

**Programare orientată pe obiecte – Test de laborator Seria 13/14 - 26 mai 2020**

**Enunț**

Un producător de **măști de protecție și dezinfectanți** dorește să pornească o afacere (cât mai urgent posibil în contextul actual) și în acest sens el dorește să creeze un sistem de gestionare a măștilor și dezinfectanților pe care le fabrică. Producătorul dorește să realizeze aplicația în C++ folosind concepte de OOP.

Producătorul își propune să fabrice **măști chirurgicale, dar și măști din polycarbonat**. Iar pentru secția de dezinfectanți, se dorește fabricarea de **dezinfectanți împotriva bacteriilor, împotriva fungilor dar și împotriva virusurilor**. Bineînțeles, producătorul va fabrica și dezinfectant care ucide toate aceste microorganisme.

Despre un model de mască chirurgicală, producătorul dorește să rețină următoarele informații: **tipul de protecție (ffp1, ffp2, ffp3), culoarea și numărul de pliuri**.

În plus pentru un model de mască din polycarbonat se dorește să se rețină: **tipul de protecție (care este ffp0) și tipul de prindere (elastic, scai, etc.)**.

Pentru fiecare tip de dezinfectant în parte se dorește să se rețină: **numărul de specii de organisme ce vor fi ucise, ingredientele și tipurile de suprafață** pe care poate fi aplicat (spre exemplu dezinfectantul antifungic nu se aplică pe marmură deoarece poate albi marmura).

Pentru a calcula eficiența unui dezinfectant vom recurge la următoarea formulă:

$$Eficienta_{TipOrganism} = \begin{cases} \frac{\#specii\ de\ bacterii\ ucise}{TotalBacterii}, TipOrganism = bacterie \\ \frac{\#specii\ de\ fungi\ ucisi}{TotalFungi}, TipOrganism = fung \\ \frac{\#specii\ de\ virusuri\ ucise}{TotalVirusuri}, TipOrganism = virus \end{cases}$$

Unde:

$$\begin{cases} TotalBacterii = 10^9 \\ TotalFungi = 1,5 \cdot 10^6 \\ TotalVirusuri = 10^8 \end{cases}$$

Ca orice producător care funcționează legal, acesta va dori să rețină o arhivă contabilă – toate **achizițiile** onorate. Pentru o achiziție se vor reține **data**, **numele** clientului, **un vector de pointeri către dezinfectanții achiziționați** și **un vector de pointeri către măștile achiziționate**. În plus pentru fiecare achiziție trebuie să se știe totalul comenzii. Pentru acestea, se cunosc următoarele prețuri, raportate la bucată:

{	<i>masca policarbonat</i>	<i>→ 20 lei</i>
	<i>masca textila ffp1</i>	<i>→ 5 lei</i>
	<i>masca textila ffp2</i>	<i>→ 10 lei</i>
	<i>masca textila ffp3</i>	<i>→ 15 lei</i>
	<i>dezinfectant cu eficienta &lt; 90%</i>	<i>→ 10 lei</i>
	<i>dezinfectant cu eficienta &lt; 95%</i>	<i>→ 20 lei</i>
	<i>dezinfectant cu eficienta &lt; 97.5%</i>	<i>→ 30 lei</i>
	<i>dezinfectant cu eficienta &lt; 99%</i>	<i>→ 40 lei</i>
	<i>dezinfectant cu eficienta &lt; 99.99%</i>	<i>→ 50 lei</i>

În plus, în momentul în care un client comanda o mască chirurgicală, acesta poate solicita un anumit model al măștii (de exemplu un steag sau un logo al unei echipe sportive). Pentru acest tip de preferință la prețul unei măști se adaugă 50% față de prețul normal.

### Cerințe:

I. Să se implementeze clasele necesare astfel încât aplicația să funcționeze. II. Programul este prevăzut cu un meniu interactiv în care administratorul are posibilitatea de a alege dintre următoarele opțiuni:

1. Adaugă un nou dezinfectant în stoc
2. Adaugă o nouă mască în stoc
3. Adaugă o nouă achiziție
4. Afișează dezinfectantul cel mai eficient
5. Calculează venitul dintr-o anumită lună
6. Calculează venitul obținut din măștile chirurgicale cu model.
7. Modifică rețeta unui dezinfectant, modificând unul sau mai multe ingrediente, lucru ce conduce și la o modificare a numărului de specii de organisme ce vor fi ucise și a suprafețelor pe care poate fi aplicat.
8. Afișează cel mai fidel client (clienții NU au nume unic)
9. Afișează ziua cu cele mai slabe venituri, de la deschidere până în prezent.
10. Calculează TVA-ul (19% din venituri) care trebuie returnat la ANAF pentru un anumit an.

## Demo:

```
int main() {
    MascaChirurgicala mc1, mc2("ffp2", "verde brotăcel", 55),
    mc3(m1), mc4, mc5;
    mc4 = mc2;
    std::cin >> mc5;
    std::cout << mc1 << mc2;
    MascaPolicarbonat *mp1 = new MascaPolicarbonat(), *mp2 =
new MascaPolicarbonat();
    MascaPolicarbonat *mp3 = new MascaPolicarbonat("elastic");
    std::cin >> mp1 >> mp2;
    std::cout << mp3;
    Dezinfectant *d1 = new DezinfectantBacterii(100000000,
std::vector<string>(
{"sulfati non-ionici", "sulfati cationici", "parfumuri",
"Linalool", "Metilpropanol butilpentil"}),

std::vector({"lemn, sticla, metal, ceramica, marmura"}));
    Dezinfectant *d2 = new DezinfectantVirusuri(50000000,
std::vector<string>(
{"Alkil Dimetilm Benzil Crlorura de amoniu", "parfumuri",
"Butilpentil metilpropinal"}),

std::vector({"lemn, sticla, ceramica, marmura"}));
    Dezinfectant *d3 = new DezinfectantFungi(1400000,
std::vector<string>(
{"Alkil
Etil Benzil Crlorura de amoniu", "parfumuri", "Butilpentil
metilpropinal"}),

std::vector({"sticla, plastic"}));
    std::cout << d1->eficienta() << " " << d2->eficienta() <<
" " << d3->eficienta() << "\n";
    Achizitie *a1 = new Achizitie(26, 5, 2020, "PlushBio
SRL
");
    *
    a1 += mp1; //se adauga masca de policarbonat mp1 in lista
de masti achizitionate *a1 += (&mc1); //se adauga masca
chirurgicala mc1 in lista
    *a1 += d3; // se adauga dezinfectantu de fungi d3 in lista
```

```

de dezinfectanti achizitionati
    Achizitie *a2 = new Achizitie(25, 5, 2020, "Gucci");
    *a2 += d1;
    *a2 += d2;
    *a2 += d2;
    Achizitie a3, a4(*a1);
    a3 = *a2;
    if (*a1 < *a2) {
        std::cout << a1->nume() << " are valoarea facturii mai
mica.\n";
    } else if (*a1 == *a2) {
        std::cout << a1->nume() << " si " << a2->nume() << "
au aceasi valoare a facturii.\n";
    } else {
        std::cout << a2->nume() << " are valoarea facturii
mai mica.\n";
    }

}

}

```

//Continuati voi cu restul de cerințe

### **Precizări**

1. Timpul de lucru este de 150 de minute. Orice lucrare trimisă după 11:30 nu va fi luată în considerare.
2. Pentru întrebări despre subiect, tutorii de laborator vor fi online pe Google Meets, la adresa: [meet.google.com/vbc-kqnh-ojc](https://meet.google.com/vbc-kqnh-ojc)
3. La sfârșitul timpului de lucru, studenții vor trimite pe adresa [examen.oop.fmi@gmail.com](mailto:examen.oop.fmi@gmail.com) fișierul/fișierele sursă cu extensia cpp/h/hpp (fără executabil). Acesta trebuie să conțină pe primul rând un comentariu cu numele și prenumele studentului, grupa și compilatorul folosit.
4. Puteți submida mai multe surse, dar să menționați în ultimul mail care să fie cea corectată. Alt fel, ultima sursă trimisă va fi notată.
5. Sursa predată trebuie să compileze. Sursele care au erori de compilare nu vor fi luate în considerare. Înainte de predarea surselor, studenții vor pune în comentariu eventualele părți din program care au erori de compilare sau nu funcționează corespunzător.
6. Se acceptă și soluții parțiale, care nu respectă toate cerințele din enunț, dar sunt funcționale. Acestea vor fi depunctate corespunzător.
7. În implementarea programului se vor utiliza cât mai multe dintre noțiunile de programare orientată pe obiecte, care au fost studiate pe parcursul semestrului

și care se potrivesc cerințelor din enunț.

8. Condițiile minimale de promovare a testului sunt ca codul din demo să ruleze.

9. Orice tentativă de fraudă se va pedepsi conform regulamentelor

Universității.

```
#include <iostream>

using namespace std;

// Prima data vom implementa clasa parinte (toate sunt produse)

// Produs
class Produs {
public:
    virtual double price() {

    }
};

// Masti

class Masca : public Produs {
protected:
    string tipProtectie;
};

class MascaChirurgicala : public Masca {
private:
    string culoare;
    int numarPliuri;
};

class MascaPolicarbonat : public Masca {
private:
    string tipPrindere;
};

// TODO
// Constructori
// functiile virtuale
// operator>> deoarece avem inheritance,
// vom folosi citirea cu functii virtuale
```

```

// Dezinfectantul
class Dezinfectant : public Produs {
    // TODO Ctrl+O to override the price() function
public:
    double price() override {

        // in functie de eficient decidem pretul.
    }
};

```

```

// Exista doua variante:
// Achizitie in care retinem si numele clientului
// Achizitie+Client(cu un membru id), iar in Achizitie retinem id-ul clientului

```

```

//int main() {
//  MascaChirurgicala mc1, mc2("ffp2", "verde brotăcel", 55), mc3(m1), mc4,
//  mc5;
//  mc4 = mc2;
//  std::cin >> mc5;
//  std::cout << mc1 << mc2;
//  MascaPolicarbonat *mp1 = new MascaPolicarbonat(), *mp2 = new
//  MascaPolicarbonat();
//  MascaPolicarbonat *mp3 = new MascaPolicarbonat("elastic");
//  std::cin >> mp1 >> mp2;
//  std::cout << mp3;
//  Dezinfectant *d1 = new DezinfectantBacterii(100000000,
//  std::vector<string>(
//      {"sulfati non-ionici", "sulfati cationici", "parfumuri",
//  "Linalool", "Metilpropanol butilpentil"}),
//      std::vector({"lemn, sticla, metal, ceramica, marmura"}));
//  Dezinfectant *d2 = new DezinfectantVirusuri(50000000,
//  std::vector<string>(
//      {"Alkil Dimetilm Benzil Crlorura de amoniu",
//  "parfumuri", "Butilpentil metilpropinal"}),

```

```

//          std::vector({"lemn, sticla, ceramica, marmura"}));
//  Dezinfectant *d3 = new DezinfectantFungi(1400000, std::vector<string>(
//          {"Alkil Etil Benzil Crlorura de amoniu", "parfumuri",
// "Butilpentil metilpropinal"}),
//          std::vector({"sticla, plastic"}));
//  std::cout << d1->eficienta() << " " << d2->eficienta() << " " << d3->eficienta()
//  << "\n";
//  Achizitie *a1 = new Achizitie(26, 5, 2020, "PlushBio
//  SRL
//  ");
//  *
//  a1 += mp1; //se adauga masca de polycarbonat mp1 in lista de masti
//  achizitionate *a1 += (&mc1); //se adauga masca chirurgicala mc1 in lista
//  *a1 += d3; // se adauga dezinfectantu de fungi d3 in lista de dezinfectanti
//  achizitionati
//  Achizitie *a2 = new Achizitie(25, 5, 2020, "Gucci");
//  *a2 += d1;
//  *a2 += d2;
//  *a2 += d2;
//  Achizitie a3, a4(*a1);
//  a3 = *a2;
//  if (*a1 < *a2) {
//      std::cout << a1->nume() << " are valoarea facturii mai mica.\n";
//  } else if (*a1 == *a2) {
//      std::cout << a1->nume() << " si " << a2->nume() << " au aceasi valoare a
//  facturii.\n";
//  } else {
//      std::cout << a2->nume() << " are valoarea facturii mai mica.\n";
//  }
//  }
//  }
//}

```