2022. 05. 19. 10:00

Feladat –NagyZH– FONTOS INFÓK

- A feladat során alkalmazd a megtanult objektum-orientáltsági elveket, figyelj a konstansok és referenciák megfelelő használatára! A javító képes ezeket is ellenőrizni, ami eredményezhet nem forduló kódot. Például, ha egy függvény nem const, pedig annak kellene lennie, lehet, hogy nem fordul majd le a kód.
- A megadott példakódon ne módosíts, hacsak a feladat nem kéri! A megoldásnak ezen fájlokkal kell mennie, hiszen az ellenőrzés során a moodle biztosítja őket.
- Figyelj a kiírás megfelelő formátumára, szóközökre, új sorokra! Az automata javító csak a tényleges kimenetet látja, nem tudja mit akartál.
- Ügyelj arra, hogy minden lefoglalt memória kerüljön megfelelően felszabadításra!
- Minden pont értékeléséhez szükséges, hogy az adott ponthoz tartozó define szerepeljen valamelyik header fájlban (#define PART1 az 1. feladathoz, #define PART4 a 4. feladathoz, stb., a **megoldott_feladatok.h** kifejezetten ezek megadására van, célszerű azt használni).
- A header fájlokban csak a header guard-on belülre dolgozz!
- Csak olyan kódot tölts fel moodle-be, ami nálad fordul. Ami nálad nem fordul, a moodle-ben sem fog.
- A fájlokat nem tömörítve kell feltölteni, hanem önmagukban (drag-and-drop-pal egyszerre be lehet húzni az összeset).

2022. 05. 19. 10:00

Feladat – NagyZH

- A feladat elkezdéséhez a mellékelt projekt tartalmaz kódokat. A megadott **main.cpp** fájlt módosítani nem lehet, de segít a tesztelésben.
- A feladatban egy számítógépes alkatrészeket áruló bolt kezdetleges rendszerét kell elkészíteni.
- A megoldás során figyelj a tanult elvekre (konstans, referencia, memóriakezelés).
- Az 1a és 1b feladatok kötelezőek, de a többi között is lehet ráépülés.

A feladatok:

1. **Absztrakt ősosztály**
    a. Készíts egy **Alkatresz** osztályt, amely az absztrakt ősosztály lesz az alkatrészek tárolásához. Ne felejtsd el a megfelelő destruktort sem. Az osztály tárolja az alkatrész gyártóját és típusát (szövegek), amelyeket a konstruktor ilyen sorrendben meg is kap. Mindkettőhöz legyen getter (*getGyarto*, *getTipus*). **(3 pont)**
    b. Az osztályban legyen 1 tisztán virtuális függvény *ar* metódus, amely paramétert nem vár, nem módosítja a belső adatokat, és visszatér az alkatrész árával (egész). **(1 pont)**
    c. Az osztályban legyen egy virtualis *kiir* metódus, amely a standard kimenetre kiírja az alkatrész gyártóját, típusát, valamint az árát, sortöréssel a végén (lásd minta-stdout.txt). Az adatokon nem módosít. **(2 pont)**
2. Származtass az **Alkatresz** osztályból egy **Processzor** osztályt, ami egy processzort jelöl, új adatként tárolja a processzor órajelét és a magok számát (egészek). Legyen hozzájuk getter is (*getOrajel*, *getMagok*). A konstruktor paraméterben várja az ősosztály két adatát, és a két új adatot is. Az *ar* metódus adja vissza a processzor árát az adatait alapján: órajel * 11 + magok száma * 16000. **(2 pont)**
3. Származtass az **Alkatresz** osztályból egy **Memoria** osztályt, ami egy memória adatait tárolja, új adatként tárolja a memória kapacitását és sebességét (egészek). Legyen hozzájuk getter is (*getKapacitas*, *getSebesseg*). A konstruktor paraméterben várja az ősosztály két adatát, és a két új adatot is. A kapacitás és a sebesség legyen elhagyható a konstruktorból, ebben az esetben a kapacitás legyen 16, a sebesség pedig 3200. Az *ar* metódus adja vissza a memória árát az adatait alapján: kapacitás * 800 + sebesség * 7. **(2 pont)**
4. **Egyéb alkatrész osztály:**
    a. Származtass az **Alkatresz** osztályból egy **EgyebAlkatresz** osztályt, ami extra adatként tárolja még az alkatrész leírását (szöveg), valamint az árát. A leíráshoz legyen getter (*getLeiras*). A konstruktor paraméterben várja az ősosztály két adatát, és a két új adatot is. Az *ar* metódus adja vissza a tárolt árat. **(2 pont)**
    b. Az osztály írja felül az ősosztály *kiir* metódusát úgy, hogy jelenjen meg a leírás is benne (a három korábbi érték mellett, lásd minta-stdout.txt). **(2 pont)**
5. **Készítsd el a következő operátorokat az Alkatresz osztályhoz:**
    a. **<** operátor, amelynek bal oldalán egy alkatrész van, jobb oldalán egy egész szám (egy ár), és akkor tér vissza igazzal, ha az alkatrész ára kisebb a jobb oldali értéknél, **(2 pont)**

   b. **<<** operátor, amivel az alkatrész tartalma tetszőleges kimenet folyamra kiírható, a kimenet ugyanazt tartalmazza, ugyanolyan formátumban, mint a *kiir* metódus, csak most sortörés nélkül, **(3 pont)**
   c. **!** operátor, amely visszaadja az alkatrész gyártóját és típusát egy szövegként (gyártó + szóköz + típus). **(2 pont)**

6. **Statikus adattag az Alkatresz osztályhoz:**
   a. Adja egy statikus egész változót az **Alkatresz** osztályhoz, amely egy értékhatárt tárol. A kezdeti értéke legyen 450 000, és legyen hozzá getter és setter (*getErtekhatar*, *setErtekhatar*), **(2 pont)**
   b. Legyen az osztálynak egy *teljesAr* metódusa, amelyik az értékhatárt is figyelembe veszi. Amennyiben az alkatrész ára az értékhatár alatt van, úgy ezzel az árral térjen vissza. Amennyiben meghaladja ezt a határt, akkor az eredeti árra még 14000 adót rászámolnak, ezt hozzá kell adni. Amennyiben az értékhatár kétszeresét is túl lépi, akkor 32000 adót számolnak rá az eredeti árra. **(2 pont)**
7. Származtass az **Alkatresz** osztályból egy **Csomag** osztályt, amely valamennyi darab, tetszőleges típusú alkatrészt köt össze egy kedvezményes csomagban.
   a. Hozd létre a **Csomag** osztályt, amely tároljon tetszőleges mennyiségű alkatrészt (a gyerekosztályokat). A csomagban lévő alkatrészek számát a konstruktor kapja meg, később már nem változik (itt célszerű is létrehozni a megfelelő tömböt). A konstruktor más paramétert nem kap, az ősosztályban a gyártó és típus adatok üres szövegek legyenek. Készíts el a destruktort is, amely az alkatrészeket tároló tömböt felszabadítja, de magukat az egyes alkatrészeket nem kell. Legyen az osztálynak egy *darabszam* metódusa, amely visszaadja az éppen tárolt objektumok számát (amennyit már hozzáadtunk). Az ar metódust is ki kell fejteni, hogy lehessen tesztelni. Egyelőre adjon vissza 0-t. **(3 pont)**
   b. Legyen az osztálynak egy *beallit* metódusa, amely megkap egy alkatrészt (**Alkatresz** mutató), és beilleszti a csomagba. Az osztály tartsa számon, hogy mennyi alkatrészt adtunk már hozzá, és mindig a következő helyre tegye a következőt. Ha már betelt a csomag (nincs hely több alkatrésznek), akkor ne csináljon semmit a függvény. **(4 pont)**
   c. Fejtsd ki az osztály egy *ar* metódusát, hogy határozza meg a csomagban lévő alkatrészek árának összegét, és adja vissza ennek az összegnek a 90%-át. **(2 pont)**
8. Készíts egy sablonos **Kupon** osztályt, ami egyedi, típushoz kötött kedvezményt tárol az alkatrészekre:
   a. Készíts egy **Kupon** osztályt egy sablon paraméterrel. Az osztálynak egyetlen adattagja legyen, egy kedvezmény (egész szám), amit a konstruktor paraméterként meg is kap, és eltárol. Legyen hozzá getter (*getKedvezmeny*). **(2 pont)**
   b. Készíts az osztályba egy *alkalmaz* metódust, amely paraméterben egy tetszőleges alkatrészt kap mutatóként. A függvény döntse el (célszerűen dynamic_cast segítségével), hogy a paraméterben kapott alkatrész a sablon paraméter által megadott gyerekosztályra mutat-e. Ha igen, akkor adja vissza az alkatrész árát a kedvezményt kivonva. Ha nem, akkor egyszerűen adja vissza az alkatrész eredeti árát. Feltehetjük, hogy az osztály sablon paramétere az **Alkatresz** egyik gyerekosztálya lesz. **(4 pont)**