ciklusa sa kontejnerom. Ova znači da ako kontejner bude uništen, sadržaj ne mora biti uništen.



Slika broj 9. Primer agregacije

2.5. Kompozicija

Kompozicija je jača varijanta relacije asocijacije koja je još više određena od agregacije. Ona obično ima jaku zavisnost životnog ciklusa između instanci klase kontejnera i instanci sadržane klase. Ukoliko se uništi kontejner, uništena je i klasa koju on sadrži. Svaka instanca tipa *Circle* čini se da sadrži instancu tipa *Point*. Ova relacija je poznata kao kompozicija. On se može prikazati u UML-u pomoću odnosa među klasama. Sledeća slika prikazuje relaciju kompozicije.



Slika broj 10. Primer kompozicije

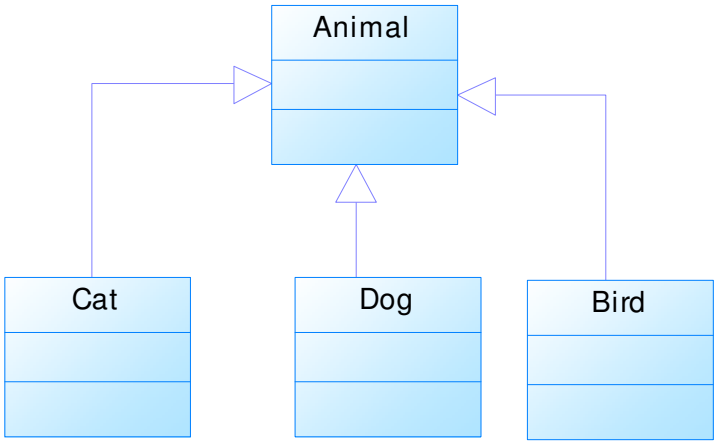
Crni romb predstavlja kompoziciju. On je postavljen na klasu *Circle* jer je klasa *Circle* kreirana od klase *Point*.

Relacije kompozicije predstavljaju jake forme relacija asocijacije. Kompozicija je relacija celina/deo. U ovom slučaju *Circle* predstavlja celinu a *Point* predstavlja deo klase *Circle*. Kompozicija ukazuje da životni vek *Point* zavisi od *Circle*. Ovo znači da ukoliko je *Circle* uništen, zajedno sa njom će biti uništen i *Point* .

2.6. Generalizacija

Generalizacija je veza između opšteg elementa (roditelja) i više određenog elementa (deteta). Relacija generalizacije se koristi kada više objekata ima slične osobine.

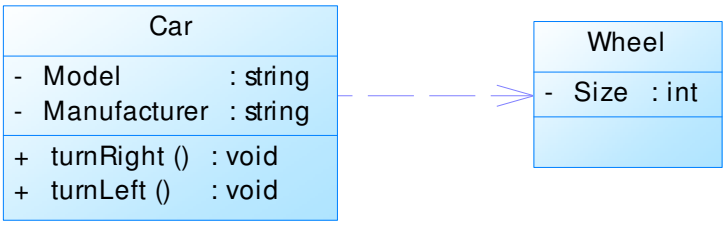
Generalizacija u UML-u je predstavljena pomoću karakteristične trougaone strelice. Ova strelica upućuje na osnovnu klasu. Jedna ili više linija koje kreću od osnovne klase povezuju ovu klasu sa klasama koje je nasleđuju. Generalizacija se može kreirati između dve klase ili između dva interfejsa.



Slika broj 11. Primer generalizacije

2.7. Zavisnost

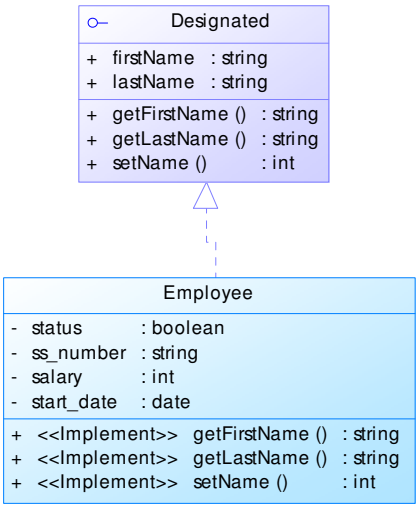
Zavisnost je semantička zavisnost između dva objekta u kojoj izmena u jednom objektu može uticati na značenje drugog objekta. U dijagramima klasa, zavisnost se može kreirati između klase i interfejsa, dve klase ili dva interfejsa. To je slabija forma relacije koja ukazuje da jedna klasa zavisi od druge jer je koristi u određenom vremenskom trenutku. Zavisnost postoji ako je klasa parametar u pozivu metode, ili varijabla u okviru metode druge klase.



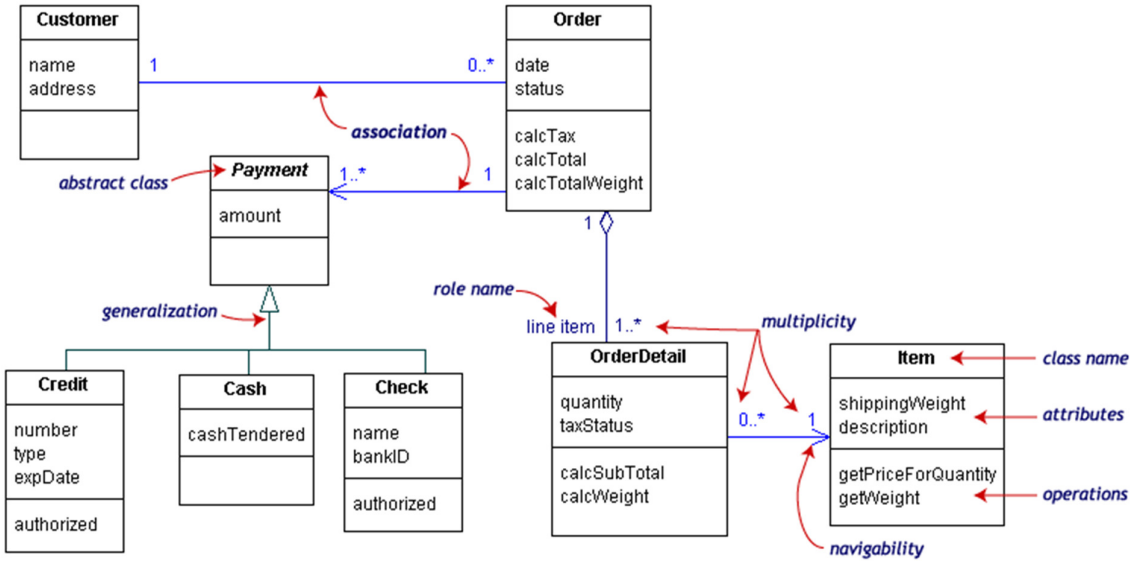
Slika broj 12. Primer zavisnosti

2.8. Realizacija

Realizacija je relacija između klase i interfejsa. U realizaciji klasa implementira metode navedene u interfejsu. Interfejs se naziva element specifikacije a klasa element implementacije.



Slika broj 13. Primer realizacije



Slika broj 14. Primer klasnog dijagrama

Za vežbu na času / Domaći zadatak:

Kreirati dijagram klasa koji prikazuje računanje obima i površine za pravougaonik, kvadrat i krug. Ove klase su nasleđene iz klase oblik.

Pitanja za obnavljanje gradiva:

1. Čemu služe dijagrami klasa?
2. Šta je klasa?
3. Šta je interfejs?
4. Šta je asocijacija?
5. Šta je agregacija?
6. Šta je kompozicija?
7. Šta je Generalizacija?
8. Šta je Zavisnost?
9. Šta je Realizacija?