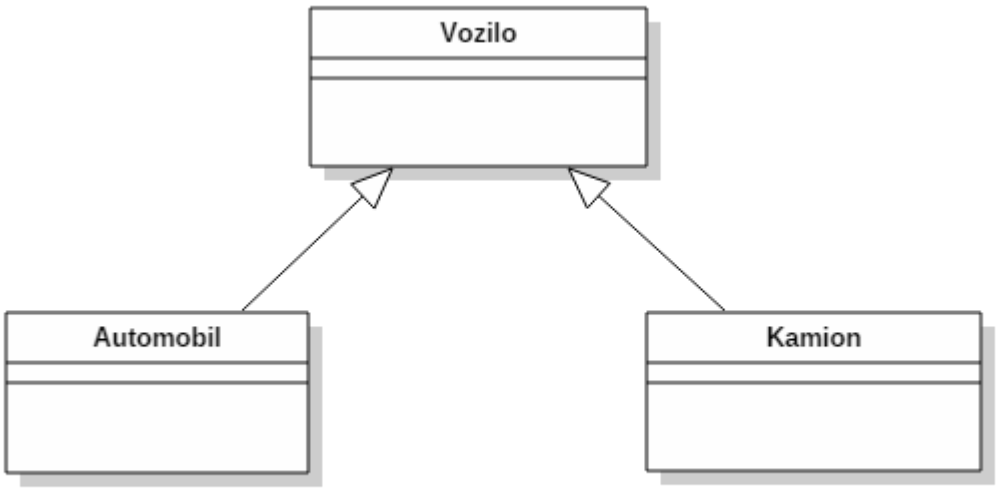
UML class dijagrami i modelovanje

U ovom dokumentu ćemo pokazati osnovne relacije između klasa, načine na koje ih možemo predstaviti na UML class dijagramu i primere koda koje ih implementiraju.

Nasleđivanje

Nasleđivanje je relacija generalizacije odnosno specijalizacije između dve klase. Kažemo da klasa naslednica proširuje funkcionalnost klase pretka, odnosno predstavlja užu kategoriju klase pretka. Na primer, imamo klase Vozilo, Automobil i Kamion. Automobil je vozilo koje dodatno opisujemo brojem putnika, brojem vrata, i slično, dok je Kamion vozilo koje dodatno opisujemo atributima kao što su nosivost. U tom slučaju, klase Kamion i Automobil nasleđuju klasu Vozilo. Na UML dijagramu se prijazuje ovako:



U programskom kodu jednostavno koristimo mehanizme nasleđivanja da implementiramo ovu relaciju:

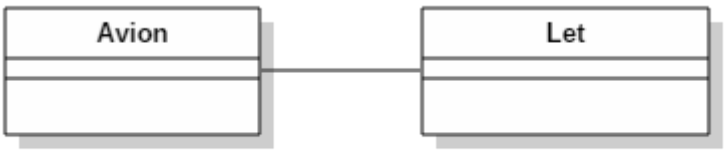
```
class Vozilo {
    ...
}

class Automobil extends Vozilo {
    ...
}

class Kamion extends Vozilo {
    ...
}
```

Asocijacija - dvosmerna

Asocijacija nam govori da su dve klase na neki način međusobno povezane. Na primer, ukoliko imamo klase Avion i Let, svaki avion zna listu letova na kojima je angažovan, a svaki let zna koji avion mu je pridružen. Obratiti pažnju da i Avion i Let su svesni druge klase – zbog toga kažemo da je asocijacija dvosmerna. Na UML dijagramu dvosmernu asocijaciju prikazujemo običnom linijom:



U programskom kodu obično obe klase imaju polja koja sadrže reference ka drugoj klasi. Bitno je napomenuti da Let nije sastavni deo aviona, te su svi letovi uglavnom čuvani na još nekom mestu u programu. Avion samo takođe ima referencu na neke od tih letova, ali ako obrišemo avion, neće nestati i njegovi letovi:

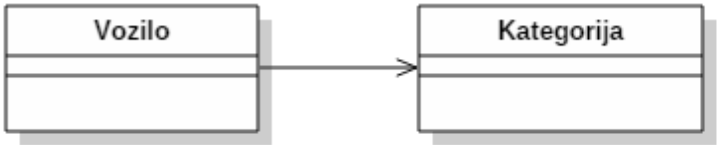
```
class Avion {
    List<Let> sviLetoviAviona;
}

class Let {
    Avion av;
}

class Program {
    List<Let> sviLetovi;
    List<Avion> sviAvioni;
    ...
}
```

Asocijacija – jednosmerna

Ukoliko u asocijaciji samo jedna klasa sadrži informacije o drugoj klasi, dok ta druga klasa ne zna ništa o prvoj, asocijacija je jednosmerna. Na primer imamo klase Vozilo i Kategorija. Svako vozilo zna kategoriju kojoj pripada i može da joj pristupi, ali Kategorija nema informacije o svim vozilima te kategorije koji postoje. Na dijagramu stavljamo strelicu koja nam govori smer asocijacije:



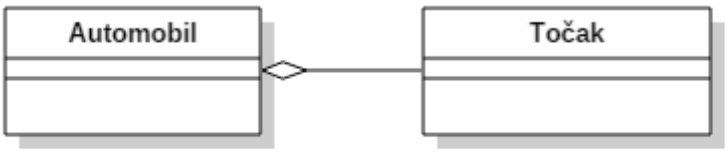
U programskom kodu jednosmernu asocijaciju implementiramo slično kao dvosmernu, ali samo jedna klasa sadrži referencu na drugu klasu:

```
class Kategorija {
    ...
}

class Vozilo {
    Kategorija kat;
    ...
}
```

Agregacija

Agregacija je specijalan slučaj asocijacije i opisuje odnos celine sa njenim delovima (**veza sadržanja - sastoji se**). Na primer, svaki automobil ima četiri točka. Međutim, delovi mogu živeti nezavisno od celine. Točkovi se mogu praviti bez automobila, pa mu se tek kasnije pridružiti. Kasnije se točkovi mogu skinuti sa automobila i razdvojiti. Na dijagramu ovu relaciju crtamo ovako (jednosmernu ili dvosmernu veza):



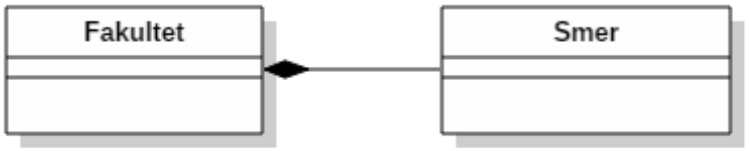
U programskom kodu agregaciju implementiramo slično kao jednosmernu ili dvosmernu asocijaciju – klasa koja predstavlja celinu sadrži reference ka svojim delovima. Međutim, uglavnom ne može da jedan objekat bude deo više celina. Na primer, više vozila je moglo da ima istu kategoriju, ali ne može da više automobila ima isti točak.

```
class Tocak {
    Automobil au;
}

class Automobil {
    List<Tocak> tockovi;
    ...
}
```

Kompozicija

Kompozicija je specijalan slučaj agregacije i opisuje relaciju u kojoj su delovi potpuno zavisni od celine i ne mogu da postoje nezavisno od nje. Na primer, imamo klase Fakultet i klasu Smer. Fakultet ima podatke o svim svojim smerovima, ali smer ne može da postoji pre nego što se fakultet napravi, i ne može da postoji dalje ukoliko se fakultet uništi. Drugim rečima, objekat celine u potpunosi upravlja životnim ciklusom objekata delova. Na dijagramu se prikazuje ovako (jednosmernu ili dvosmernu veza):



U programskom kodu se slično realizuje kao agregacija:

```
class Smer {
    Fakultet fa;
}

class Fakultet {
    List<Smer> smerovi;
    ...
}
```

Međutim, za razliku agregacije, objekti delova se uglavnom kreiraju unutar objekta celine, i uništavaju zajedno sa objektom celine. Drugim rečima, reference na objekte delove uglavnom postoje jedino u okviru objekta celine.