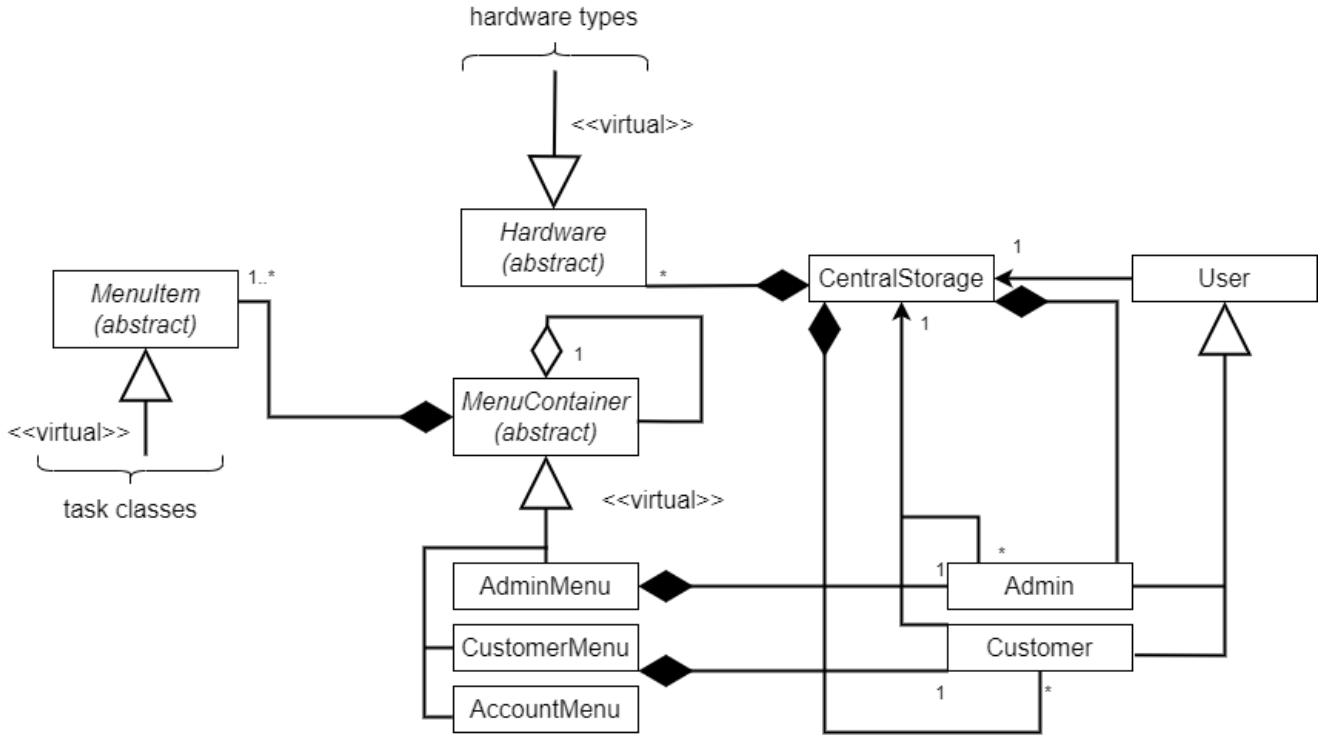
Nagy házi terv és dokumentáció

Webshop és futárszolgálat

Általános leírás

A kész program számos osztállyal, attribútummal, függvénnyel dolgozik, a teljes osztálydiagram értelmezése időbe telhet, ezért egy egyszerűsített diagramon fogom bemutatni az osztályok működését. Ez a diagram tartalmaz minden olyan fontos osztályt, amik képet adnak a program működéséről. A teljes osztálydiagram a dokumentáció végén található.



Osztályok

MenuContainer

A MenuContainer absztrakt osztály tartalmazza az egységes megjelenítéshez, és kezeléshez szükséges függvényeket. Az osztály segítségével lehet a menüt, adatbeviteli oldalakat kirajzolni, itt kapnak helyet a felhasználói bevitel függvényei is. Az osztály legfontosabb függvényei a Loop(), és a MainMenu() függvények, ezek alkalmazása adja az egész kezelés, feladatok, kezdő logikáját, kezdőpontját. Kapcsolatok szempontjából, ez az osztály egy absztrakt osztály, az AdminMenu, CustomerMenu, AccountMenu leszármazott osztályok létrehozásával férünk hozzá a függvényekhez. Ezek a leszármazott osztályok csupán a személyreszabott feladatokban különböznek, működésben ugyan azt kell tudniuk, maximum máshogy kell ezt elérni, ezért absztrakt az ősosztály. Az osztályban van, egy saját osztályra mutató aggregáció, ez pontosan egy saját osztály pointer, neve nextmenu, ami elég árulkodó, azért került bele, mert ennek segítségével tudunk a további menükre lépni. Ahol

létrehoztuk az osztályt ott ezt meg is szüntetjük, de előtte a következő menüre tudunk lépni, ugyanis a nextmenu alaposztály pointer már tartalmazza a következő dinamikusan foglalt menüt. Ezenfelül, ide csatlakozik még a MenuItem absztrakt osztály is egy kompozícióval. Minden menü típus létrehozza magának a kellő MenuItemeket, de amint megszűnik az adott Menü, a MenuItemek sem kellenek már többé. MenuItemek dinamikus heterogén kollekcióval vannak implementálva, tehát tetszőleges számú MenuItemet létre lehet hozni.

AccountMenu
Programindításkor ez az osztály példányosodik. Ebből lehet belépéssel továbblépni az AdminMenube, vagy a CustomerMenube, továbbá itt lehet új admint, vagy vásárlót regisztrálni. Első belépéskor pedig meg lehet adni azt a pinkódot, amit admin regisztrációnál használni kell. Tisztán virtuális függvények amiket kötelezően örököl, az ExitMenu, itt minden osztály személyreszabhatja, hogy mit hozzon létre a nexmenu pointerre. A getType függvény, valamint van egy nem tisztán virtuális függvény is, ez a MenuExtraDisplay(), ide extra tartalmakat lehet betenni ha, nem elég a szabványos megjelenítés.

AdminMenu

Adminként belépve ebbe az osztályba térünk be. A sok ősosztály függvény mellett, elérhetőek lesznek az adminra jellemző feladatok. Ezek a függvények meghívják az admin megfelelő folyamatát, valamint, meghatározzák a feladathoz kellő megjelenítést, és irányítást. Az AdminMenuben egy kompozíció található, az admin pointerre létrejön dinamikusan még az AccountMenuben belépés után egy admin, ami addig aktív amíg az AdminMenuben tevékenykedünk.

CustomerMenu

A CustomerMenu pontosan ugyanazt tudja mint az AdminMenu, csak a feladatai fel vannak cserélve a customerre jellemző feladatokkal. Szintén kompozícióval tartjuk számon a customer pointeren az adott vásárlót.

Egyéb leszármazottak a MenuContainerből
Vannak bonyolultabb menüpontok, ahol meg kell nyitni egy további almenüt, ami ugyanúgy működik, mint a többi menü, csak más függvények csatlakoznak hozzá. Az ilyen almenük is külön osztályt kaptak, ősosztályuk a MenuContainer.

MenuItem és leszármazottai
Ez az absztrakt osztály tartalmazza a menüpontok jellemzőit. A menüpont nevét, indexét, egy castnamet, valamint tisztán virtuális függvényként a Taskot. Ennek az osztálynak a leszármazottjai úgy működnek, hogy az adott leszármazott a Task függvényében elvégez egy ellenőrzött downcastinget, erre szolgál a castname, amit a konstruktorban adunk meg neki, ezt hasonlítja össze az adott menü típusával, mindezek után, a leszármazott osztály pointerével meg tudja hívni a kijelölt feladatot, legyen szó admin, customer, vagy account feladatról. Azért választottam ezt a megoldást, mert így ha bővíteni akarjuk a feladatokat, a menü logikájába, kijelzésbe, adatbevitelbe nem kell belenyúlni.

User
A User osztály rendelkezik azokkal az attribútumokkal és tagfüggvényekkel amik mindkét

leszármazott osztályra jellemzőek. Ezenfelül ezt az osztályt is lehet példányosítani, amikor még nem tudjuk milyen felhasználót akarunk létrehozni, például belépésnél. Továbbá van egy virtuális függvénye, ami összegyűjti a user settereit.

Admin
Az Admin osztály az adminra jellemző attribútumokkal, és tagfüggvényekkel egészíti ki az alaposztályt. Az admin tagfüggvények között kapnak helyet az adminra jellemző feladatok is.

Customer
A Customer osztálynak több attribútuma is van, ugyanis regisztrációnál több személyes adatot is meg kell adni. Ezenfelül ugyanúgy működik mint az Admin osztály, csak Customer rangú feladatokkal.

Hardware és leszármazottjai
A Hardware absztrakt osztály tartalmazza minden hardware közös attribútumait. Ezenfelül van egy tisztán virtuális függvénye, ami összegyűjti a hardware settereit.

BaseCentralStorage
A BaseCentralStorage kezeli az adatokat. Itt tárolódnak az Adminok, Customerek, Hardwarek, Orderek heterogén kollekciókban. Ezt a központi tárolót a program elején hozzuk létre, és a futás végén szabadítjuk fel. Ez az osztály a program elején ellenőrzi, hogy minden fájl megvan e, ha valami nincs meg akkor azt létrehozza. Ezután kiolvassa belőle az adatokat. Ha érkezik egy parancs a belépett felhasználó osztályának, akkor az az osztály innen kérdezi le az adatokat, amikkel dolgozni szeretne. Az osztályhoz tartozik három kompozíció, Adminra, Customerre, és Hardwerre, mivel ilyen osztályokat tárol el, és mikor megszűnik a program végén ez az osztály, a tárolt adatokra sincs már szükség, persze először visszamenti őket a fájlokba, de minden adatmozgatás után történik egy mentés.

További osztályok

CheckError
Ez az osztály majdnem mindenkihez kompozícióval van kapcsolva, ugyanis ahol adatellenőrzés, adatbevitel, fájlművelet, kivétel van, ott jelen kell lennie ennek az osztálynak, viszont mindenhol helyi működése van, ezért lokálisan van létrehozva, és addig lehet vele az adott osztályban dolgozni, amíg meg nem szűnik az osztály. Ez az osztály egy szabványos hiba, vagy kivétel kimenetet biztosít, vagyis meg van benne határozva egy olyan rész a konzolba, ahova mindig kiírja a hibát, vagy kivételt. Minden futásidőbeli, elkapott hibánál, vagy kivételnél ez az osztály hívódik meg, ami kiírja a megadott szöveget a szabványos helyre. Ez az osztály elég gyakran van használva, például már ha csak a felhasználói érvénytelen adatbevitelre gondolunk.

FileHandler
A központi tároló ezt az osztályt használja az adatok mentésére, és betöltésére. Kompozícióval van hozzákapcsolva a BaseCentralStoragehez.

HardwareStorage
Mivel hardverekből sok külön osztály van, ezért a központi tárolóba, egy másik osztályt

helyezünk be ahelyett, hogy minden hardvernak létrehozzunk ott egy tárolót. Tárolói lesznek minden hardvernek, de ezek el lesznek különítve ebbe a HardwareStorageba.

OrdersPerUser
Az OrdersPerUser osztály egy kompozíciója a BaseCentralStoragenak. Egy ilyen osztály egy vásárló rendeléseit tudja tartalmazni, ezért ilyen osztályból egy heterogén kollekció található a BaseCentralStorageban. Ez az osztály tartalmazza a vásárló felhasználónevét, illetve egy Orders osztály heterogén kollekciót. A BaseCentralStorageban, akkor adódik új elem az ebből az osztályból álló tömbhöz, ha egy olyan vásárló rendel, akinek nem volt bejegyezve egy rendelése sem, ha megszűnik egy vásárlónak minden rendelése, törlődik az adott elem.

Order
Az Order osztály egy egyedi azonosítót tartalmaz, amivel megkülönböztethetjük az adott felhasználó többi rendelése között. Ezenfelül tartalmaz még egy enumot, ami a rendelés státuszát jelöli, valamint egy HardwareStorage osztályt, ugyanis egy rendelés több fajta hardvert is tartalmazhat. Új rendelés esetén egy ilyen elem adódik hozzá a OrdersPerUser osztály rendeléseket tartalmazó tömbjéhez.

Setters és leszármazottjai
A Setters absztrakt osztály és leszármazottjai abban segítenek hogy a felhasználói inputot, és a fájlbeolvasást generalizálni lehessen. Ezek a setter osztályok mindig rendelkeznek egy mutatóval arra az objektumra aminek be akarjuk állítani valamelyik változóját. Ezenfelül rendelkeznek egy virtuális Set függvénnyel ami meghívja annak az objektumnak az adott setterét, ami be lett rajta állítva.

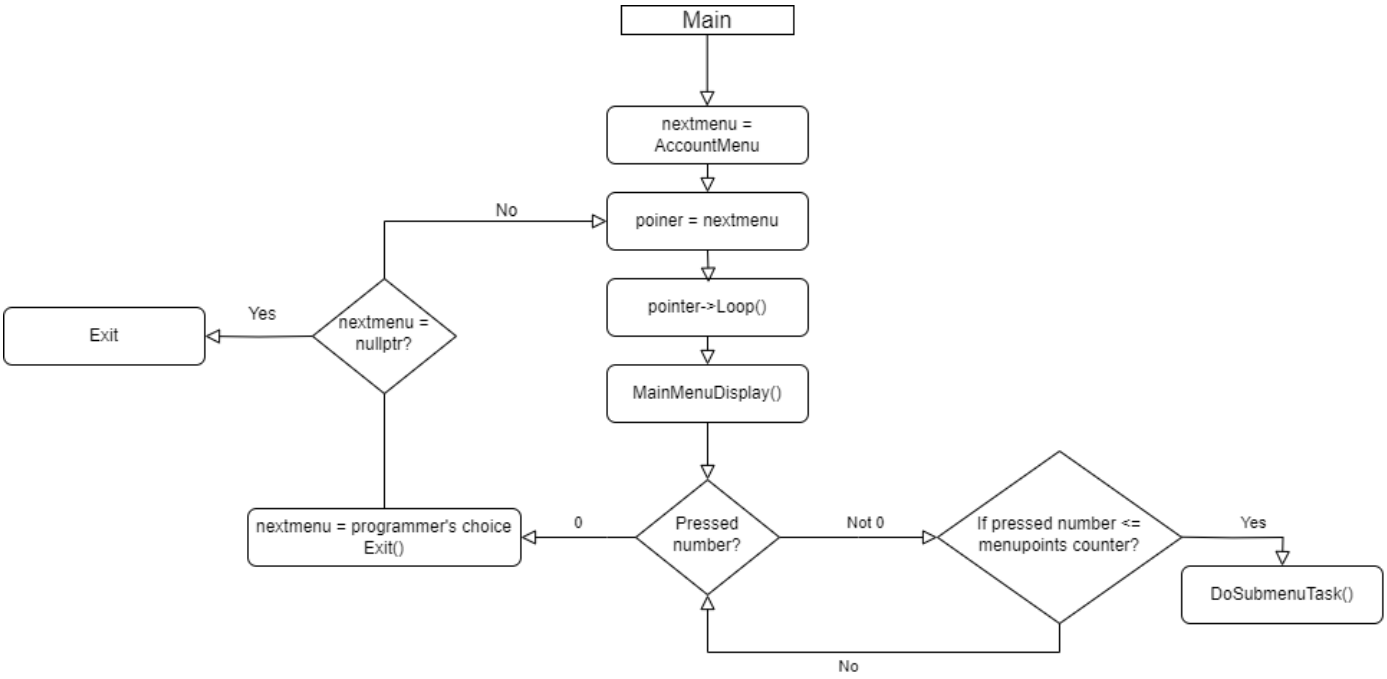
Array<T> és Array<T> template osztályok**
Ezek az osztályok egydimenziós, illetve kétdimenziós tömbként, heterogén kollekcióként szolgálnak. Tudunk sorozatosan, illetve egyesével hozzáadni tagokat, törölni is tudunk, valamint ha tömb adatot akarunk berakni mint amekkorát foglaltunk akkor újraméretezi magát. Ezenfelül nem kell a felszabadításra figyelni, ugyanis a destruktorban ez megtörténik.

InvalidFormatException
Ez a kivételosztály az std::runtime_error osztályból öröklődik. Ilyen kivételt dob a program akkor, ha futási időben, a felhasználó, nem a kért adatnak megfelelő formátumú szöveget ír be. A catch ágakból általában a CheckError egyik hibakazelő függvénye hívódik meg, ami kiírja a hibát.

InvalidProcess
Ez a kivételosztály az std::logic_error osztályból öröklődik. Ilyen kivételt dob a program akkor, ha valamilyen futási időben kiderülő probléma van, ami megszakítja a program működését, ilyen egy rossz castolás, vagy egy sikertelen fájl megnyitás. Ahol számítani lehet ilyen hibára, ott a catch ág a CheckError egyik hibakezelő függvényét hívja meg, ami kiírja a hibát.

Fontos algoritmusok, függvények

Alább a main függvény logikáját mutatom be, vagyis azt, hogy miként lehet a menük között közlekedni, és mi történik a menüben.



A main függvényben először létrehozzuk dinamikusan az AccountMenut. Elindítjuk a Loop függvényt, amiben addig maradunk bent, amig menüváltás nem történik. Alapértelmezetten először a MainMenu függvénybe megyünk be, ami megjeleníti a szabványos menüt, majd inputra vár. Ezután ha 0 volt az input akkor kilépünk a menüből, de előtte a nextmenure lefoglaljuk a következő menüt, vagy nullptr állítva kilépünk a programból. Ha nem 0-át nyomunk, akkor a program megnézi, hogy a nyomott számhoz, 9 menüpont után betűhöz, van e rendelve valamilyen feladat, itt jönnek be a képpbe majd a MenuItem leszármazottak. Ha van adott számhoz, vagy betűhöz tartozó menüpont, akkor végrehajtja a hozzá rendelt feladatot, ha nincs akkor továbbra is inputra vár. Ebből az algoritmusból az olyan lépések, mint az előző menü felszabadítása, vagy kilépéskor való felszabadítás kimaradtak, de természetesen implementálva vannak.

Menu.h, Menu.cpp

```
void MenuContainer::Loop()
```

Ez a függvény adja egy menün belül a logikát. Ha a főmenü részbe vagyunk, akkor egy szabványos logikát követ, ha elindítjuk a menünek az egyik feladatát, akkor az irányítást átadja annak a függvénynek.

```
void MenuContainer::MainMenu()
```

Ez a függvény az a szabványos logika, ami a loop főmenü részébe van. Először kiírja a menü nevét, majd az opciókat, majd választ vár a felhasználótól.

```
void MenuContainer::MenuItemPrint(const int count, std::string* menu
points, const bool displaysubmenu) const
```

Ez a függvény írja ki megformálva a menü opciókat. A count, a menüpontok számát, a menupoints pointer magát a menüpontokat tartalmazza. A displaysubmenu változóval eldönthető, hogy a menü neve a többivel együtt íródjon e ki, vagy külön akarunk vele foglalkozni.

```
void MenuContainer::Table(std::string** content, const int rows, con
st int columns, const int columswidth, const int startheight, const
int selected) const
```

Ez a függvény adja a táblázatos formátum alapját. A content tömb értéket jeleníti meg rows sorban, columns oszlopban, a columswidth jelöli az oszlopok szélességét, a startheight jelöli a kirajzolás kezdeti magasságát, a selected pedig azt jelöli, hogy melyik sor van kiválasztva.

```
template<class Tclass>

void TextInput(Tclass& selectedclass, const int textsnumber, std::st
ring* menupoints, const int offset = 0, const bool resetpos = false)
```

Ez a sablon függvény segítségével lehet beolvasni bevitelimezőbe, és beírni a beolvasott adatot egy osztály attribútumába. A selectedclass referencia az az osztály kell legyen, amelynek attribútumaiba be akarunk olvasni. Ehhez minden ilyen osztálynak kell legyen egy segédosztálya, ami meghatározza, hogy mely attribútumok settereit hívja meg a TextTinput. A textsnumber a beolvasandó adatok számát jelöli, a menupoints a beolvasandó adatok megnevezését, az offset, a függőleges kirajzolás kezdetét, a resetpos segítségével pedig eldönthető, hogy a kurzor visszaálljon e a kezdő állásba, vagy maradjon a kurzor ott, ahol a függvény hagyta.

```
virtual void MenuExtraDisplay() {}
```

Ez a függvény virtuális, és lehetőséget kínál bármilyen speciális menünek arra, hogy a szabvány megjelenítés mellett, extra megjelenítési logikát alkalmazzon.

```
void AccountMenu::RegistrationProcess()
```

A további függvények ehhez hasonlóak, az egyes menük ilyen függvényeken keresztül valósítják meg a menüopciók működését.

MenuItem.h, MenuItem.cpp

```
virtual void Task(MenuContainer* container) = 0;
```

Minden MenuItem leszármazott osztály, ilyen függvényben hívja meg az átadott MenuContainer leszármazott osztály megfelelő menüopció végrehajtását. Mivel alaposztály pointert adunk át, ezért ezekben a függvényekben ellenőrzött downcasting történik.

User.h, User.cpp

```
bool User::operator==(const User& user) const
```

Az összehasonlító operátor dönti el, hogy két felhasználó megegyezik e. Mivel minden felhasználó egyedi felhasználónévvel rendelkezik, ezért a felhasználóneveket hasonlítja össze a függvény.

```
bool User::isStringEmpty(const std::string& text)
```

Ez a függvény minden beviteli mezőnél ellenőríz, azt ellenőrzi, hogy a felhasználó írt e valamit a beviteli mezőbe.

```
bool User::isContainSplit(const std::string& text)
```

Ez a függvény minden beviteli mezőnél ellenőríz, a beviteli mezőbe nem kerülhet | karakter, mert azt használja a fájlkezelés az adatok elválasztásához.

```
virtual void FormatInfo(std::string* text) const;
```

Ez a függvény virtuális, ezért minden felhasználó osztálynál személyre szabható. A függvény a beadott tömbbe visszaadja az osztály attribútumait.

```
virtual void GetMenuPoints(Array<std::string>* strings, std::string*
 otherconnection = nullptr) const;
```

Ez a függvény virtuális, a felhasználó osztályoknak a dolga, hogy meghatározzák, hogy milyen attribútumaikba akarnak olvasni a felhasználótól, ezekhez az attribútumokhoz kell biztosítani megnevezéseket, amelyeket az Array string pointer specializációba, vagy sima string tömbbe tud betölteni a függvény.

```
Array<Setters**>& GetSetters();
```

Ez a függvény Array setter pointer pointer specializációba adja vissza a Setter osztály leszármazottosztályait, amik meghívják az adott felhasználó osztályhoz tartozó attribútumok settereit.

```
void AddUser();
```

A felhasználó osztályok, vagyis a User, Admin, Customer osztály ehhez hasonló függvényekben határozzák meg, hogy az adott osztály milyen műveletek tud csinálni, mire jogosultak.

CentralStorage.h, CentralStorage.cpp

```
template<class Tclass, class StorageClass>

void ReadFile(StorageClass& storageclass, const std::string filename
)
```

Sablon függvény, amely a programra jellemző formátumú txt fájlokat olvassa be, egy storageclass osztályba. A speciális formátummal rendelkező txt fájlokat külön függvénnyel olvassuk.

```
void ReadHardwareFile(std::string filename);

void ReadOrderFile(std::string filename);
```

Ez a két függvény a speciális formátúmmal rendelkező hardware.txt, és orders.txt fájlokból olvas be.

```
template<class Tclass, class StorageClass>

void SaveFile(StorageClass& storageclass, const std::string filename
)
```

Ez a sablon függvény visszamenti az adott osztály adatait a megadott fájlba. A jellemző formátúmú txt fájlokra használjuk.

```
template<class Tclass, class StorageClass>

void SaveOneToFile(StorageClass& storageclass, const std::string fil
ename)
```

Ez a sablonfüggvény hozzáfűzi a storageclass adatait a megadott fájlhoz.

```
const Array<Admin**>& GetAdminArray() const;
```

Ilyen függvényekkel lehet a Központi Tárolóból lekérni az egyes osztályok tárolóit.

```
static BaseCentralStorage* GetCentralStorage();
```

Ez a statikus függvény visszaadja a dinamikusan foglalt Központi Tároló saját pointerét.

hardware.h, hardware.cpp

```
virtual Hardware* Clone() const = 0;
```

Ez a virtuális függvény minden hardware leszármazottnál dinamikusan létrehoz egy másolatot, és visszaadja a hardver pointerét.

```
bool Hardware::operator==(const Hardware& hardware)
```

Ez a függvény két hardware osztályból származó osztályt hasonlít össze, id alapján különböztetjük meg a hardvereket.

```
CPU& CPU::operator=(const CPU& cpu)
```

Minden hardware leszármazottnak van egy értékadó operátora.

order.h, order.cpp

```
void Order::OrderDecode(std::string& decoderow, size_t** counts, siz
e_t** ids, size_t& itemnumber)
```

Ez a függvény dekódolja a txt fájlban lévő rendeléseket formátumát.

```
void Order::OrderEncode(std::string& encodedmessage, std::string del
imeter)
```

Ez a függvény visszaalakítja a rendeléseket a txt fájlban található formátumra.

A program futtatása fejlesztő környezetben

mainek
A programnak két main függvénye van. Az egyik a test.cpp fájlban van, a teszteseteket futtatja le, és csak szabvány könyvtárral dolgozik. Az előbbi miatt, nincs képernyő törlés, bemenet olvasás enter nélkül, konzol színezés. A másik, a main.cpp fájlban van, a normális program logikát követi, alapesetben ezt érdemes használni, ugyanis itt minden fentebb írt funkció működik, viszont ehhez nem szabványkönyvtárakat is használ.

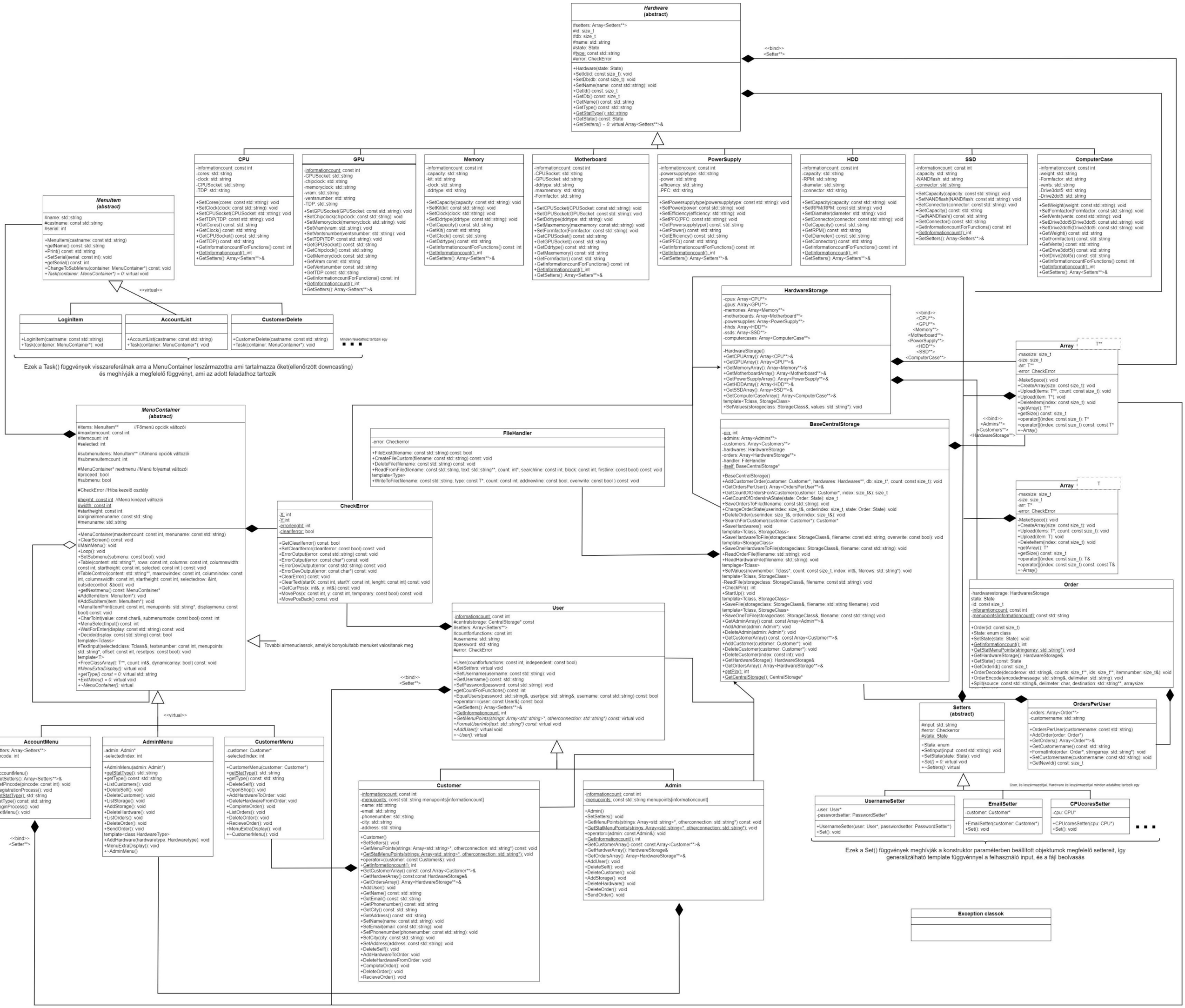
Szabványkönyvtáron kívüli könyvtárat
<conioh.h>
<Windows.h>
”memtrace.h”
”gtest_lite.h”

defineok
A két fajta futtatás között a CportaDefine.h fájlban található CPORTA makró tesz

különbséget. Ha nincs definiálva, akkor a normális futás érvényesül, ha definiálva van, akkor a tesztesetek futnak.

A teljes diagram egy oldallal lejjebb található.

A felhasználói dokumentáció, és a specifikáció is be lett másolva, a diagram után található


Hardware
(abstract)
#setters: Array<Setters**>
#id: size_t
#db: size_t
#name: std::string
#state: State
#type: const std::string
#error: CheckError
+Hardware(state: State)
+SetId(id: const size_t): void
+SetDb(db: const size_t): void
+SetName(name: const std::string): void
+GetId() const: size_t
+GetDb() const: size_t
+GetName() const: std::string
+GetType() const: std::string
+GetStatType(): std::string
+GetState() const: State
+GetSetters() = 0: virtual Array<Setters**>&
<<bind>>
<Setter**>
CPU
-informationcount: const int
-cores: std::string
-clock: std::string
-CPUSocket: std::string
-TDP: std::string
+SetCores(cores: const std::string): void
+SetClock(clock: const std::string): void
+SetCPUSocket(CPUSocket: std::string): void
+SetTDP(TDP: const std::string): void
+GetCores() const: std::string
+GetClock() const: std::string
+GetCPUSocket() const: std::string
+GetTDP() const: std::string
+GetInformationcountForFunctions() const: int
+GetInformationcount(): int
+GetSetters(): Array<Setters**>&
GPU
-informationcount: const int
-GPUSocket: std::string
-chipclock: std::string
-memoryclock: std::string
-vram: std::string
-ventsnumber: std::string
-TDP: std::string
+SetGPUSocket(GPUSocket: const std::string): void
+SetChipclock(chipclock: const std::string): void
+SetMemoryclock(memoryclock: std::string): void
+SetVram(vram: std::string): void
+SetVentsnumber(ventsnumber: std::string): void
+SetTDP(TDP: const std::string): void
+GetGPUSocket() const: std::string
+GetChipclock() const: std::string
+GetMemoryclock const: std::string
+GetVram const: std::string
+GetVentsnumber const: std::string
+GetTDP const: std::string
+GetInformationcountForFunctions() const: int
+GetInformationcount(): int
+GetSetters(): Array<Setters**>&
Memory
-informationcount: const int
-capacity: std::string
-kit: std::string
-clock: std::string
-ddrtype: std::string
+SetCapacity(capacity: const std::string): void
+SetKit(kit: const std::string): void
+SetClock(clock: std::string): void
+SetDdrtype(ddrtype: const std::string): void
+GetCapacity() const: std::string
+GetKit() const: std::string
+GetClock() const: std::string
+GetDdrtype() const: std::string
+GetInformationcountForFunctions() const: int
+GetInformationcount(): int
+GetSetters(): Array<Setters**>&
Motherboard
-informationcount: const int
-CPUSocket: std::string
-GPUSocket: std::string
-ddrtype: std::string
-maxmemory: std::string
-Formfactor: std::string
+SetCPUSocket(CPUSocket: const std::string): void
+SetGPUSocket(GPUSocket: const std::string): void
+SetDdrtype(ddrtype: std::string): void
+SetMaxmemory(maxmemory: const std::string): void
+SetFormfactor(Formfactor: const std::string): void
+GetCPUSocket() const: std::string
+GetGPUSocket() const: std::string
+GetDdrtype() const: std::string
+GetMaxmemory() const: std::string
+GetFormfactor() const: std::string
+GetInformationcountForFunctions() const: int
+GetInformationcount(): int
+GetSetters(): Array<Setters**>&
PowerSupply
-informationcount: const int
-powersupplytype: std::string
-power: std::string
-efficiency: std::string
-PFC: std::string
+SetPowersupplytype(powersupplytype: const std::string): void
+SetPower(power: const std::string): void
+SetEfficiency(efficiency: std::string): void
+SetPFC(PFC: const std::string): void
+GetPowersupplytype() const: std::string
+GetPower() const: std::string
+GetEfficiency() const: std::string
+GetPFC() const: std::string
+GetInformationcountForFunctions() const: int
+GetInformationcount(): int
+GetSetters(): Array<Setters**>&
HDD
-informationcount: const int
-capacity: std::string
-RPM: std::string
-diameter: std::string
-connector: std::string
+SetCapacity(capacity: const std::string): void
+SetRPM(RPM: const std::string): void
+SetDiameter(diameter: std::string): void
+SetConnector(connector: const std::string): void
+GetCapacity() const: std::string
+GetRPM() const: std::string
+GetDiameter() const: std::string
+GetConnector() const: std::string
+GetInformationcountForFunctions() const: int
+GetInformationcount(): int
+GetSetters(): Array<Setters**>&
SSD
-informationcount: const int
-capacity: std::string
-NANDflash: std::string
-connector: std::string
+SetCapacity(capacity: const std::string): void
+SetNANDflash(NANDflash: const std::string): void
+SetConnector(connector: const std::string): void
+GetCapacity() const: std::string
+GetNANDflash() const: std::string
+GetConnector() const: std::string
+GetInformationcountForFunctions() const: int
+GetInformationcount(): int
+GetSetters(): Array<Setters**>&
ComputerCase
-informationcount: const int
-weight: std::string
-Formfactor: std::string
-vents: std::string
-Drive3dot5: std::string
-Drive2dot5: std::string
+SetWeight(weight: const std::string): void
+SetFormfactor(Formfactor: const std::string): void
+SetVents(vents: const std::string): void
+SetDrive3dot5(Drive3dot5: const std::string): void
+SetDrive2dot5(Drive2dot5: const std::string): void
+GetWeight() const: std::string
+GetFormfactor() const: std::string
+GetVents() const: std::string
+GetDrive3dot5() const: std::string
+GetDrive2dot5() const: std::string
+GetInformationcountForFunctions() const: int
+GetInformationcount(): int
+GetSetters(): Array<Setters**>&
MenuItem
(abstract)
#name: std::string
#castname: const std::string
#serial: int
+MenuItem(castname: const std::string)
+getName() const: std::string
+Print() const: std::string
+SetSerial(serial: const int): void
+getSerial() const: int
+ChangeToSubMenu(container: MenuContainer*) const: void
+Task(container: MenuContainer*) = 0: virtual void
<<virtual>>
LoginItem
+LoginItem(castname: const std::string)
+Task(container: MenuContainer*): void
AccountList
+AccountList(castname: const std::string)
+Task(container: MenuContainer*): void
CustomerDelete
+CustomerDelete(castname: const std::string)
+Task(container: MenuContainer*): void
Minden feladathoz tartozik egy
Ezek a Task() függvények visszareferálnak arra a MenuContainer leszármazottra ami tartalmazza őket(ellenőrzött downcasting)
és meghívják a megfelelő függvényt, ami az adott feladathoz tartozik
HardwareStorage
-cpus: Array<CPU**>
-gpus: Array<GPU**>
-memories: Array<Memory**>
-motherboards: Array<Motherboard**>
-powersupplies: Array<PowerSupply**>
-hhds: Array<HDD**>
-ssds: Array<SSD**>
-computercases: Array<ComputerCase**>
-HardwareStorage()
+GetCPUArray(): Array<CPU**>&
+GetGPUArray(): Array<GPU**>&
+GetMemoryArray(): Array<Memory**>&
+GetMotherboardArray(): Array<Motherboard**>&
+GetPowerSupplyArray(): Array<PowerSupply**>&
+GetHDDArray(): Array<HDD**>&
+GetSSDArray(): Array<SSD**>&
+GetComputerCaseArray(): Array<ComputerCase**>&
template<Tclass, StorageClass>
+SetValues(storageclass: StorageClass&, values: std::string*): void
<<bind>>
<CPU**>
<GPU**>
<Memory**>
<Motherboard**>
<PowerSupply**>
<HDD**>
<SSD**>
<ComputerCase**>
Array
T**
-maxsize: size_t
-size: size_t
-arr: T**
-error: CheckError
-MakeSpace(): void
+CreateArray(size: const size_t): void
+Upload(items: T**, count: const size_t): void
+Upload(item: T*): void
+DeleteItem(index: const size_t): void
+getArray(): T**
+getSize() const: size_t
+operator[](index: const size_t): T*
+~Array()
<<bind>>
<Admins**>
<Customers**>
<HardwareStorage**>
BaseCentralStorage
-pin: int
-admins: Array<Admins**>
-customers: Array<Customers**>
-hardwares: HardwareStorage
-orders: Array<HardwareStorage**>
-handler: FileHandler
-itself: BaseCentralStorage*
+BaseCentralStorage()
+AddCustomerOrder(customer: Customer*, hardwares: Hardwares**, db: size_t*, count: const size_t): void
+GetOrdersPerUser(): Array<OrdersPerUser**>&
+GetCountOfOrdersForACustomer(customer: Customer*, index: size_t&): size_t
+GetCountOfOrdersInAState(state: Order::State): size_t
+SaveOrdersToFile(filename: const std::string): void
+ChangeOrderState(userindex: size_t&, orderindex: size_t, state: Order::State): void
+DeleteOrder(userindex: size_t&, orderindex: size_t&): void
+SearchForCustomer(customer: Customer*): Customer*
+SaveHardwares(): void
template<Tclass, StorageClass>
+SaveHardwareToFile(storageclass: StorageClass&, filename: const std::string, overwrite: const bool): void
template<StorageClass>
+SaveOneHardwareToFile(storageclass: StorageClass&, filename: const std::string): void
+ReadOrderFile(filename: std::string): void
+ReadHardwareFile(filename: std::string): void
temnplage<Tclass>
+SetValues(newmember: Tclass*, count: const size_t, index: int&, filerows: std::string*): void
template<Tclass, StorageClass>
-ReadFile(storageclass: StorageClass&, filename: const std::string): void
+CheckPin(): int
+StartUp(): void
template<Tclass, StorageClass>
+SaveFile(storageclass: StorageClass&, filename: std::string filename): void
template<Tclass, StorageClass>
+SaveOneToFile(storageclass: StorageClass&, filename: const std::string) void
+GetAdminArray() const: const Array<Admin**>&
+AddAdmin(admin: Admin*): void
+DeleteAdmin(admin: Admin*): void
+GetCustomerArray() const: const Array<Customer**>&
+AddCustomer(customer: Customer*): void
+DeleteCustomer(customer: Customer*): void
+DeleteCustomer(index: const int): void
+GetHardwareStorage(): HardwareStorage&
+GetOrdersArray(): Array<HardwareStorage**>&
+getPin(): int
+GetCentralStorage(): CentralStorage*
Array
T
-maxsize: size_t
-size: size_t
-arr: T*
-error: CheckError
-MakeSpace(): void
+CreateArray(size: const size_t): void
+Upload(items: T*, count: const size_t): void
+Upload(item: T): void
+DeleteItem(index: const size_t): void
+getArray(): T*
+getSize() const: size_t
+operator[](index: const size_t): T&
+operator[](index: const size_t) const: const T&
+~Array()
Order
-hardwarestorage: HardwaresStorage
state: State
-id: const size_t
-inforamtioncount: const int
-menupoints[informationcount]: const std::string
+Order(id: const size_t)
+State: enum class
+SetState(state: State): void
+GetInformationcount(): int
+GetStatMenuPoints(stringarray: std::string*): void
+GetHardwareStorage(): HardwareStorage&
+GetState() const: State
+GetOrderId() const: size_t
+OrderDecode(decoderow: std::string&, counts: size_t**, ids: size_t**, itemnumber: size_t&): void
+OrderEncode(encodedmessage: std::string&, delimeter: std::string): void
+Split(source: const std::string&, delimeter: char, destination: std::string**, arraysize:
Setters
(abstract)
#input: std::string
#error: Checkerror
#state: State
+State: enum
+SetInput(input: const std::string): void
+SetState(state: State): void
+Set() = 0: virtual void
+~Setters(): virtual
OrdersPerUser
-orders: Array<Order**>
-customername: std::string
+OrdersPerUser(customername: const std::string)
+AddOrder(order: Order*)
+GetOrders(): Array<Order**>&
+GetCustomername() const: std::string
+FormatInfo(order: Order*, stringarray: std::string*): void
+SetCustomername(customername: const std::string): void
+GetNewId() const: size_t
User, és leszármazottjai, Hardware és leszármazottjai minden adatához tartozik egy
UsernameSetter
-user: User*
-passwordsetter: PasswordSetter*
+UsernameSetter(user: User*, passwordsetter: PasswordSetter*)
+Set(): void
EmailSetter
-customer: Customer*
+EmailSetter(customer: Customer*)
+Set(): void
CPUcoresSetter
-cpu: CPU*
+CPUcoresSetter(cpu: CPU*)
+Set(): void
Ezek a Set() függvények meghívják a konstruktor paraméterben beállított objektumok megfelelő settereit, így
generalizálható template függvénnyel a felhasználó input, és a fájl beolvasás
FileHandler
-error: Checkerror
+FileExist(filename: const std::string) const: bool
+CreateFileCustom(filename: const std::string) const: void
+DeleteFile(filename: const std::string) const: void
+ReadFromFile(filename: const std::string, text: std::string**, count: int*, searchline: const int, block: const int, firstline: const bool) const: void
template<Type>
+WriteToFile(filename: const std::string, type: const T*, count: const int, addnewline: const bool, overwrite: const bool) const: void
CheckError
-X: int
-Y:int
-errorlenght: int
-clearferror: bool
+GetClearIferror() const: bool
+SetClearIferror(clearferror: const bool) const: void
+ErrorOutput(error: const std::string) const: void
+ErrorOutput(error: const char*) const: void
+ErrorDevOutput(error: const std::string) const: void
+ErrorDevOutput(error: const char*) const: void
+ClearError() const: void
+ClearText(startX: const int, startY: const int, lenght: const int) const: void
+GetCurPos(x: int&, y: int&) const: void
+MovePos(x: const int, y: const int, temporary: const bool) const: void
+MovePosBack() const: void
MenuContainer
(abstract)
#items: MenuItem** //Főmenü opciók változói
#maxitemcount: const int
#itemcount: int
#selected: int
#submenuitems: MenuItem** //Almenü opciók változói
#submenuitemcount: int
#MenuContainer* nextmenu //Menü folyamat változói
#proceed: bool
#submenu: bool
#CheckError //Hiba kezelő osztály
#height: const int //Menü kinézet változói
#width: const int
#startheight: const int
#originalmenuname: const std::sting
#menuname: std::string
+MenuContainer(maxitemcount: const int, menuname: const std::string)
+ClearScreen() const: void
#MainMenu(): void
+Loop(): void
+SetSubmenu(submenu: const bool): void
+Table(content: std::string**, rows: const int, columns: const int, columnswidth: const int, startheight: const int, selected: const int) const: void
#TableControl(content: std::string**, maxrowindex: const int, columnindex: const int, columnswidth: const int, startheight: const int, selectedrow: &int, outsidecontrol: &bool): void
+getNextmenu() const: MenuContainer*
#AddItem(item: MenuItem*): void
#AddSubItem(item: MenuItem*): void
+MenuItemPrint(count: const int, menupoints: std::string*, displaymenu: const bool) const: void
+CharToInt(value: const char&, submenumode: const bool) const: int
+MenuSelectInput() const: int
+WaitForEnter(display: const std::string) const: void
+Decide(display: const std::string) const: bool
template<Tclass>
#TextInput(selectedclass: Tclass&, textsnumber: const int, menupoints: std::string*, offset: const int, resetpos: const bool): void
template<T>
+FreeClassArray(t: T**, count: int&, dynamicarray: bool) const: void
#MenuExtraDisplay(): virtual void
+getType() const = 0: virtual std::string
+ExitMenu() = 0: virtual void
+~MenuContainer(): virtual
Tovabbi almenuclassok, amelyik bonyolultabb menuket valositanak meg
<<virtual>>
AccountMenu
-setters: Array<Setters**>
-pincode: int
+AccountMenu()
+GetSetters(): Array<Setters**>&
+SetPincode(pincode: const int): void
+RegistrationProcess(): void
+getStatType(): std::string
+getType() const: std::string
+LoginProcess(): void
+ExitMenu(): void
AdminMenu
-admin: Admin*
-selectedIndex: int
+AdminMenu(admin: Admin*)
+getStatType(): std::string
+getType() const: std::string
+ListCustomers(): void
+DeleteSelf(): void
+DeleteCustomer(): void
+ListStorage(): void
+AddStorage(): void
+DeleteHardware(): void
+ListOrders(): void
+DeleteOrder(): void
+SendOrder(): void
template<class HardwareType>
+AddHardware(hardwaretype: Hardwaretype): void
+MenuExtraDisplay(): void
+~AdminMenu()
CustomerMenu
-customer: Customer*
-selectedIndex: int
+CustomerMenu(customer: Customer*)
+getStatType(): std::string
+getType() const: std::string
+DeleteSelf(): void
+OpenShop(): void
+AddHardwareToOrder: void
+DeleteHardwareFromOrder: void
+CompleteOrder(): void
+ListOrders(): void
+DeleteOrder(): void
+RecieveOrder(): void
+MenuExtraDisplay(): void
+~CustomerMenu(): void
<<bind>>
<Setter**>
<<bind>>
<Setter**>
User
-informationcount: const int
#centralstorage: CentralStorage* const
#setters: Array<Setters**>
#countforfunctions: const int
#username: std::string
#password: std::string
#error: CheckError
+User(countforfunctions: const int, independent: const bool)
#SetSetters: virtual void
+SetUsername(username: const std::string): void
+GetUsername() const: std::string
+SetPassword(password: const std::string): void
+getCountForFunctions() const: int
+EqualUsers(password: std::string&, usertype: std::string&, username: const std::string) const: bool
+operator==(user: const User&) const: bool
+GetSetters(): Array<Setters**>&
+GetInformationcount: int
+GetMenuPoints(strings: Array<std::string>*, otherconnection: std::string*) const: virtual void
+FormatUserInfo(text: std::string*) const: virtual void
+AddUser(): virtual void
+~User(): virtual
Customer
-informationcount: const int
-menupoints: const std::string menupoints[informationcount]
-name: std::string
-email: std::string
-phonenumber: std::string
-city: std::string
-address: std::string
+Customer()
+SetSetters(): void
+GetMenuPoints(strings: Array<std::string>*, otherconnection: std::string*) const: void
+GetStatMenuPoints(strings: Array<std::string>*, otherconnection: std::string*): void
+operator=(customer: const Customer&): void
+GetInformationcount(): int
+GetCustomerArray() const: const Array<Customer**>&
+GetHardverArray(): const:const HardwareStorage&
+GetOrdersArray(): Array<HardwareStorage**>&
+AddUser(): void
+GetName() const: std::string
+GetEmail() const: std::string
+GetPhonenumber() const: std::string
+GetCity() const: std::string
+GetAddress() const: std::string
+SetName(name: const std::string): void
+SetEmail(email: const std::string): void
+SetPhonenumber(phonenumber: const std::string): void
+SetCity(city: const std::string): void
+SetAddress(address: const std::string): void
+DeleteSelf(): void
+AddHardwareToOrder: void
+DeleteHardwareFromOrder: void
+CompleteOrder(): void
+DeleteOrder(): void
+RecieveOrder(): void
Admin
-informationcount: const int
-menupoints: const std::string menupoints[informationcount]
+Admin()
+SetSetters(): void
+GetMenuPoints(strings: Array<std::string>*, otherconnection: std::string*) const: void
+GetStatMenuPoints(strings: Array<std::string>*, otherconnection: std::string*): void
+operator=(admin: const Admin&): void
+GetInformationcount(): int
+GetCustomerArray() const: const Array<Customer**>&
+GetHardverArray(): HardwareStorage&
+GetOrdersArray(): Array<HardwareStorage**>&
+AddUser(): void
+DeleteSelf(): void
+DeleteCustomer(): void
+AddStorage(): void
+DeleteHardware(): void
+DeleteOrder(): void
+SendOrder(): void
Exception classok

Felhasználói dokumentáció

Webshop és futárszolgálat

Futtatás

A Debug mappában található Futárszolgálat.exe fájlal lehet elindítani a programot. A program teljes képernyős módban indul el.

Irányítás

Menüpontok

A menüpontok 1…9 ig számokkal, utána betűkkel vannak jelölve. A 0. menüpont mindig a menüből, almenüből való kilépést jelöli. A felhasználó, a menüpontok előtti szám, vagy betű lenyomásával tudja érvényesíteni választását. Általában ilyenkor egy almenü nyílik meg.

Beviteli mezők

A beviteli mezőkbe a Kilepes szót beírva visszaléphetünk az előző menübe. Ha nem megfelelő formátumban adjuk meg az adatot, akkor azt a program nem fogadja el, és kiírja, hogy miért nem fogadta el a bemenetet.

Táblázat

A táblázatban a fel, és le nyilakkal tudunk lépkedni. A táblázat felett írja ki a program, hogy milyen opciók közül lehet választani, és a menüpontoknál megismerten lehet kiválasztani az adott opciót.

Döntés és nyugtázás

Beviteli mezők kitöltése után nyugtázni kell az enterrel, hogy sikeresen kitöltöttük a mezőket. Ha olyan műveleteket végzünk, amiket a legvégén kell érvényesíteni, akkor a nyugtázás előtt, egy döntést kapunk a programtól, ahol választhatunk, hogy valóban érvényesítjük a változtatásokat, vagy nem.

Fájlok

A program 5 fájlal dolgozik összesen, ezek a Futárszolgálat mappában találhatóak meg.

Nagy házi specifikáció

Webshop és futárszolgálat

A program célja

A **nyilvántartás jellegű** program egy **hardver webshopot és annak futárszolgálatát** fogja modellezni. A felhasználók **vásárlóként**, és **adminként** is regisztrálhatnak a programban, persze admin szintű regisztráláshoz, először egy előre meghatározott **4 jegyű pin kódot** fog kelleni megadni, ami biztosítja, hogy csak is az arra jogosultak rendelkezzenek admin fiókkal. Belépés után a vásárlói, és admin **privilégiumok** teljesen el lesznek különítve. Az admin felhasználó fog tudni rendelést **törölni**, illetve **kézbesíteni**, valamint lehetősége lesz **feltölteni** a raktárat, és **új hardvert** is hozzáadni a kínálathoz. A vásárló értelemszerűen **rendelhet** a webshopról a készlet erejéig, **lekérdezheti** az eddigi, valamint a **függőben** lévő rendeléseit, **törölhet** rendelést, valamint **átveheti** a kézbesített rendeléseket.

Fájlkezelés

A program **öt** fájlal fog dolgozni. Az egyik txt fájl a 4 jegyű pin kódot fogja tartalmazni az **admin regisztrációhoz**, ez az **első programindításnál** lesz bekérve a felhasználótól. Ezenfelül lesz **négy txt** fájl. Ebből kettő a vásárlók, és az adminok adatait fogja tárolni. Lesz egy fájl ami a vásárlók rendeléseit fogja tárolni felhasználónként. Az utolsó fájl a raktárat fogja kitenni, vagyis milyen hardverből mennyi van.

A felhasználók adatai, **felhasználónként** lesznek csoportosítva, a felhasználók **egy üres sorral** lesznek egymástól elválasztva. A vásárlókhoz több adat fog tartozni, az adminhoz csak a **kötelező felhasználónév és jelszó**.

```
User1 //felhasználónév
userpwd1234 //jelszó
Felhasználó János //név
user@gmail.com  //email
+36201231231 //telefonszám
Budapest //város
User utca 5 //cím

//következő felhasználó
```

A rendelések **felhasználónevenként** lesznek csoportba rendezve, a csoportok **üres sorral lesznek** elkülönítve. Ezekben a csoportokban soronként lesznek a rendelések, egy **–** jelü sor fogja jelezni, hogy az után már a **kézbesített** rendelések adatai vannak. A könnyebb olvashatóság miatt a rendeléseknél a **darabszám**, és a hardver **azonosítója** lesz feltüntetve.

```
User1 //felhasználónév
2X#1 //2db kézbesítetlen rendelés a #1 azonosítójú harverből
-
3X#5 //3 db kézbesített rendelés a #5 azonosítójú harverből
1X#8 //1 db kézbesített rendelés a #8 azonosítójú harverből

//következő felhasználó
```

A hardverek szintén **üres sorokkal** lesznek megkülönböztetve egymástól, első sor az azonosítót fogja tartalmazni, a második a típusát, harmadik a nevét, aztán jönni fognak a hardver adatai, majd végül, a darabszám amennyi raktáron van.

```
#1 //azonosító
videókártya //típus
Nvidia RTX 3060 //név
6GB //vram méret
15000Mhz //vram sebesség
többi adat...
12 //12 db van raktáron

//következő hardver
```

Bemenet

A bemenet a **billentyűzet** lesz. A program alapvető felépítése szerint, lesznek a **főmenük**, és onnan a billentyűzet számaival fogunk tudni lejjebb lépni **almenükbe**. Az almenük kétféle típusba tartozhatnak. Lesznek az **adatbekérő** almenük, például a regisztrációs, a raktár feltöltős, vagy a rendelős almenü, valamint lesz az **adatkiíró** almenü, ami kiírja majd az adott lekérdezést, például az eddigi rendelések almenü.

Kimenet

A két fő kimenet a **konzolablak**, valamint a **fájlok** lesznek.

Hibakezelés

Ha a program nem talál egy fájlt, akkor azt létrehozza. Minden felhasználótól bekért adat **ellenőrzésre** kerül, például felhasználónév nem tartalmazhat space karaktert, vagy nem kerülhet betű oda, ahol számot kér a program. Fájlon belüli formátum ellenőrzés **nem történik**, ugyanis a program feltételezni fogja, hogy adat csak rajta keresztül került bele a fájlba.

Alapértelmezett adatok

A hardverek típusai **fixek** lesznek a programban, vagyis új hardver hozzáadásnál csak a megadott típusokból lehet választani.