**Maksukalkulaator**.

Koostada maksukalkulaator, millega saab arvutada töötajate brutopalga põhjal pangaarvele üle kantavat summat banaanides ja riigile makstavat tulumaksu.Tulumaks arvutatakse järgmiste reeglite põhjal:

* Kõikidel Mugawaga banaanivabariigi residentidel (kodumaised töötajad) on maksumääraks 20%. Maksuvaba miinimumi (150 banaani) sisse jääva summa pealt tulumaksu ei arvutata.
* Võõrtöölistest oskustööjõu sissetoomise soodustamiseks on mitteresidentidel maksumääraks kehtestatud 16%, kuid nende tulumaksu arvutamisel ei arvestata maksuvaba miinimumi.
* Ülemaailmse Banaanivabariikide Liidu liikmena kehtib teistest banaanivabariikidest pärit töötajatele erikokkulepe: kui töötaja brutopalk jääb alla 500 banaani, siis tulumaksu ei arvestata. Sissetulekust alates 500 banaanist arvestatakse maksumääraks 30%.

**Hot Drinks**.

On antud kaks klassi: Kohv ja Tee. Koostada kuumade jookide (HotDrink) automaat (VendingMachine), mis oskab valmistada nii kohvi kui teed. Joogi valmistamise (make\_drink) käigus tuleb teha järgmisi tegevusi:

* Ajada vesi kuumaks.
* Lisada vette maitset andev koostisosa (sõltub joogist).
* Valmistada jook sõltuvalt valmistusviisist
* Lisada maitseained, nt. suhkur või koor

Masin peaks olema võimalikult universaalne, et tulevikus saaks oluliste ümberehitusteta lisada ka teisi pulbrist valmistatavaid jooke (nt. Cocoa) või olemasolevate eriliike (nt. Cappuccino).

Jooke tähistavate klasside (Coffee, Tea) struktuur on ette antud, kuid neid võib muuta, vajadusel lisada meetodeid või olemasolevaid ümber kohandada.

class Coffee : public HotDrink {

public:

void boilWater() {

cout << "Kuumutan vett 95C-ni" << endl;

}

void addCoffeePowder() {

cout << "Panen kohvipulbri filtrisse" << endl;

}

void prepare() {

cout << "Valan vee läbi filtri" << endl;

}

void addSugar() {

cout << "Lisan suhkru ja koore" << endl;

}

};

class Tea : public HotDrink {

public:

void boilWater() {

cout << "Keedan vett" << endl;

}

void soakTeaBag() {

cout << "Pistan teepaki kuuma vette" << endl;

}

void prepare() {

cout << "Valan teevee tassi" << endl

}

void addLemon() {

cout << "Lisan sidrunit ja mett" << endl;

}

};

Masina kasutamine peab lõpptarbijale olema võimalikult lihtne. Luua meetod orderDrink(DrinkType), millega saab ette anda, millist jooki valmistatakse. Vastavat tüüpi objekti loomine, joogi valmistamine jm. tegevused tehakse selle meetodi sees. Jookide tüübi etteandmiseks võib kasutada *enum* andmetüüpi.

enum DrinkType { TEA, COFFEE };

VendingMachine v;

HotDrink \*t = v.orderDrink(TEA);

t->drink(); // küll on hea :)

Masin peaks oskama iga joogi liigi korral arvutada joogi maksumuse. Maksumus on leitav järgmistest komponentidest:

* Vee soojendamine (0.03€)
* Pulbri maksumus (kohvipuru 0.25€, teepuru 0.05€)
* Lisandite maksumus (suhkur 0.07€, koor 0.15€, sidrun 0.10€, mesi 0.22€)

Koostada klassis HotDrink meetodid, mis arvutavad äsjavalmistatud joogi kohta:

* getCostAmount() – joogi valmistusmaksumuse
* getProfitAmount() – kaupmehe poolt võetava müügikatte e. kasumi
* getSalesAmount() - summa, palju klient peaks maksma (maksumus + kasum)

Müügikate e. marginaal on kaupmehe määratud "vaheltkasu“. Ülesandes piisab, et müügikate sõltuks protsendist, mida saab automaadis seadistada (vaikimisi 50%). Näide:

cout << t->getCostAmount() << endl; // tee sidruni ja meega: 0.03+0.05+0.10+0.22

// = 0.40€

cout << t->getSalesAmount() << endl; // klient peab maksma: 0.60€

HotDrink \*c = v.orderDrink(COFFEE);

cout << c->getCostAmount() << endl; // kohv suhkru ja koorega: 0.03+0.25+0.07+0.15

// = 0.50€

cout << c->getSalesAmount()<< endl; // klient peab maksma: 0.75€

Automaat peaks ka arvet pidama, kui palju on selle abil kaupmees kasu saanud:

float totalprofit = v.getTotalProfit();

cout << totalprofit) << endl; // kahe tellitud joogi peale kokku: 0.45€

NB! Sõltumata lahendusviisist:

* Programmis ei tohi kasutada tingimuslauseid (if, switch, vm.) jookide erisuste kirjeldamiseks.
* Polümorfismi kasutamine on kohustuslik.
* Koodis ei tohi olla copy-paste stiilis lauseid (nii vormilisi kui sisulisi). Kui vaja, looge meetodid, mille poole korduvalt pöördutakse.
* Mõelge läbi meetodite nähtavus.

**Lisaülesanne**: ideaalis ei tohiks lõppkasutajal olla võimalust luua „otse“ konkreetseid jooke, st. öelda

ilma automaadi orderDrink() meetodit kasutamata new Coffee().

**Vihje**: joogi valmistamine põhineb Šabloonmeetodil ([Template Method Pattern](http://sourcemaking.com/design_patterns/template_method/cpp/1)), tellimise juures saab kasutada Tehase meetodit ([Factory Method Pattern](http://sourcemaking.com/design_patterns/factory_method/cpp/1)).