

9. LABOR

VIRTUÁLIS TAGFÜGGVÉNYEK, ABSZTRAKT OSZTÁLYOK, HETEROGÉN KOLLEKCIÓ

Általános információk

1 db iMSc pont jár a 4. feladat megvalósításáért.

Kötelező feladatok

1. Számítógép-alkatrészek

Írj egy programot *Equipment* néven, amely különböző számítógép-alkatrészeket tart nyilván!

- Monitorok (*Monitor*) esetén a sorozatszámot (*serialNumber*), az életkort (*age*) és az árat (*price*) kell nyilvántartani. Nyomtató (*Printer*) esetén a sorozatszámot (*serialNumber*), az árat (*price*) és a patron árát (*cartridgePrice*).
- Az alkatrészeket rendezzük tömbbe. Feltételezhetjük, hogy *maxEquipment* alkatrésznél többet nem kell beolvasnunk! Keressünk közös *őosztályt* az *alkatrészeknek*!
- A tömböt bejárva írassuk ki az eddig nyilvántartott elemek adatait (*print*)! Mi történik, ha az *őosztály print* függvénye nem virtuális? Próbáljuk ki végiglépkedve debuggerrel!
- Nézd végig debuggerben, hogy melyik függvények hívódnak meg a kiíratáskor (de teszt kiíratás is megengedett, mint pl. „*Printer()* *lefut*”).

2. CallofC++: Fegyverek refaktorálása

Alakítsd át a múlt héten elkészített CallofC++ osztályokat (kiindulást megtalálod a portálon, CallofCpp_01) a következőképp:

- Oldd meg, hogy a *use()* és a *toString()* függvények definícióját minden esetben a leszármazottól vegye a program (late binding)
- A *use()* és a *toString()* függvényeknek ne legyen definíciója a *Weapon* osztályban, hiszen nem szükséges
- Foglald össze röviden, hogy mit is csináltál az előző két pontban és mi a következmény!
- Teszteld le a megoldásodat a *callofCppTest* átírásával: **Tárold a fegyvereket közös tömbben (heterogén kollekció), a kiírásokat cseréld le úgy, hogy for ciklussal történjen a heterogén kollekció felhasználásával!**
- Készítsd fel a fegyverek osztályait arra, hogy esetleges dinamikus memóriahasználat esetén minden esetben lefussanak a leszármazottak destruktora. Teszteld is le!

Ha nem emlékeznél pontosan, hogy melyik függvény mit csinál, nézd meg az előző laborfeladatsort.

3. CallofC++: Player - UML

A főnökeid megnézték a munkád és hosszas gondolkodás után rájöttek, hogy csak úgy lehet átütő siker a játék, ha vannak benne játékosok is (*Player*).

A játékosra a következők jellemzők:

- Van neve (*name*), valamint életereje (*health: unsigned*)
- Le lehet kérdezni, hogy még él-e (*isAlive()*)
- Lehetnek nála fegyverek (*weapons: Weapon***), de csak korlátozott számban (*maxWeaponsCount: const unsigned*). Tároljuk, hogy egy adott pillanatban hány fegyver van nála (*weaponsCount*). Mindig van egy kiválasztott fegyvere, aminek tároljuk a *weapons*-beli indexét (*selectedWeaponIndex*).
- Fel tud venni egy fegyvert (*equipWeapon(Weapon*)*)
- El tudja dobni a kiválasztott fegyvert (*dropSelected()*)
- Át tud váltani a következő és az előző fegyverre (*switchToNextWeapon()*, *switchToPreviousWeapon()*)
- A játékost meg lehet sebezni (*takeDamage(unsigned)*).
- A kiválasztott fegyverrel meg tud támadni egy másik játékost (*attack(Player&)*).
- A konstruktorban lehessen megadni a nevet, életerőt, maximálisan tartható fegyverek számát, valamint kezdő fegyvert, de egyik se legyen kötelező.



1. ábra Használj heterogén kollekciót!

A következő függvények visszatérnek igazgal, ha sikeres az adott művelet, ellenkező esetben hamissal: *equipWeapon*, *dropSelected*, *switchToNextWeapon*, *switchToPreviousWeapon*, *takeDamage*, *attack*. Minden privát tagváltozóhoz készíts gettert és settert (*getName()*, *setName(std::string)*, *getWeaponsCount()*, *getSelectedWeapon(): Weapon**, *getHealth()*).

Gondold végig, miért *Weapons*** (mutatók tömbje) tárolja a *Player* a fegyvereket (mikor működnek a virtuális függvények)? Ezt beszéljétek át közösen.

A CallofC++ összes osztálya alapján rajzolj UML diagrammot! A *Knife*, *Pistol* és *Railgun* osztályoknál elegendő csak az öröklést feltüntetni.

4. CallOfC++: Player – implementáció és teszt

- Nézd végig és értsd meg az osztályok implementációját! Nézzétek át közösen és értsétek meg sorról sorra az equipWeapon függvényt!
- Önállóan valósítsd meg a **Player**: :attack függvényt a benne leírt segítségek felhasználásával!
- A callOfCppTest fájlban valósítsd meg az elkészült elemek tesztelését, a TODO utasításokban leírtaknak megfelelően úgy, hogy utána a leírt, elvárt kimenetet kapjuk.

A callOfCppTest.cpp-ben található eheti teszt elvárt kimenete:

Init state

```
Player#1 is alive; health: 100; has 2 weapon(s); selected Knife  
Player#2 is alive; health: 50; has 1 weapon(s); selected Railgun  
Enemy#1 is alive; health: 100; has 0 weapon(s); selected no weapon
```

Player#1 switches to next weapon:

```
Player#1 is alive; health: 100; has 2 weapon(s); selected Pistol  
Player#2 is alive; health: 50; has 1 weapon(s); selected Railgun  
Enemy#1 is alive; health: 100; has 0 weapon(s); selected no weapon
```

Player#2 attacks Enemy#1:

```
Player#1 is alive; health: 100; has 2 weapon(s); selected Pistol  
Player#2 is alive; health: 50; has 1 weapon(s); selected Railgun  
Enemy#1 is alive; health: 10; has 0 weapon(s); selected no weapon
```

Player#2 drops selected weapon:

```
Player#1 is alive; health: 100; has 2 weapon(s); selected Pistol  
Player#2 is alive; health: 50; has 0 weapon(s); selected no weapon  
Enemy#1 is alive; health: 10; has 0 weapon(s); selected no weapon
```

Player#2 attacks Enemy#1:

Player#1 killed Enemy#1 with a(n) Pistol

```
Player#1 is alive; health: 100; has 2 weapon(s); selected Pistol  
Player#2 is alive; health: 50; has 0 weapon(s); selected no weapon  
Enemy#1 is not alive; health: 0; has 0 weapon(s); selected no weapon
```

Gyakorló feladatok

- [Vállalati alkalmazottak](#)
- [Nemzetközi vadásztársaság](#)
- [Teknőskaland](#)
- [Mikulás járművei](#)
- [Kanadai jégkorong-válogatott reklámcikkei](#)
- [Madárgyűjtemény eszmei értéke](#)
- [Medvemenhely tartási költsége](#)
- [Bútorbolt nyilvántartórendszere](#)
- [Szerepijáték](#)