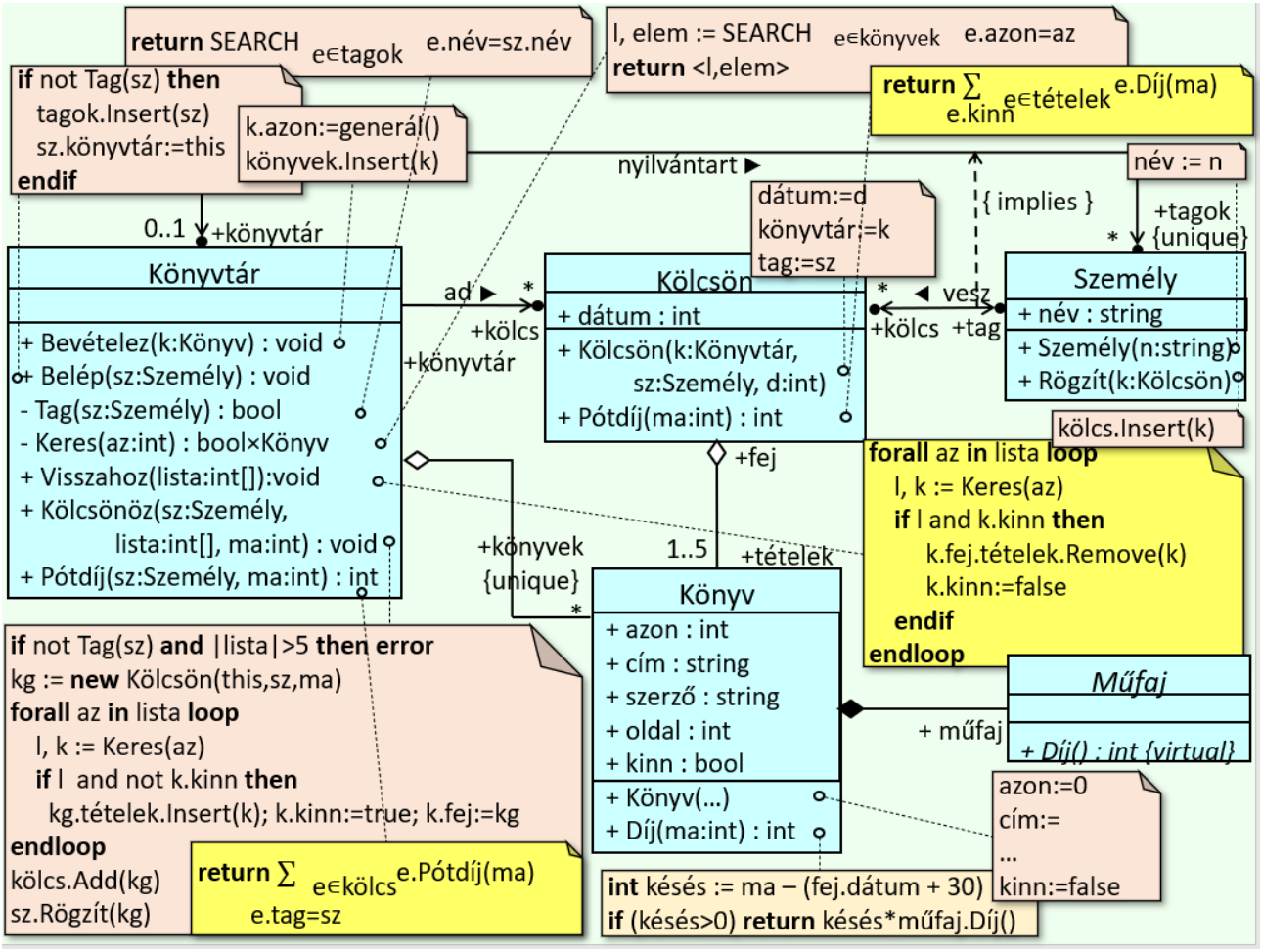
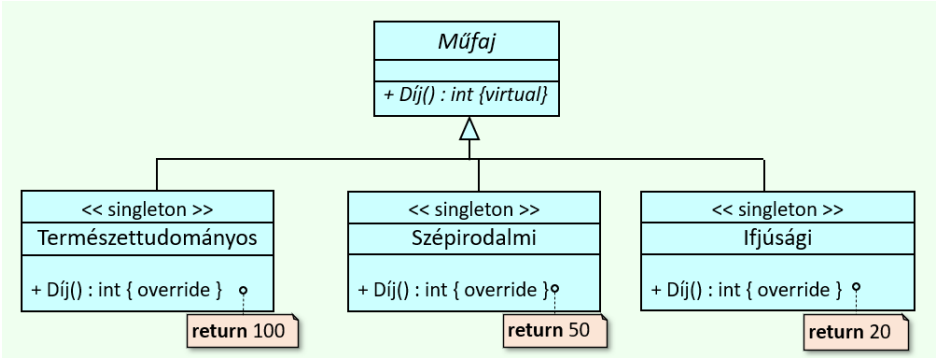
Egy könyvtár nyilvántartja a könyvtárba beiratkozott személyeket, és a kikölcsönözhető könyveit. Könyvtári tag az a személy lehet (ismerjük a nevét), aki beiratkozik a könyvtárba. Feltehetjük, hogy a név egyedi. Egy könyvtári tag egy alkalommal legfeljebb öt, a könyvtárban meglévő könyvet kölcsönözhet ki. Egy könyvnek ismert a címe, a szerzője, a műfaja és az oldalszáma, valamint van egy azonosítója, amit akkor rendelünk a könyvhöz, amikor a könyvtár könyvállományába kerül. A műfaj három féle lehet: természettudományi, szépirodalmi és ifjúsági. Az egyszerre kikölcsönzött könyveket több részletben is vissza lehet hozni, így egy kölcsönzés eseményhez tartozó könyvek listája folyamatosan csökkenhet. A kölcsönzési idő 30 nap. Egy könyv kölcsönzési pótdíja a kölcsönzés lejárati idejétől számított napok számának és a könyv műfaja által meghatározott együtthatójának (természettudományi könyvnél 100, szépirodalmi könyvnél 50, ifjúsági könyvnél 20) a szorzata.

Listázzuk ki a könyvtár valamennyi tagjának a nevét, és hogy a pillanatnyi kölcsönzéseik alapján mennyi pótdíjat kell fizetniük.

A terv osztálydiagramja:





Közepes szint:

Implementálja a feladat mellékelt modelljét a *sárga színű metódustörzsek nélkül*, majd a saját kódját a megadott tester.cpp fájllal szerkessze össze, és próbálja ki.

Implementálásnál az adattagok és a szerepnevek láthatósága legyen private vagy protected. A közepes szint teszteléséhez szükséges getterek elkészítése a feladat része. A szükséges getter metódusok a teszteléshez megadott teszt.cpp fájlból olvashatók ki. Ha egy gyűjtemény tartalmát kérdezzük le, akkor egy megfelelő típusú vector-t kell visszaadni. A getterek neve „get_" szöveggel kezdődjön, és az adott adattag, illetve szerepnév nevével folytatódjon!

A könyvtárba kerülő könyvek azonosítóját automatikusan generáljuk. Legyen a Könyvtár osztálynak egy változója, melynek értékét eggyel növeljük, amikor egy új könyvet felveszünk, és ez a sorszám legyen a könyv azonosítója. Az első könyv 1-es sorszámú legyen. Ezt a modellben a generál() függvény jeleníti meg.

A dátumokat az egyszerűség kedvéért egy nem negatív egész számmal ábrázoljuk. Két dátum különbsége az eltelt napok számát adja meg.

Az egyke osztályoknak legyen egy publikusan hívható instance() és destroy() metódusuk.

Ügyeljünk a tervben szereplő nevek pontos megvalósítására: kisbetű/nagybetű fontos, az ékezetes magánhangzók helyett az ékezet nélküli megfelelőjüket használják!

A destruktorok szabadítsák fel a dinamikusan létrehozott objektumokat, de figyeljünk arra, hogy egy objektum több különböző osztály gyűjteményében is szerepelhet, nehogy többször is meghívódjon a destruktoruk.

Miután feltöltötte a munkáját a TMS-be és úgy látja, hogy a programja működik, szóljon az egyik felügyelő tanárnak, hogy ellenőrizze a programját.

Jeles szint:

Implementálja a feladat mellékelt *teljes modelljét,* majd az *input.txt* szöveges állományban elhelyezett adatok alapján populálja azt fel objektumokkal, hajtsa végre az input fájlban leírt tranzakciókat, majd írassa ki a tagok pótdíját. A hibás eseteket kivétel dobással kezelje. A bemeneti fájl neve rögzített: „input.txt".

Az input fájl

- első sora tartalmazza a mai dátumot egy egész számmal ábrázolva.
- második sora tartalmazza a könyvtár könyveinek számát (nem negatív egész szám)
- majd ezt követően egy-egy sorban a megfelelő számú könyv adatai jönnek elválasztójelekkel tagolva: cím (elválasztójelek nélküli sztring), szerző (elválasztójelek nélküli sztring), oldalszám (természetes szám), műfaj (egy karakter: t-természettudományi, s-szépirodalmi, i-ifjúsági). (A könyv azonosítóját a program állítja elő, a fájlból érkező sorrend szerint kapnak sorszámot, 1-től indulva.)
- a következő sor tartalmazza a könyvtár tagjainak számát (nem negatív egész szám).
- ezután tagonkénti bontásban a tag neve és tranzakciói vannak megadva a következők szerint:
  - tag első sorában a neve (szóköz nélküli sztring) majd a tranzakcióinak száma (nem negatív egész szám) van megadva,
  - majd annyi sor következik, ahány tranzakciója volt az adott tagnak. A sor elején a „ki" vagy „vissza" szöveg adja meg, hogy kivett, vagy visszahozott könyveket.
  - kivétel esetén a következő adat egy egész szám, ez adja meg a kivétel dátumát, majd a könyvek azonosítóinak felsorolása következik, amelyeket ki szeretne venni.
  - visszahozáskor a sor további része csak a visszahozott könyvek azonosítóit tartalmazza.

Példa:

```
100
5
Lathatatlan_ember    Gardonyi_Geza           330   s
Richelieu            Anthony_Levi            256   s
Az_elo_bolygo        David_Attenborough      323   t
Istenek_kertje       Gerald_Durrell          260   s
Oz_a_csodak_csodaja  Frank_L_Baum            196   i
3
Pista    2
ki       67    3     1
vissza   3
Anna     3
ki       50    2     5
vissza   2
ki       70    4
Jani     1
ki       90    3
```

Kimenet: az egyes tagok neve és pótdíja. A program a konzolba írja ki. A tagokat abban a sorrendben kell kiírni, ahogyan a bemenetben szerepeltek, a név és a pótdíj között pontosan egy szóköz legyen. Ha a tagnak nincs pótdíja, akkor 0-át írjunk ki!

A fenti példa esetén a kimenet:

Pista 150
Anna 400
Jani 0

Miután feltöltötte a munkáját a TMS-be és úgy látja, hogy a programja működik, szóljon az egyik felügyelő tanárnak, hogy ellenőrizze a programját.

c:\Kleon_egyetem\000tantárgyak\2_felev\OEP_programozos\masodik_mintaZH\konyvtar\kolcson.h

```
#pragma once
#include "konyvtar.h"
#include "konyv.h"
#include "kolcson.h"
#include "mufaj.h"
#include "szemely.h"
#include <string>
#include <iostream>
#include <vector>

using namespace std;
class Mufaj;
class Konyvtar;
class Konyv;
class Szemely;

class Kolcson{
    private:
        int datum;
        Konyvtar* konyvtar;
        Szemely* tag;
        vector<Konyv*> tetelek;

    public:
        Kolcson(Konyvtar* k, Szemely* sz, int d): datum(d),konyvtar(k),tag(sz){}
        ~Kolcson(){}

        //**********


        vector<Konyv*> &get_tetelek(){ return tetelek; }
        int get_datum(){return datum;}

};
```

c:\Kleon_egyetem\000tantárgyak\2_felev\OEP_programozos\masodik_mintaZH\konyvtar\Konyv.cpp

```
#include "konyv.h"
 int Konyv::Dij(int ma)
        {
            int keses= ma -(fej->get_datum() + 30);
            if(keses>0)
            {
                return keses*mufaj->Dij();
            }
            return 0;
        }

Konyv:: ~Konyv(){
    delete mufaj;
}
```

c:\Kleon_egyetem\000tantárgyak\2_felev\OEP_programozos\masodik_mintaZH\konyvtar\konyv.h

```
#pragma once
#include "konyvtar.h"
#include "konyv.h"
#include "kolcson.h"
#include "mufaj.h"
#include "szemely.h"
#include <string>
#include <iostream>
class Kolcson;
class Konyvtar;
class Mufaj;
class Szemely;

using namespace std;


class Konyv{
    private:

        int azon, oldal;
        string cim, szerzo;
        bool kinn;
        Mufaj*mufaj;
        Kolcson* fej;

    public:
        Konyv()=default;
        Konyv(string cim, string szerzo,int oldal, Mufaj* mufaj):azon(0), szerzo(szerzo),oldal(oldal),mufaj(mufaj),cim(cim),
fej(0),kinn(false){}
        int Dij(int ma);
        ~Konyv();


        //**********
        void set_azon(int azon){this->azon=azon;}
        int get_azon(){ return azon;}
        bool get_kinn(){ return kinn;}
        void set_kinn(bool k) {kinn=k; }
        Kolcson* get_fej(){ return fej;}
        void set_fej(Kolcson*k ){fej=k;}
        string get_cim(){return cim;}
        string get_szerzo(){return szerzo;}
        int get_oldal(){return oldal;}
        Mufaj* get_mufaj(){return mufaj;}

};
```

c:\Kleon_egyetem\000tantárgyak\2_felev\OEP_programozos\masodik_mintaZH\konyvtar\konyvtar.cpp

```cpp
#include "konyvtar.h"

void Konyvtar::Bevetelez(Konyv* k)
{
    k->set_azon(c);
    c++;
    konyvek.push_back(k);

}

void Konyvtar::Belep(Szemely* &sz){
    if(!Tag(sz))
    {
        tagok.push_back(sz);
        sz->set_konyvtar(this);
    }

}

bool Konyvtar::Tag(Szemely* sz)
{
    for(Szemely*  e : tagok)
    {
        if(e->get_nev()==sz->get_nev())
        {
            return true;
        }

    }
    return false;
}

bool Konyvtar::Keres(int az,Konyv* &k)
{
    bool van=false;
    for(Konyv* e : konyvek)
    {
        if(e->get_azon()==az)
        {
            k=e;
            van=true;
            return true;
        }
    }
    return van;
}

void Konyvtar::Kolcsonoz(Szemely* &sz, vector<int> lista, int ma)
{
    if(!Tag(sz) && lista.size()>5) { throw exception();}
    Kolcson* kg = new Kolcson(this,sz,ma);
    Konyv* k;

    for(int az : lista)
    {
        bool van=false;
        van=this->Keres(az,k);
        if(van && !k->get_kinn())
        {
            kg->get_tetelek().push_back(k);
            k->set_kinn(true);
            k->set_fej(kg);
        }
    }
    kolcs.push_back(kg);
    sz->Rogzit(kg);

}
```

c:\Kleon_egyetem\000tantárgyak\2_felev\OEP_programozos\masodik_mintaZH\konyvtar\konyvtar.h

```h
#pragma once
#include "konyvtar.h"
#include "konyv.h"
#include "kolcson.h"
#include "mufaj.h"
#include "szemely.h"
#include <string>
#include <iostream>
#include <vector>
class Mufaj;
class Kolcson;
class Konyv;
class Szemely;

using namespace std;

class Konyvtar{

    private:
        vector<Kolcson*> kolcs;
        vector<Konyv*> konyvek;
        vector<Szemely*> tagok;
        int c;
    public:
        Konyvtar():c(1){}
        void Bevetelez(Konyv* k);
        void Belep(Szemely* &sz);
        bool Tag(Szemely* sz);
        bool Keres(int az,Konyv* &k);
        void Kolcsonoz(Szemely* &sz, vector<int> lista, int ma);



        vector<Konyv*> get_konyvek(){return konyvek;}
        vector<Kolcson*> get_kolcs(){return kolcs;}
        vector<Szemely*> get_tagok(){return tagok;}



};
```

c:\Kleon_egyetem\000tantárgyak\2_felev\OEP_programozos\masodik_mintaZH\konyvtar\main.cpp

```cpp
//ide tedd az a header fájlok includejait
#include "konyvtar.h"
#include "konyv.h"
#include "kolcson.h"
#include "mufaj.h"
#include "szemely.h"

#include <iostream>


int testCounter = 1;
bool jo = true;

void check(bool l)
{
    if(!l)
    {
        jo = false;
        std::cerr<<testCounter<<". teszt sikertelen."<<std::endl;
    }
    testCounter++;
}

int main()
{

    Konyvtar* konyvtar = new Konyvtar();

    Konyv* konyv1 = new Konyv("Cim1","Szerzo1",100,Termeszettudomanyos::instance());
    Konyv* konyv2 = new Konyv("Cim2","Szerzo2",200,Szepirodalmi::instance());

    Szemely* szemely1 = new Szemely("Szemely1");

    konyvtar->Bevetelez(konyv1);
    check(konyvtar->get_konyvek().size() == 1);
    check(konyvtar->get_konyvek()[0] == konyv1);
    konyvtar->Bevetelez(konyv2);

    check(konyvtar->get_konyvek()[0]->get_azon() == 1);
    check(konyvtar->get_konyvek()[0]->get_cim() == "Cim1");
    check(konyvtar->get_konyvek()[0]->get_szerzo() == "Szerzo1");
    check(konyvtar->get_konyvek()[0]->get_oldal() == 100);
    check(konyvtar->get_konyvek()[0]->get_kinn() == false);
    check(konyvtar->get_konyvek()[0]->get_mufaj() == Termeszettudomanyos::instance());

    konyvtar->Belep(szemely1);
    check(szemely1->get_konyvtar() == konyvtar);
    check(konyvtar->get_tagok().size() == 1);
    check(konyvtar->get_tagok()[0]->get_nev() == "Szemely1");

    konyvtar->Kolcsonoz(szemely1,{1,2},0);
    check(konyv1->get_kinn() == true);
    check(szemely1->get_kolcs()[0]->get_tetelek()[0] == konyv1);
    check(szemely1->get_kolcs()[0]->get_tetelek().size() == 2);
    check(konyv1->get_fej() == konyvtar->get_kolcs()[0]);


    check(konyv1->get_mufaj()->Dij() == 100);
    check(konyv2->get_mufaj()->Dij() == 50);

    Termeszettudomanyos::destroy();
    Szepirodalmi::destroy();
    Ifjusagi::destroy();

    if (!jo)
    {
        return 1;
    }
    else
    {
        std::cout << "Ok";
        return 0;
    }

}
```

c:\Kleon_egyetem\000tantárgyak\2_felev\OEP_programozos\masodik_mintaZH\konyvtar\mufaj.cpp

```cpp
#include "mufaj.h"

Termeszettudomanyos* Termeszettudomanyos::_instance=nullptr;
Ifjusagi* Ifjusagi::_instance=nullptr;
Szepirodalmi* Szepirodalmi::_instance=nullptr;
```

c:\Kleon_egyetem\000tantárgyak\2_felev\OEP_programozos\masodik_mintaZH\konyvtar\mufaj.h

```
#pragma once
#include "konyvtar.h"
#include "konyv.h"
#include "kolcson.h"
#include "mufaj.h"
#include "szemely.h"
#include <string>
#include <iostream>

using namespace std;

class Kolcson;
class Konyvtar;
class Konyv;
class Szemely;

class Mufaj{

    public:
        virtual int Dij()=0;
        virtual ~Mufaj()=default;
};

class Termeszettudomanyos : public Mufaj{

    private:
        static Termeszettudomanyos* _instance;
        Termeszettudomanyos(){}
        Kolcson* fej;
    public:
        static Termeszettudomanyos* instance()
        {
            if(_instance==nullptr)
            {
                _instance= new Termeszettudomanyos();
            }
            return _instance;
        }
        static void destroy()
        {
            if(_instance!=nullptr)
            {
                delete _instance;
            }
            _instance=nullptr;
        }
        int Dij() override
        {
            return 100;
        }
};

class Szepirodalmi : public Mufaj{

    private:
        static Szepirodalmi* _instance;
        Szepirodalmi(){}
    public:
        static Szepirodalmi* instance()
        {
            if(_instance==nullptr)
            {
                _instance= new Szepirodalmi();
            }
            return _instance;
        }
        static void destroy()
        {
            if(_instance!=nullptr)
            {
                delete _instance;
            }
            _instance=nullptr;
        }
        int Dij() override
        {
            return 50;
        }
};


class Ifjusagi : public Mufaj{

    private:
```

```
        static Ifjusagi* _instance;
        Ifjusagi(){}
        Kolcson* fej;
    public:
        static Ifjusagi* instance()
        {
            if(_instance==nullptr)
            {
                _instance= new Ifjusagi();
            }
            return _instance;
        }
        static void destroy()
        {
            if(_instance!=nullptr)
            {
                delete _instance;
            }
            _instance=nullptr;
        }
        int Dij() override
        {
            return 20;
        }
};
```

c:\Kleon_egyetem\000tantárgyak\2_felev\OEP_programozos\masodik_mintaZH\konyvtar\szemely.cpp

```
#include "szemely.h"

void Szemely::Rogzit(Kolcson*k){
    kolcs.push_back(k);
}
```

c:\Kleon_egyetem\000tantárgyak\2_felev\OEP_programozos\masodik_mintaZH\konyvtar\szemely.h

```
#pragma once
#include "konyvtar.h"
#include "konyv.h"
#include "kolcson.h"
#include "mufaj.h"
#include "szemely.h"
#include <string>
#include <iostream>
#include <vector>

using namespace std;
class Kolcson;
class Konyvtar;
class Mufaj;
class Konyv;



class Szemely{
    private:
        string nev;
        vector<Kolcson*> kolcs;
        Konyvtar* konyvtar;

    public:
        Szemely(string n):nev(n){}
        void Rogzit(Kolcson*k);


        void set_konyvtar(Konyvtar* k){ konyvtar = k;}
        string get_nev(){ return nev;}
        vector<Kolcson*> get_kolcs(){return kolcs;}
        Konyvtar* get_konyvtar(){return konyvtar;}


};
```