## Ülesanne – Stock (maksimaalne hinne – 10 punkti)

**Põhiülesanne (maksimaalne hinne – 4 punkti).**

Luua laoarvestuse tarkvara jaoks äriloogika klassid, mis aitavad kaupade ja tellimustega läbi viia erinevaid operatsioone.

* Kaup (Item) on ostetav ja müüdav ese, mida kirjeldab nimetus ja hind. Kaupade mugavamaks loomiseks peaks looma konstruktorid, mida saab kasutada järgmiselt:

Item \*i1 = new Item("Leib", 0.95);

Item \*i2 = new Item("Piim", 0.55);

* Tellimus (Order) on kogum kaupadest ja kogustest. Igal tellimusel peaks olema unikaalne number, mille väärtustamine võiks toimuda automaatselt uue tellimuse loomisel. Realiseerida meetod add(), millega saab tellimusele tooteid lisada. Toote lisamisel tuleb määrata tellitud kogus:

Order \*receipt = new Order();

receipt->add(i1, 1);

receipt->add(i2, 5);

* Kaupasid hoitakse laos (Stock), mis talletab endas kõikide ladustatud kaupade laoseisu. Luua kaupade lattu arvele võtmiseks meetod receive(), mis võtab argumendiks etteantud tellimuse ning suurendab kaupade laoseisu vastavalt tellimusel märgitud kogustele:

Stock s = new Stock();

s.receive(receipt); // laos on nüüd 1 leib ja 5 piima

* Realiseerida meetod getAvailable(), millega on võimalik teada saada etteantud kauba laoseis praegusel ajahetkel:

int q = s.getAvailable(i2); // peaks tagastama 5

* Luua lattu meetod ka kauba väljastamiseks – dispatch() võtab argumendiks mõne tellimuse ning vähendab kauba laoseise etteantud tellimusel määratud koguste võrra:

Order \*shipment = new Order();

shipment->add(i2, 2);

s.dispatch(shipment); // lattu jäi alles 3 piima

* Jälgida, et sellisel moel moodustatud tellimusel ei oleks sama kaupa mitu korda, vaid kaubapõhised kogused oleksid summeeritud.
* Kindlasti tuleb jälgida, et meetodid töötaksid korrektselt ka sama kauba erinevate eksemplaride korral:

s.getAvailable( new Item("Piim", 0.55) ); // tagastab samuti 3

* Kirjutada testprogramm, millega on näidatud, et kõik meetodid toimivad korrektselt. Testprogrammis tuleb defineerida kaupadest väike sortiment, luua neist tellimused, mida võetakse lattu vastu ning väljastatakse. Igal sammul näidata, et laoseis on korrektselt arvutatud – seejuures võiks kasutada ka meetodit s.to\_string().

**Lisaülesanne 1 (maksimaalne hinne – 6 punkti).**

* Luua mehhanism, mis väldib laoseisu negatiivseks muutumist ning tellimustele vigaste andmete sisestamist (0 või negatiivsed kogused). Vastavad meetodid (nt. receive() jm) peaksid oma ebaõnnestumisest väljakutsujale kuidagi märku andma (nt. tagastustüübi kaudu) – tekstiväljundit (konsoolile printimine) äriloogika klassides kasutada ei tohi!

**Lisaülesanne 2 (maksimaalne hinne – 8 punkti).**

* Luua ostukorv (Cart), mida saaks kasutada veebipoes kliendi ostusoovide salvestamiseks. Realiseerida meetodid ostukorvi toodete lisamiseks add(), mõne korvi lisatud toote koguse muutmiseks change() või toote korvist eemaldamiseks remove(). Ostukorvi peaks saama ka tervikuna tühjendada meetodi clear() abil. Ostukorvi sisene andmestruktuur on vabalt valitav, kuid arvestada võimalusega, et mõni ostuhulluse käes vaevlev klient võib sama toodet mitu korda korvi lisada ja sealt eemaldada ning kogust korduvalt muuta enne lõplikku vormistamist:

Cart \*c = new Cart();

c->add(i1); // võtame siit ühe leiva...

c->add(i2, 2); // ja kaks piima

c->change(i1, 3); // tahan ikka 3 leiba

c->remove(i2); // ei, piima ma vist ei taha

c->add(i2, 1); // siiski, võtaks ühe piima

c->add(i2, 1); // võtame ühe piima veel

Pärast eelnevat peaks ostukorvis olema 3 leiba ja 2 piima. Ostukorvis ei tohiks olla 0-koguseid.

* Luua meetod getTotal(), mis arvutab ja tagastab ostukorvi summa.
* Realiseerida ostukorvis meetod checkOut(), mis vormistab ostukorvi sisu tellimuseks. Meetod peaks tagastama uue Order-objekti, millele on märgitud kõik tellitud tooted. Selliselt loodud tellimust on võimalik samuti kasutada laost kauba väljastamiseks

Order \*shipment2 = c->checkOut();

s->dispatch(shipment2);

**Lisaülesanne 3 (maksimaalne hinne – 10 punkti).**

* Ladu (Stock) peab arvet pidama kõikidest laoliikumistest, mis on aja jooksul laoseisu muutnud. Laoliikumist (Transaction) iseloomustab toode, mida liigutati, kogus ning viide tellimusele, mis liikumise põhjustas. Vastavalt tuleb realiseerida meetodid getItem(), getQuantity() ja getOrder(). Lattu saabunud kaupade korral on kogus positiivne, väljastamise korral negatiivne. Seega kõikide laoliikumiste koguste summeerimine peaks andma korrektse laoseisu. Laoliikumised salvestada laos Stock objekti sees nende toimumise järjekorras, vastav andmestruktuur on vabalt valitav.
* Luua meetod validateIntegrity(), mis iga ladustatud toote kohta, kas tehingute koondsumma annab kokku hetkel kehtiva laoseisu. Vigade korral tagastab meetod false. Laoseisu kontrollimise meetodit võiks saada välja kutsuda ka tootepõhiselt:

s.validateIntegrity(); // kõikide toodete laoseis on kooskõlaline

s.validateIntegrity(new Item("Piim", 0.55); // piima laoseis on ka Ok

Kuna korrektselt realiseeritud loogika korral kooskõla vigu ei tohiks tekkida, leidke mingi lihtne viis, kuidas tekitada vigane laoseis ning näidata, et valideerimine tõepoolest töötab ning suudab ka vigu avastada.

Tähelepanu: selles ülesandes ei ole vaja interaktiivset (sh. konsooli klaviatuurisisendi lugemine) kasutajaliidest! Testimiseks vajalikud käsud (nt. ostukorvi kaupade lisamine) kirjutatakse main()-i sisse.

**Üldised nõuded kodutöödele**.

* Programm peab töötama!
* Programm teeb täpselt seda ja ainult seda, mis on nõutud ülesandes.
* Programmi ülesehitus peab baseeruma OOP 3 vaalal: [kapseldamine (ka realisatsiooni varjamine!)](http://en.wikipedia.org/wiki/Encapsulation_%28object-oriented_programming%29), tuletamine, polümorfism.
* Programmi lähtetekst **peab** olema „[trepitud](http://foorum.hinnavaatlus.ee/viewtopic.php?t=294507)“ («[лесенкой](http://www.amse.ru/courses/algorithms6/03/Silina/03.php)») – treppimise *stiili* võite valida ise. Treppimata lähtekoodiga programm ei kuulu hindamisele! Ka klasside, nende väljade ja meetodite **nimede** juures tuleks järgida mingit ühte [stiili](https://en.wikibooks.org/wiki/C%2B%2B_Programming/Weblinks#C.2B.2B_Coding_Conventions).
* Kogu programm (eriti aga kõigi klasside kõik väljad ja meetodid) peab olema varustatud asjakohaste kommentaaridega.
* [Copy/Paste stiilis programmeerimine](https://en.wikipedia.org/wiki/Copy_and_paste_programming) on karistatav trahvipunktidega! Alternatiiviks on korduvalt kasutatava koodi paigutamine alamprogrammi või meetodisse.
* Kui töö kaitsmisel tudeng ei oska seletada oma programmi ülesehitust ja selle toimimise loogikat, siis ka funktsionaalselt õige lahenduse korral saab tudeng hindeks mitte rohkem kui 1/3 ülesande maksimumpunktidest.

TÄHTAEG: **17. november**