Korszerű Programozási Technikák

NagyZH – 2017.12.06. Név: Neptun kód:

A feladat egy italokat forgalmazó vállalat napi összesített rendeléseinek feldolgozása és a cég különböző alosztályainak értesítése a feldolgozási eredményekről. Adottak az alábbi osztályok:

- **Order**: egy rendelést reprezentáló osztály, a következő adattagokkal: *orderNumber* (rendelés azonosító, **int**), *status* (rendelés státusza, **string**), *itemNumber* (termék azonosító, **int**), *quantity* (rendelt mennyiség, **int**). Az adattagok konstruktoron keresztül állíthatók be, illetve getterekkel kérhetők le. A rendelés státusza és mennyisége setterekkel módosítható is. A *print* metódus pedig kiírja a konzolra a rendelés adatait,
- **Item**: a vállalat egy terméket reprezentáló osztály, tároljuk a termék azonosítóját (*itemNumber*, **int**), nevét (*itemName*, **string**) és az elérhető raktárkészletet (*stock*, **int**). Az adattagok a konstruktorral állítható be és getterekkel lekérhetők. A *decreaseStock* metódus a termék raktárkészletét csökkenti a paraméterben megkapott értékkel,
- **OrderObserver**: a programban ezt az interfészt használjuk az alosztályok értesítésére egy virtuális *orderProcessed* függvény megvalósításával, ami paraméterben megkapja a feldolgozott **Order** objektumot,
- **OrderManager**: a rendelések kezelését megvalósító osztály. Az elérhető termékeket az *items* asszociatív tömbben tároljuk, ez a konstruktorban feltöltésre kerül a *populateItems* függvénnyel.

Rendelések beolvasása

Az első részfeladat az **orders.txt** fájlban megadott rendelés rekordok beolvasása és tárolása az **OrderManager** osztályban. A fájlban minden sor egy rendelést reprezentál a következő adatokkal: azonosító, státusz, termék azonosító, rendelt mennyiség. A fájlban minden sor megfelelő adatot tartalmaz, nem kell ellenőrizni.

1. Az **OrderManager** osztályba hozz létre egy asszociatív tömböt, aminek a kulcsa a rendelés azonosító, az érték pedig az **Order** objektum, valamint készíts egy *readOrders* metódust, ami egy fájlnevet vár paraméterként és az alábbiakban részletezett módon beolvassa a fájlból a rekordokat és eltárolja a **map**-ben, *(3 pont)*
    a. a ***NEW*** státuszú sorokat szúrd be a tárolóba *(1 pont)*
    b. a ***MODIFIED*** státuszú sorok egy korábban ***NEW*** státusszal felvett rendelés módosítását tartalmazza, a rekordban csak a megrendelt mennyiség változik. A már beolvasott rekordot keresd meg a tárolóban és módosítsd a korábbi mennyiséget az újra (egy rendelést többször is lehet módosítani), *(2 pont)*
    c. a ***DELETED*** státuszú sorok egy korábban felvett rendelés törlését jelzik. A már beolvasott (esetleg módosított) rekordot keresd meg és töröld a tárolóból (egy rendelés törlése után már nem szerepel rekord ahhoz a rendelés azonosítóhoz), *(2 pont)*
    d. a termék azonosító csak az 1-5 intervallumból kerülhet ki, amelyik rendelésnél ezen kívül esik a termék azonosító, akkor írj ki hibaüzenetet és ne tárold el a rekordot, *(1 pont)*
    e. + a fenti hibaüzent mellett egy az **OrderManager** osztályban létrehozott saját kivétel osztályt is használj úgy hogy a beolvasás során egy listába gyűjtsd össze a hibás rendelések azonosítóit és a beolvasás végén ezt a listát add át a kivétel osztálynak és dobd el a kivételt (amennyiben volt hiba), ami a példakimenetben megadott módon írja ki a hibát. A *main* függvényben kezeld az esetleges kivételt a *readOrders* függvényhívás körül. *(3 pont)*

Rendelések feldolgozása

2. Az **OrderManager** osztályba készíts egy paraméter nélküli, **void** *manageOrders* metódust, ami az alábbi módon feldolgozza a beolvasott rendeléseket:
    a. készíts egy *processOrders* függvényt, amiben menj végig a tárolt rendeléseken. Ha a rendelésben szereplő mennyiség nem több mint az *items*-ben lévő, a megadott termékhez tartozó aktuális készlet, akkor a rendelés státuszát állítsd át „***PROCESSED***”-re és a raktárkészletet csökkentsd a rendelés mennyiségével (*decreaseStock*). Ha nem elég a készlet, akkor a rendelés státuszát állítsd át „***REJECTED***”-re (figyelj arra, hogy a rendelés és termék objektumok megfelelő módosításához referenciaként kell lekérni őket az asszociatív tárolóból pl.: `Order& order = orders[key]`, különben másolódnak és a módosítások nem látszódnak a függvényen kívül). A *manageOrders* függvényben hívd meg a *processOrders*-t, *(4 pont)*
    b. + a *manageOrders* függvényen belül egy külön szálon indítsd el a *processOrder* metódust (használhatod az objektum tagfüggvényének hívása szintaktikát vagy lambda függvényként is meghívhatod). A *processOrders* függvény minden iterációban aludjon 50 milliszekundumot, *(3 pont)*
    c. + készíts egy lambda függvényt a *manageOrders* függvényen belül, ami 200 milliszekundumonként kiírja, hogy a *processOrders* függvény dolgozik, addig, amíg be nem fejezi a rendelések feldolgozását (használj kölcsönös kizárást, megfelelően módosítsd a *processOrders* függvényt is), indítsd el a lambda függvényt egy külön szálon. Figyelj a szálak megfelelő visszacsatolására a fő szálhoz. *(5 pont)*

Alosztályok értesítése és rendelések kiíratása

3. A rendelések feldolgozásáról értesítést kapnak a vállalat kiszállítással illetve ügyfelekkel foglalkozó alosztályai is. Az értesítések kezeléséhez az **Observer** mintának megfelelően egészítsd ki az **OrderManager** (**observable**) osztály egy listával, amiben tárolod az **observer**eket (**OrderObserver***). Hozd létre az **OrderObserver** interfész alapján az alábbi konkrét megfigyelő osztályokat és az **OrderManager** konstruktorában szúrj be a megfigyelő listába egyet-egyet belőlük:
    a. **CustomerServiceOrderObserver**: az ügyfelekkel foglalkozó alosztályt értesítő osztály. Az *orderProcessed* metódusban minden esetben írasd ki a példa kimenetben lévő értesítő szöveget, *(2 pont)*
    b. **ShippingOrderObserver**: a szállítással foglalkozó alosztályt értesítő osztály. Az *orderProcessed* metódusban csak akkor írasd ki a példa kimenetben szereplő szöveget ha a rendelés „***PROCESSED***” állapotú, *(2 pont)*
4. Készíts az **OrderManager** osztályba egy *printOrders* metódust, ami a kiírja a rendeléseket az **Order** objektumok print metódusának segítségével (használj **range-based loop**-ot). A kiíratót függvényt hívd meg a *manageOrders* metódusban a rendelések feldolgozása után. *(1 pont)*
5. Az **OrderManager** osztályba hozz létre egy *notifyObservers* metódust, ami egy **Order** referenciát vár. A függvényben menj végig a megfigyelők listáján és értesítsd mindegyiket a paraméterben megkapott rendelés objektummal. A *notifyObservers* metódust a *processOrders* függvényen belül minden iterációban hívd meg, *(2 pont)*
6. Az **OrderManager** osztály destruktorában gondoskodj arról, hogy a dinamikusan foglalt adatok (megfigyelők) törlése megtörténjen. *(1 pont)*