# Loeng 10: Koostöö- ning jadadiagrammi tehnika

## Kollaboratsioonidiagrammi tehnika

### Klasside ja eksemplaride tähistamine

UMLis on ühtne lähenemine tüüpide ja eksemplaride eristamiseks diagrammidel (vt. Larmani raamat joonis 17.5):

- Igasuguse UMLi elemendi korral (klass, actor,…) kasutab eksemplar samasugust graafilist sümbolit kui tüüp, kuid …. String joonitakse alla.

Seepärast näidatakse interaktsioonidiagrammis objekti (klassi eksemplari) tavalise klassi sümboliga (ristkülik), kus nimi on alla joonitud. Lisaks peab kollaboratsioonidiagrammis klassi nimele alati eelnema koolon ( :Sale ).

On võimalik kasutada ka eksemplari jaoks unikaalset nime ( s1: Sale )

### Linkide tähistamine

**Link** on ühendustee kahe eksemplari vahel, mis näitab navigatsiooni ja nähtavuse ( visibility ) vormi nende eksemplaride vahel.

Link on assotsiatsiooni eksemplar.
Vaadates kahte eksemplari klient-server seoses, tähendab navigatsioonitee kliendist serverini, et klient saab saata sõnumeid serverile. Näiteks on link ehk navigatsioonitee müügipunktist (POST) müügini (Sale) koos sellel võimalike sõnumitega, näiteks *addPayment*.

Link tähistatakse joonega (joonis 17.6)

### Sõnumite tähistamine

Sõnumeid objektide vahel tähistatakse märgistatud noolega ühendusjoonel. Ühe lingiga võib olla seotud palju sõnumeid. Järjenumber lisatakse, kui soovitakse näidata sõnumite järjestust juhtimisvoos. (joonis 17.7).

## Sõnumi parameetrite tähistamine

Sõnumi parameetreid võib näidata sulgudes sõnumi nime järel. On võimalik näidata ka parameetri tüüpi. (joonis 17.8).

## Tagastatava väärtuse tähistamine

Tagastatava väärtuse saab näidata sõnuminime ette kirjutatud muutujanime ning omistusoperaatori (‘:=’) kaudu. On võimalik näidata ka tagastatava väärtuse tüüpi. (joonis 17.9)

## Sünumi süntaks

UMLi standardne süntaks sõnumi jaoks on järgmine:

Return := message(parameter : parameterType) : returnType

Kuid on lubatud kasutada kollaboratsioonidiagrammis ka konkreetse programmeerimiskeele, nagu Java või Smalltalk, süntaksi.
UMLi standardse süntaksi kasutamist soovitatakse juhul, kui tahetakse hoida kollaboratsioonidiagrammi suhteliselt sõltumatuna programmeerimiskeelest.

Joonis 17.10

## Objekti sõnumid iseendale

Objekt võib saata sõnumi iseendale. Selleks on vaja joonistada link iseendaga ning sõnum sellele lingile.

## Iteratsiooni tähistamine

Iteratsiooni näidatakse sõnumi järjenumbrile järgneva tärniga (‘*’).

See näitab, et sõnumit saadetakse vastuvõtjale korduvalt tsüklis(joonis 17.12).
On võimalik sisse tuua iteratsiooniklausli, mis näitab iteratsiooni tingimust (näiteks [I := 1..10]) (joonis 17.13).

Selleks et näidata mitme sõnumi toimumist samas iteratsiooniklauslis (sõnumite hulk ühes tsüklis), tuleb korrata iteratsiooniklauslit iga sellise sõnumi jaoks (joonis 17.4).

## Eksemplaride loomise tähistamine

Keeltest sõltumatu loomise sõnum on *create* , mis saadetakse loodavale eksemplarile. (joonis 17.15).

Kuigi enamus OO keeltes tehakse eksemplaride loomine tavaliselt *new* operaatorit kasutades klassiga ja mitte elsemplariga, on selline tähistus ökonoomne, kuid mitte päris korrektne.

Uus loodav eksemplar võib sisaldada <<new>> sümbolit (stereotüüpi).

Sõnum *create* võib omada parameetreid, mis näitavad algväärtuste edastamist. See vastab konstruktori parameetritele Java’s.

## Sõnumite järjestuse tähistamine

Sõnumite järjestus esitatakse järjenumbritega (joonis 17.16):

1. Esimest sõnumit ei nummerdata (msg1() pole nummerdatud).
2. Võib kasutada hierarhilisi numbreid, kus väljuva sõnumi numbrile lisatakse ette siseneva sõnumi number.

Joonis 17.17.

## Tingimuslike sõnumite tähistamine

Tingimuslik sõnum (joonis 17.18) näidatakse järjenumbrile järgneva tingimusklausliga kandilistes sulgudes, sarnaselt iteratsiooniklauslile. Sõnum saadetakse üksnes siis, kui klausli väärtuseks on *true*.

## Vastastikku välistavate tingimuslike harude tähistamine

Näide joonisel 17.19 illustreerib järjenumbreid vastastikku välistavate tingimuslike harude korral.

Siin kasutatakse järjenumbri avaldises tingimuslikku haru tähistavat tähte (a või b või ..). Joonis näitab, et kas *1a* või *1b* võiks toimuda *msg1()* järel. Mõlemad omavad numbrit 1, kuna kumbki neist võib osutuda esimeseks sisemiseks sõnumiks.

Kummaski harus on oma hierarhia, näiteks *1b.1* on sõnumi *1.b* alamsõnumiks.

## Kollektsioonide tähistamine

Multiobjekti ehk eksemplaride hulka saab tähistada pinu ikooniga nagu joonisel 17.20.

Multiobjekt realiseeritakse tavaliselt eksemplaride grupina, mis on salvestatud konteinerisse või kollektsiooni objekti, näiteks vektor. Kuid see ei pea nii olema, multiobjekt tähistab eelkõige loogilist eksemplaride hulka.

## Sõnumid multiobjektidele

Sõnum multiobjekti ikoonile näitab, et see saadetakse kollektsiooni objektile, mitte selle elementidele.
Joonisel 17.21 saadetakse sõnum *size* vektori eksemplarile selleks, et pärida selle vektori elementide arvu.

Joonis 17.22

## Sõnumid klassiobjektile

Sõnumeid saab saata mitte üksnes eksemplaridele, vaid ka klassile endale, selleks välja kutsuda klassi skoobiga meetodeid.
Selline sõnum siseneb ristkülikusse, mille nimi ei ole alla joonitud, järelikult ta saadetakse klassile, mitte eksemplarile.(joonis 17.23).

Järelikult on tähtis alla joonida eksemplari nimed, kui peetakse silmas eksemplari, vastasel korral võidakse sõnumit interpreteerida valesti.

# Jadadiagrammi tehnika

Loe vastavat peatükki raamatust “UML Toolkit” ning vaata sealseid näiteid. Võimalused sarnanevad koostöödiagrammi omadega. Raskem on näidata valikuid ja kordusi. Kasutatakse mitte niivõrd süsteemi osade (programmide) “kogukäitumise” disainiks, vaid pigem üksikute kulgemisstsenaariumide (käitumise eksemplaride) analüüsiks. Näiteks ühe koostöödiagrammi konkreetne testimisstsenaarium. Jadadiagramm omab  eeliseid, kui tahetakse graafiliselt esitada ajaga seotud sünkroniseerimistingimusi.