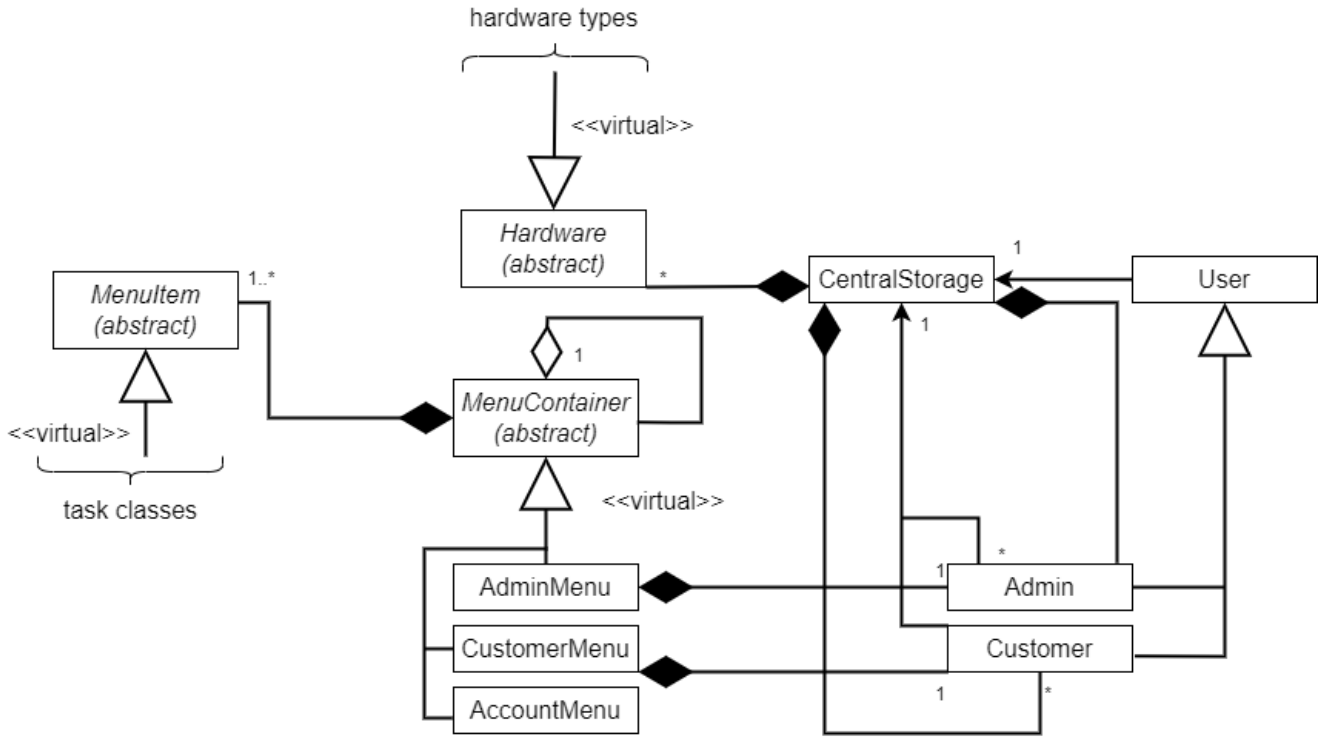
Nagy házi terv

Webshop és futárszolgálat

Általános leírás

A készülőben lévő program számos osztállyal, attribútummal, függvénnyel dolgozik, a teljes osztálydiagram értelmezése időbe telhet, ezért egy egyszerűsített diagramon fogom bemutatni az osztály működését. Ez a diagram tartalmaz minden olyan fontos osztályt, amik képet adnak a program működéséről. A teljes osztálydiagram a terv 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.

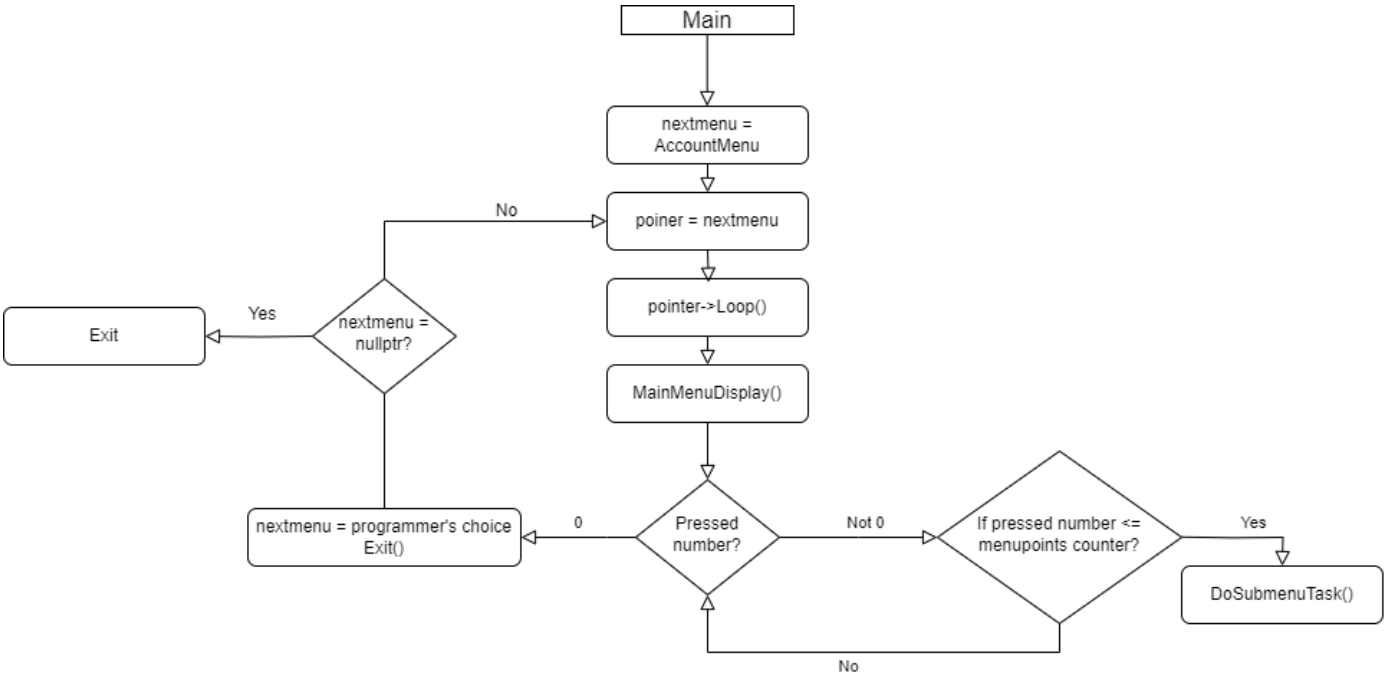
Order
Az Order osztály egy egyedi azonosítót tartalmaz, amivel megkülönböztethetjük az adott felhasználó többi rendelése között. Ezenfelül tartalmaz még egy enumot, ami a rendelés státuszát jelöli, valamint egy HardwareStorage osztályt, ugyanis egy rendelés több fajta hardvert is tartalmazhat. Új rendelés 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.

Fontos algoritmusok

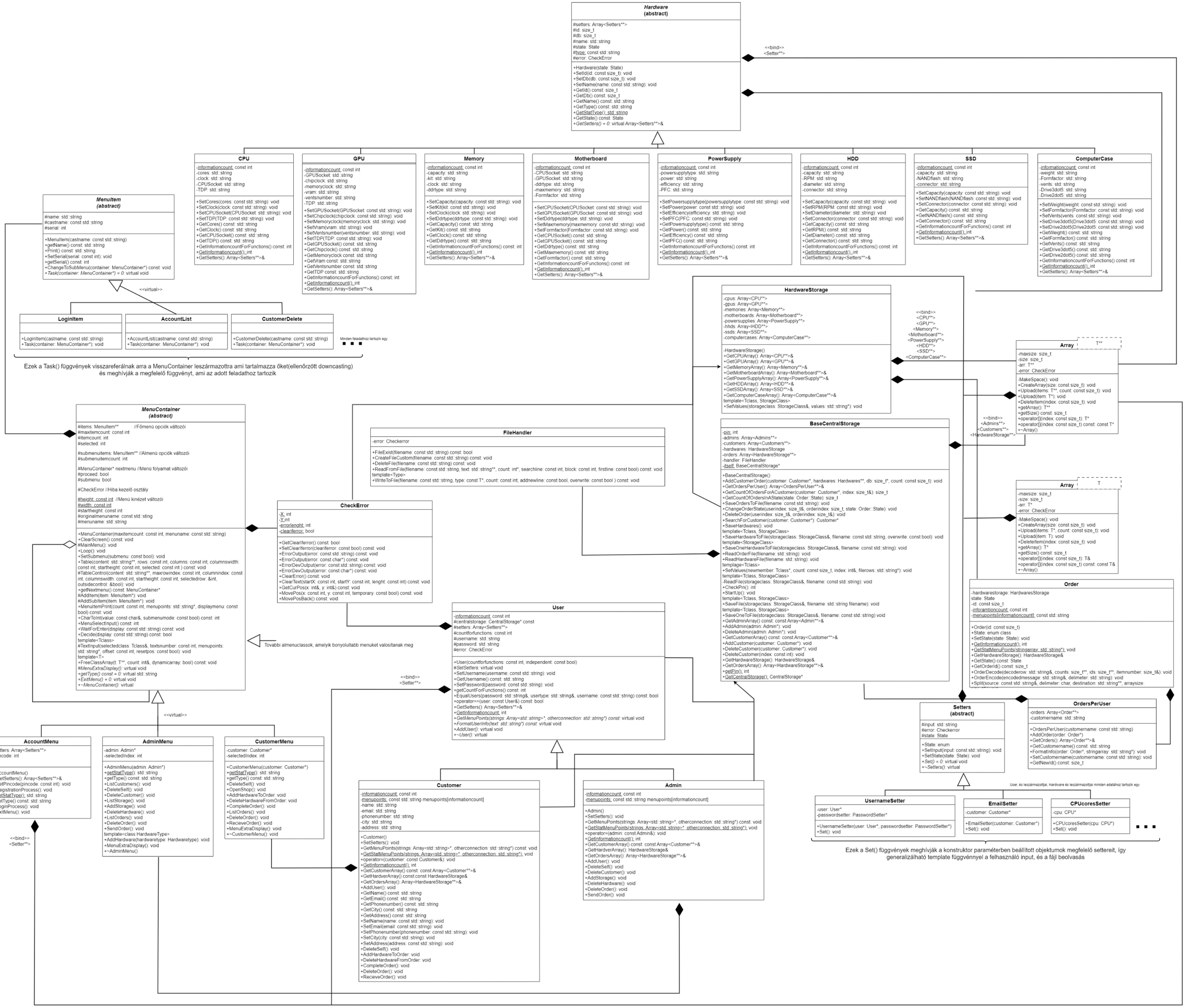
A tervben a legfontosabb algoritmust mutatom be, vagyis azt, hogy miként lehet a menük között közlekedni, és mi történik a menüben.



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, mert csak számot fogad el, van e rendelve valamilyen feladat, itt jönnek be a képpbe majd a MenuItem leszármazottak. Ha van adott számhoz 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 lesznek.

A teljes diagram egy oldallal lejjebb található.

A már elfogadott 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*
+operator[](index: const size_t) const: const T*
+~Array()
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
templage<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*
<<bind>>
<Admins**>
<Customers**>
<HardwareStorage**>
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:
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
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
-clearIferror: bool
+GetClearIferror() const: bool
+SetClearIferror(clearIferror: 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
További almenuclassok, amelyik bonyolultabb menuket valosítanak meg
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
<<bind>>
<Setter**>
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
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
<<virtual>>
<<bind>>
<Setter**>
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