8. LABOR
ÖRÖKLÉS ÉS POLIMORFIZMUS

Általános információk

Összesen 1 db iMSc pont jár az összes feladat hibátlan megoldására.

Kötelező feladatok

1. Gumimaci: UML

Egy Mindenízű Gumimacikkal kereskedő MLM **cég partneradatbázisát** karbantartó szoftver tervezési feladatát kaptuk meg. A cég partnerei (*Partner*) **magánszemélyek** (*Person*) és **vállalkozások** (*Company*) is lehetnek. (*A szokásos karaktertömb helyett dologzzunk c++ stringekkel, ami a sztenderd kódkönyvtárban található osztály, vagyis minden környezetben rendelkezésünkre áll. Használata amiatt lehet könnyebb, hogy átveszi tőlünk a memória menedzsment feladatait.* A magánszemélynek van *vezetékneve (firstName: std::string)*, *keresztneve (lastName: std::string)*, *adóazonosító jele (taxNumber: std::string)*. A cégnek van *cégneve (businessName: std::string), adószáma (VATIN (std::string: Value Added Tax Identifier Name)*. Mindkét típusú partnernek van egy 2000 karakteres egyedi *titkosítási kulcsa (privateKey: char[2001])*, valamint egy egyedi *azonosítója* (*id: long*).

Közösen vitassátok meg a következő kérdéseket:

- Mutassátok meg, hogy itt **öröklésről** van szó!
- Mik az ősosztály, illetve a leszármazott osztályok **adattagjai**? Mik a **láthatóságuk**? Honnan tudjuk, hogy melyik kihez tartozik?
- Mit tud az ősosztály **konstruktora**? Mit tud a leszármazottaké? Ősosztály konstruktorának hívása a leszármazott konstruktorban lehetséges?
- Tervezz **getter** függvényeket az egyedi azonosító, a vezeték-és keresztnév, valamint a cégnév lekérdezésére. Melyik hol lesz?

Rajzolj UML diagrammot a leírás alapján!

2. Gumimaci: implementálás

A közösen tervezett UML diagram alapján implementáld az osztályokat a kiindulási kód segítségével! Írj tesztet minden megírt függvényre!

3. CallOfC++: Weapon - UML

Egy új FPS (First Person Shooter) játékon, a CallOfC++-on dolgoztok csapatoddal és te kaptad a fegyver (*Weapon*) modellezését.

A fegyver jellemzői:

- van 0-tól 100-ig terjedő sebzése (*damage*: *unsigned*), annak pedig gettere és settere
- alapértelmezetten 10-et sebez
- ki lehet íratni a konzolra a tulajdonságait diagnosztikai függvénnyel (*toString()*)
- lehet használni (*use()*: visszaadja az eggyel csökkentett sebzést (mert kopik a fegyver))

Ezen paraméterek segítségével rajzoljatok közösen UML diagrammot!

4. CallOfC++: Weapon – implementálás

Az előző feladat UML diagrammját implementáljátok egy CallOfCpp nevű solution keretén belül!

5. CallOfC++: Knife, Pistol, Railgun - UML

Az egyik közösségi médiás felmérésetekből kiderült, hogy a célközönség nem csak egy egyszerű fegyverre vágyik, hanem mindjárt háromra:

1) Kés (*Knife*), ami rendelkezik egy 0 és 1 közötti sebzési koefficienssel (*sharpness: double*), ami 95%-ára csökken minden egyes használatkor, viszont lehet növelni 10%-kal élesítéskor (*sharpen()*). Alapértelmezetten 30 egységet sebez és csak 80%-osan éles. A tényleges sebzés *sharpness*damage*.
2) Pisztoly (*Pistol*), aminek van pisztolytár kapacitása (*clipSize*), amiben *bulletsInClip* db töltény van. Ezeken kívül van egy tölténytáskája, amiben *totalBullets* db tölténye van, innen újra tudja tölteni (*reload()*) a pisztolytárat. Alapértelmezetten 60-at sebez, 12 töltény fér el a tárban, teli tárral kezd és ezen kívül kap még 36 töltényt.
3) *Railgun* (elektromágneses fegyver), amivel lövedék kilövés hatására a 0 és 100 közötti energiájából (*energy*) 10 egységet veszít, viszont újra lehet tölteni 8 energiaegységgel (*recharge()*). Alapértelmezetten 90 egységet sebez és 100-as energia szintről indul. Kilövéshez legalább 10 energiaegység szükséges.

Minden privát tagváltozót el lehessen érni getterrel és setterrel és lehessen beállítani konstruktorral. A fenti leírás alapján rajzolj UML diagrammot!

6. CallOfC++: Knife, Pistol, Railgun - implementálás

Az előző feladatban megrajzolt UML diagram alapján implementáld a rajta szereplő osztályokat!

Az megírt osztályok teszteléséhez használd fel a *callOfCppTest.cpp*-ben szereplő kódot!

Ha nem marad rá laboron idő, otthon fejezd be!

Az elvárt kimenet a következő:

```
Weapon; damage can be caused=100
Weapon<|Knife; damage can be caused when stabbing=24; sharpness=0.8
Weapon<|Pistol; damage of a bullet=60; clipSize=12; bulletsInClip=12; totalBullets=36
Weapon<|Railgun; damage of shooting=90; energy=100

Damage caused with excalibur: 99
Damage caused with knife: 24
Damage caused with pistol: 60
Damage caused with railgun: 90

Weapon; damage can be caused=99
Weapon<|Knife; damage can be caused when stabbing=20.064; sharpness=0.836
Weapon<|Pistol; damage of a bullet=60; clipSize=12; bulletsInClip=12; totalBullets=35
Weapon<|Railgun; damage of shooting=90; energy=98
```

Gyakorlófeladatok

- Vállalati alkalmazottak
- Nemzetközi vadásztársaság
- Teknőskaland
- Mikulás járművei
- Kanadai jégkorong-válogatott reklámcikkei
- Madárgyűjtemény eszmei értéke
- Medvemenhely tartási költsége
- Bútorbolt nyilvántartórendszere
- Szerepjáték