Praktikum 11, OOP

Kompositsioon

Teooria

Kompositsioon iseenesest on imelihtne mõiste: ühe klassi väljal (muutujas) kasutame teisest klassist loodud objekti. Nt klassis Car on meil väli Engine. Näide:

```
public class Car {
	private Engine e;

	//... siin järgnevad meetodid
}
```

Miks kompositsioon? Kui klassi Engine realisatsioon (konkreetne programmikood) muutub, aga API (ehk kuidas meetodeid välja kutsutakse ja millised meetodid saadaval on) ei muutu, siis ei pea tegema ühtegi muudatust Car klassis.

Miks mitte pärimine? Põhimõtteliselt oleks klassis Engine pakutavaid meetodeid võimalik kasutada ka pärimise (klassi laiendamise) teel:

```
public class Car extends Engine {
```

... kuid sel juhul võiksime luua tahtmatuid sõltvusi Engine'st ning Engine klassi realisatsiooni muudatus võib (nagu loengus näite tõime add() ja addAll() meetodite põhjal) põhjustada vigu Car klassis. Pärimist kasutame siis kui tegemist on tüüp-alamtüüp suhtega (näiteks üldtüüp Car ja alamtüüp Truck või üldtüüp Student ja alamtüüp BakaStudent), muul juhul kasutame kompositsiooni (Car ei ole Engine alamtüüp ega vastupidi).

Ülesanne 1 - ettevalmistus

Looge klass Praks11, kus sees on vaid main meetod. Sealt saate ülejäänud koodi välja kutsuda. See klass on loodud sellise kasutaja perspektiivist, kes ei tea ülejäänud programmist midagi.

Ülesanne 2

Looge paketti *screens* arvutikuvarite kohta käivad klassid Display ja LedPanel. Esimene neist esindab kuvarit ja selle meetodeid, teine konkreetselt kuvari sees kasutatud LED-paneeli ning selle meetodeid. Kummagi klassi sisse ei ole vaja praegu ühtki meetodit luua, kuid realiseerige kompositsioon.

Ülesanne 3

Looge klassi Display setter-tüüpi meetod ekraani heleduse seadistamiseks. Heledus seadistatakse skaalal 1 kuni 100. Realiseerige see esialgu Display klassi meetodina.

- Kutsuge meetodid välja Praks11 main meetodist.
- Looge ka getter heleduse küsimiseks ja kutsuge samuti välja

Kapseldamine

Kapseldamine on realisatsiooni peitmine kasutaja eest. Antud näite puhul tähendab see, et klassi Display kasutaja ei pea ega tohi midagi teada sellest, kuidas see klass loodud on. See tähendab, et tal ei peaks olema ka aimu, kas me kasutame oma klassis LedPanel'i funktsionaalsust. Tema näeb ja kasutab ainult avalikke (public) meetodeid.

Kes on kasutaja? Kasutaja all mõtleme peamiselt arendajaid, kes meie klassi kasutavad, st kes loovad sellest uue objekti, nt: Display d = new Display();
Meie näites võiks klass Praks11 olla kasutaja loodud – see klass ei pea teadma midagi Display realisatsioonist.

Ülesanne 4

Heledust reguleerib kuvari sees olev LED paneel, niiet tegelikult peaks meetodid heleduse seadmise ja küsimise tarvis olema LedPanel klassis. Muutke oma programmi nii, et vastav funktsionaalsus oleks LedPanel klassis.

NB! Jälgige kapseldamist ehk realisatsiooni peitmist! Klassis Praks11 olev main meetod, kust kutsute välja heleduse seadmise ja küsimise meetodeid, ei tohi midagi teada LedPanel klassist ja ei vaja seetõttu mingisuguseid muudatusi! Praks11 klass peab ilma muutusteta tööle jääma!

Ülesanne 5

Selgub, et heleduse muutmisel tuleb meil kasutusel oleva LED ekraani korral teostada ka värvide kalibreerimine. Looge selleks meetod (meetod ise reaalselt midagi muud ei tee kui prindib konsooli „Kalibreerin”).

NB! Pidage silmas, et kompositsiooni üks eesmärk on samuti funktsionaalsuse kapseldamine. See tähendab, et antud muudatus LED paneeli töös ei tohiks kuidagi kajastuda klassi Display programmikoodis (ammugi mitte Praks11 programmikoodis). Display ja Praks11 klassid peavad peale seda täiendust ilma muudatusteta toimima.

Ülesanne 6

Looge klassi Display uus meetod, mille abil saab ekraani pimedaks muuta. Arvatavasti tuleb selleks luua ja välja kutsuda ka mingi uus meetod LedPanel klassis.

Looge meetodid ka blank-režiimist väljumiseks. Kutsuge see funktsionaalsus välja Praks11 seest.

Ülesanne 7

Monitoril on ka indikaatorlamp, mis reeglina põleb pidevalt. Kuid kui monitor on blank-režiimis, siis see lamp vilgub. Realiseerige vastav funktsionaalsus (piisab oleku muutmisel selle väljatrükkimisest). **NB!** Ilmselt on ka siin mõistlik kasutada kompositsiooni. **NB!** Klassi Praks11 (ehk kasutaja) jaoks ei tohi midagi muutuda. Tegemist on Display klassi sisese küsimusega.