

# A PROGRAMOZÁS ALAPJAI 2.

HÁZI FELADAT DOKUMENTÁCIÓ

## ROLEPLAY GAME

KÉSZÍTETTE: KOMONYI ORBÁN BALÁZS, HWQMMR  
[komonyibalazs@gmail.com](mailto:komonyibalazs@gmail.com)

2025. 05. 16.

# TARTALOMJEGYZÉK

Felhasználói dokumentáció .....	4
Példa bemenet/kimenet:.....	4
Név beírása: .....	4
Karakterválasztás: .....	4
Főmenü pontjai közül választás: .....	5
Osztályok statikus leírása.....	6
Game.....	6
Felelőssége:.....	6
Ősosztályok: .....	6
Attribútumok: .....	6
Metódusok:.....	6
Combat.....	7
Felelőssége:.....	7
Ősosztályok: .....	7
Attribútumok: .....	7
Metódusok:.....	7
Input Handler .....	8
Felelőssége:.....	8
Ősosztályok: .....	8
Attribútumok: .....	8
Metódusok:.....	8
Menu Manager .....	8
Felelőssége:.....	8
Ősosztályok: .....	8
Attribútumok: .....	8
Metódusok:.....	8
Character .....	9
Felelőssége:.....	9
Ősosztályok: .....	9
Attribútumok: .....	9
Metódusok:.....	9
Warrior.....	11
Felelőssége:.....	11
Ősosztályok: .....	11

Attribútumok: .....	11
Metódusok:.....	11
Harcos példányosítása névvel, 1-es szinten, alap pajzsértékkel és alap közelharc fegyverrel.....	11
Wizard .....	12
Felelőssége:.....	12
Ősosztályok: .....	12
Attribútumok: .....	12
Metódusok:.....	12
Archer .....	13
Felelőssége:.....	13
Ősosztályok: .....	13
Attribútumok: .....	13
Metódusok:.....	13
Weapon.....	13
Felelőssége:.....	13
Ősosztályok: .....	13
Attribútumok: .....	13
Metódusok:.....	13
Melee .....	14
Felelőssége:.....	14
Ősosztályok: .....	14
Attribútumok: .....	14
Metódusok:.....	14
Magic .....	15
Felelőssége:.....	15
Ősosztályok: .....	15
Attribútumok: .....	15
Metódusok:.....	15
Ranged .....	16
Felelőssége:.....	16
Ősosztályok: .....	16
Attribútumok: .....	16
Metódusok:.....	16
UML osztálydiagram: .....	17
Összegzés: .....	17
Képernyőképek: .....	18

## Felhasználói dokumentáció

Ez a program egy egyszerű szerepjáték (RPG), amelyben a játékos kiválaszthatja karakterét (harcos, varázsló, íjász), majd különböző menüpontok segítségével küzdelmeket vívhat, képességeket használhat, illetve fegyvereket választhat. A program konzolos alkalmazásként fut, indítás után a név beírása majd a karakterválasztás fogadja a felhasználót.

A felhasználó három fajta bemenetet adhat meg a játék során:

- számot (adott menüpont száma),
- szöveget (nevet a program elején és „yes” / „no” igen-nem kérdésre adható válasz)
- betűt ('y' / 'n' igen-nem kérdésre adható válasz)

### Példa bemenet/kimenet:

Név beírása:

```
Enter your name: Balázs|
```

```
Welcome Balázs!  
  
Choose your character class:  
1. Warrior  
2. Wizard  
3. Archer  
Choice: |
```

Karakterválasztás:

```
Welcome Balázs!  
  
Choose your character class:  
1. Warrior  
2. Wizard  
3. Archer  
Choice: 2|
```

```
You have chosen the Wizard!
```

Főmenü pontjai közül választás:

BASE

1. Wander in the wilderness
2. Reload, repair and regenerate
3. Change weapon
4. Check your character information
5. Quit game

Choice: 4|

INFO

Choose an option:

1. Check character
2. Check weapon
3. Display inventory
4. Back

Choice: |

# Osztályok statikus leírása

## Game

### Felelőssége:

A Game osztály felelős a játék logikájának vezérléséért, kezeli a fő játékmenetet, a menüpontokat és a játékon belüli eseményeket.

### Ősosztályok:

Nincs.

### Attribútumok:

Nincs.

### Metódusok:

#### Public

**static void start():** a játék elindításáért felel, meghívja azokat a függvényeket, amelyek kezelik a név kérést, karakter választást, illetve megjeleníti a főmenüt, ahonnan a többi menübe vezeti a játékost.

**static void handleGameOver(Character\* player):** a Combat osztály hívja meg, kezeli a játék lezárását, újbóli játékot vagy végső kilépést lehet benne választani.

**static void end(Character\* player):** a játékból való teljes kilépést kezeli, ezt a Combat osztály is használja, amennyiben úgy dönt a játékos valamelyik menüben, hogy befejezi a játékot.

**static void displayCharacterInfo(Character& player):** kiírja a játékos-specifikus információkat (életerő, tapasztalat, szint, kiválasztott fegyver, sebzés, karakter osztály), a Combat osztály is használja, ezért publikus.

**static void displayWeaponInfo(const Character& player):** kiírja a játékos kiválasztott fegyverének tulajdonságait (név, sebzés, elhasználtság, mana költség, nyilak száma), a Combat osztály is meghívja.

**static void displayEnemyInfo(Character& enemy):** kiírja az ellenség tulajdonságait a displayCharacterInfo() metódushoz hasonlóan, a fegyverének specifikus tulajdonságaival kiegészítve a displayWeaponInfo() metódushoz hasonlóan.

#### Private

**static std::string getPlayerName():**

A játékos nevének beolvasását kezelő függvény.

**static void chooseCharacter(Character\*& player, const std::string& playerName):**

A játékost néven szólítva kéri meg, hogy válasszon karakter osztályt az ezt kezelő menüvel.

**static Character\* generateRandomEnemy(int playerLevel):**

A játékos szintjének megfelelő ellenfelet hoz létre, amikor a játékos vándorlásba kezd.

**static void watchEnemy(Character& player):**

A függvény a játékos azon választását hajtja végre, ha közelebbről is meg akarja figyelni az ellenséget, kiírja az ellenségről összegyűjtött információkat, majd innen eldöntheti a játékos, hogy harcol az ellenséggel (Combat::start() metódus), vagy megfutamodik, tovább áll (wander metódus), újabb ellenséggel találja szembe magát.

**static void wander(Character& player):**

A függvény a felhasználó azon választását hajtja végre, ha a bázison (start=főmenü) úgy dönt, hogy vándorolni szeretne, ez a függvény hívja meg az ellenség létrehozásáért felelős metódust, majd döntési helyzet elé állítja a játékost, közelebből is megnézi az ellenséget (watchEnemy), a saját tulajdonságait nézné meg, (chooseInformation) vagy pedig visszamegy a bázisra (főmenübe = start).

**static void chooseInformation(Character& player):**

Meghívódik, ha a játékos úgy dönt (start, wander metódusok), hogy a saját tulajdonságait szeretné megnézni mielőtt döntene valamit, egy információs menüt nyit meg, ahol választhat a felhasználó, hogy melyik tulajdonságára kíváncsi (displayCharacterInfo, displayWeaponInfo vagy a karakter inventoryja, összes fegyvere).

**static void managePlayerRepair(Character\*& player):**

A játékos életerejét, pajzsát, manáját regeneráló és kézben lévő fegyverének újratöltését, megjavítását kezelő függvény.

### **static void changeWeapon(Character& player):**

A játékos az inventoryban lévő fegyverei közül választhat egyet, amit kézbevesz az előző helyett (ha van 1-nél több fegyvere), az inventory helyében.

---

## Combat

### Felelőssége:

A Combat osztály felelős a harci logika vezérléséért, a játékos és az ellenség közötti küzdelem lebonyolításáért. Kezeli a harci köröket, a játékos és az ellenség akcióit, a menekülést, a fegyverhasználatot, a gyógyítást, a szintlépés utáni jutalmak kiosztását, valamint a győzelem és vereség esetén történő eseményeket. Interakciót biztosít a játékos és a játék többi modulja (pl. Game, Character, InputHandler) között a harc során.

### Ősosztályok:

Nincs.

### Attribútumok:

Nincs.

### Metódusok:

#### *Public*

#### **static void start(Character& player, Character& enemy):**

Elindítja a harcot a játékos és az ellenség között. Kezeli a harc fő ciklusát, a körök váltását, győzelem és vereség esetén meghívja a megfelelő metódusokat, kezeli a szintlépést és a harcból való menekülést.

#### *Private*

#### **static void playerTurn(Character& player, Character& enemy):**

A játékos körét kezeli, lehetőséget ad támadásra, gyógyításra, fegyver javításra/újrátöltésre, menekülésre, fegyverváltásra, információk lekérésére vagy a kör kihagyására.

#### **static void enemyTurn(Character& enemy, Character& player):**

Ellenség körét bonyolítja le, automatikusan támad, javít vagy tölt fegyvert, ha szükséges.

#### **static void displayCombatInfo(const Character& player, const Character& enemy):**

Megjeleníti a játékos és az ellenség harc közbeni legfontosabb statisztikáit (név, életerő).

#### **static void displayVictoryMessage(const Character& player, const Character& enemy):**

Győzelem esetén megjeleníti a megfelelő üzenetet, valamint a szerzett tapasztalati pontokat.

#### **static void displayDefeatMessage(const Character& player):**

Vereség esetén megjeleníti a megfelelő üzenetet.

#### **static bool flee(Character& player):**

Kezeli a játékos menekülési próbálkozását a harcból; felhasználói visszajelzést kér (biztos vagy benne? igen/nem).

#### **static bool needHeal(Character& player):**

Meghatározza, hogy a játékosnak szüksége van-e gyógyításra vagy regenerálásra (karakterosztály-függő, warrior: shield, wizard: mana is szerepet játszik benne).

#### **static bool needRepair(Character& player):**

Meghatározza, hogy a játékos által használt fegyver javításra vagy újrátöltésre szorul-e (melee és ranged osztályú fegyverek esetén).

#### **static bool changeWeapon(Character& player):**

Eldönti, hogy a játékos tud-e fegyvert váltani (ha van több fegyver az inventoryban).

#### **static void manageLevelUpRewards(Character& player):**

Minden páros szintre lépéskor új fegyvert generál a játékos karakterosztályának megfelelően, és kezeli annak inventoryba helyezését, cseréjét vagy eldobását (törlését).

---

## Input Handler

### Felelőssége:

Az InputHandler osztály felelős a felhasználói bemenetek egységes és biztonságos kezeléséért. Segítségével szám-, szöveg- és igen/nem típusú adatokat lehet bekérni a felhasználótól, ellenőrzött formában. A hibás vagy érvénytelen bemeneteket kezeli, és csak akkor enged tovább a program, ha megfelelő adat érkezik.

### Ősosztályok:

Nincs.

### Attribútumok:

Nincs.

### Metódusok:

*Public*

**static int getIntInput(const std::string& prompt, int min, int max):**

Egész szám típusú bemenet bekérése a felhasználótól, adott minimum és maximum intervallumon belül. Hibás vagy érvénytelen bemenet esetén (nem szám, vagy kívül esik a tartományon) újra bekéri az adatot.

**static std::string getStringInput(const std::string& prompt):**

Szöveges bemenet bekérése a felhasználótól. A teljes sort beolvassa, szóközzel együtt.

**static bool getYesNoInput(const std::string& prompt):**

Igen/nem típusú kérdés bekérése a felhasználótól. Elfogadott válaszok: "yes", "y", "no", "n" (kisbetű). Hibás válasz esetén újra bekéri az adatot.

---

## Menu Manager

### Felelőssége:

A MenuManager osztály felelős a játék különböző menüinek és felhasználói választófelületeinek megjelenítéséért. Segítségével strukturáltan és egységesen jelennek meg a főmenü, karakterválasztó menü, harci menü, karakter- és inventory-információk, valamint egyéb, a játék során szükséges interaktív menük és visszajelzések. A felhasználó minden fontosabb döntési vagy információs pontján keresztülhívódik.

### Ősosztályok:

Nincs.

### Attribútumok:

Nincs.

### Metódusok:

*Public*

**static void displayMainMenu():**

Megjeleníti a főmenüt, ahol a játékos kiválaszthatja, hogy vándorol, regenerál, fegyvert vált, információkat néz meg vagy kilép a játékból.

**static void displayCharacterSelectionMenu(const std::string& playerName):**

Megjeleníti a karakterválasztó menüt, ahol a játékos kiválaszthatja karakterosztályát, személyre szóló megszólítással.

**static void displayCombatMenu():**

Megjeleníti a harci menüt, ahol a játékos választhat támadás, regenerálás, fegyverjavítás/újrátöltés, menekülés, fegyverváltás, információ megjelenítése és kör kihagyása között.

**static void displayPreCombatMenu():**

Megjeleníti a harc előtti menüt, ahol a játékos dönthet az ellenség megtekintése, saját információi lekérése vagy a menübe való visszalépés között.



#### **static void displayLookMenu():**

Megjeleníti a "közelebbről megnéz" menüt, ahol a játékos dönthet, hogy harcol vagy továbbmegy.

#### **static void displayInventoryMenu():**

Megjeleníti az inventory menüt, ahol a játékos fegyvereit/inventoryját láthatja.

#### **static void displayCharacterInfoMenu():**

Megjeleníti a karakterinformációs menüt, ahol a játékos a karakter, fegyver, inventory részletei vagy a visszalépés közül választhat.

#### **static void displayQuitConfirmation():**

Megjeleníti a kilépés megerősítő kérdést. (biztos vagy benne? igen/nem)

#### **static void displayInvalidChoice():**

Megjeleníti az érvénytelen választás hibaüzenetét.

#### **static void displayGameOverMenu():**

Megjeleníti a halál utáni vagy játék kijátszása utáni menüt, ahol a játékos dönthet újratekés vagy végleges kilépés között.

---

## Character

### Felelőssége:

A Character osztály felelős a játékos vagy ellenség karakterének alapvető állapotáért, tulajdonságaiért és viselkedéséért. Kezeli az életerőt, szintet, tapasztalati pontokat, fegyvereket, illetve a karakter főbb akcióit, mint például támadás, regenerálódás, szintlépés, fegyverkezelés. Lehetővé teszi a karakterek közötti harcot, sebzés és gyógyulás kezelését, valamint az inventory menedzselését (fegyverek hozzáadása, kiválasztása, cseréje).

### Össztályok:

Nincs, de virtuális metódusokat tartalmaz, így örökölhető alap.

### Attribútumok:

*Protected*

**std::string name:** A karakter neve.

**unsigned hp:** Jelenlegi életerő.

**unsigned maxHp:** Maximális életerő.

**unsigned level:** Karakter szintje.

**unsigned xp:** Aktuális tapasztalati pontok.

**unsigned maxXp:** Következő szinthez szükséges XP.

**static const unsigned maxWeaponCount:** Inventoryban tartható maximális fegyverszám (3db).

**std::vector<std::unique\_ptr<Weapon>> weapons:** Fegyverek listája (heterogén kollekció, sajátolva, unique\_ptr-rel kezelve).

**unsigned selectedWeaponIndex:** Jelenleg kiválasztott fegyver indexe.

**bool gotReward:** Szintlépési jutalom állapota. (megkapta-e már vagy még nem)

**bool isFleeing:** Menekülés állapota harcból.

### Metódusok:

*Public*

#### **Character(std::string name):**

Karakter létrehozása névvel, 1-es szinten, alap életerővel és XP-vel.

#### **Character(std::string name, unsigned level):**

Karakter létrehozása névvel és adott szinten, szinthez igazított életerővel és XP-vel.

#### **virtual ~Character():**

Virtuális destruktork.

#### **virtual void regenerate():**

Karakter regenerálja életerejét (alapértelmezett logika: max életerőig tölt).

#### **virtual void attack(Character& target):**

Támad egy másik karaktert, a kiválasztott fegyverrel sebez.

**unsigned getHealth() const:**

Jelenlegi életerő lekérése.

**unsigned getMaxHp() const:**

Maximális életerő lekérése.

**std::string getName() const:**

Karakter nevének lekérése.

**void changeHealth(int amount):**

Életerő módosítása adott értékkel (pozitív vagy negatív).

**bool isAlive() const:**

Életben van-e a karakter.

**unsigned getLevel() const:**

Szint lekérése.

**unsigned getExperience() const:**

Jelenlegi XP lekérése.

**unsigned getMaxExperience() const:**

Következő szint XP-jének lekérése.

**void gainXp(unsigned gained):**

XP növelése adott értékkel, és szintlépés kezelése.

**virtual void levelUp():**

Szintlépés logika: szint növelése, max életerő, XP frissítése, jutalom állapot alaphelyzetbe.

**bool checkLevelUp() const:**

Ellenőrzi, hogy a karakter elérte-e a szintlépési feltételeket (jutalmazás, páros szint stb.).

**virtual void wonTheBattle(const Character& enemy):**

Harc győzelme után XP-t kap, életerőt maximálisra állít, adott fegyver újratöltése/javítása.

**void setReward(bool gotReward):**

Jutalmazási állapot beállítása.

**bool wasRewarded() const:**

Jutalmazási állapot lekérdezése.

**void setFleeing(bool isFleeing):**

Menekülés állapotának beállítása.

**bool getFleeing() const:**

Menekülés állapotának lekérdezése.

**void selectWeapon(unsigned index):**

Inventoryből fegyvert választ ki.

**void takeWeapon(Weapon\* weapon):**

Fegyvert hozzáad az inventoryhoz (tulajdonjog átvétellel).

**void replaceWeapon(int index, Weapon\* newWeapon):**

Inventory adott helyén fegyvert cserél.

**void clearWeapons():**

Inventory kiürítése.

**void repairSelected():**

Kézben lévő fegyver javítása (ha javítható).

**void displayWeapons() const:**

Inventoryban lévő fegyverek listázása, a kiválasztott kiemelése.

**const std::vector<std::unique\_ptr<Weapon>>& getWeapons() const:**

Inventory fegyverlistájának lekérdezése (const referencia).

**Weapon\* getSelectedWeapon() const:**

Jelenleg kiválasztott fegyver lekérdezése (pointer, nullptr, ha nincs).

**unsigned getSelectedIndex() const:**

Kiválasztott fegyver indexének lekérdezése.

**unsigned getMaxWeaponCount() const:**

Maximális inventory-fegyverszám lekérdezése.

---

## Warrior

### Felelőssége:

A Warrior osztály a Character osztályból származó speciális karaktertípus, amely a harcos (Warrior) szerepét testesíti meg a játékban. Fő jellemzője a pajzs (shield) használata, amely extra védelmet nyújt az életerő mellett. Kezeli a pajzs értékét, regenerációját, szintlépéskor annak növelését, valamint a harc során speciális sebződés- és gyógyulásmechanikákat biztosít. A harcos alapértelmezett fegyvere egy közelharci fegyver (Melee).

### Ősosztályok:

- Character

### Attribútumok:

*Protected (örökölt attribútumok a Character-ből)*

*Private*

**unsigned shield:** Jelenlegi pajzsérték.

**unsigned maxShield:** Maximális pajzsérték.

### Metódusok:

*Public*

**Warrior(std::string name):**

Harcos példányosítása névvel, 1-es szinten, alap pajzsértékkel és alap közelharci fegyverrel.

**Warrior(std::string name, unsigned level):**

Harcos példányosítása névvel és megadott szinttel, szinthez igazított pajzs- és fegyverértékekkel.

**~Warrior():**

Destruktor, inventory (fegyverek) felszabadítása.

**unsigned getShield() const:**

Aktuális pajzsérték lekérése.

**unsigned getMaxShield() const:**

Maximális pajzsérték lekérése.

**void setShield(unsigned shield):**

Pajzs aktuális értékének beállítása.

**void levelUp():**

Szintlépés logika: szint, maximum életerő és maximum pajzs növelése, mindkettő feltöltése; majd az ős (Character) szintlépésének meghívása.

**void regenerate():**

Életerő és pajzs regenerációja (mindkettő max értékig tölt, vagy 50%-kal növel, ha nem teljes).

**void changeHealth(int amount):**

Sebződés/gyógyulás logika: először az életerő, majd a pajzs csökken vagy nő. Ha a pajzs elhasználódik, az életerő csökken, illetve fordítva.

**void wonTheBattle(const Character& enemy) override:**

Harc győzelme után a pajzs is maximálisra töltődik, majd az ős (Character) győzelmi logikája fut.

*Protected (örökölt Character metódusok)*

---

## Wizard

### Felelőssége:

A Wizard osztály a Character osztályból származó speciális karaktertípus, amely a varázsló szerepét testesíti meg a játékban. Kiemelt tulajdonsága a mana használata, amelyet varázslatokhoz (fegyverekhez) használ. Kezeli a mana értékét, annak regenerációját, szintlépéskor való növelését, valamint a varázstámadások során történő mana-fogyasztást. A varázsló alapértelmezett fegyvere egy varázspálca (Magic).

### Ősosztályok:

- Character

### Attribútumok:

*Protected (örökölt attribútumok a Character-ből)*

*Private*

**unsigned mana:** jelenlegi mana érték

**unsigned maxMana:** maximális mana érték

### Metódusok:

*Public*

**Wizard(std::string name)**

Varázsló példányosítása névvel, 1-es szinten, alap mana értékkel és alap varázsfegyverrel.

**Wizard(std::string name, unsigned level):**

Varázsló példányosítása névvel és megadott szinttel, szinthez igazított mana- és fegyverértékekkel.

**~Wizard():**

Destruktor, inventory (fegyverek) felszabadítása.

**unsigned getMana() const:**

Aktuális mana érték lekérése.

**unsigned getMaxMana() const:**

Maximális mana érték lekérése.

**void changeMana(int amount):**

Mana aktuális értékének növelése vagy csökkentése, korlátok között.

**void levelUp():**

Szintlépés logika: szint, maximális életerő és maximális mana növelése, mindkettő feltöltése; majd az űs (Character) szintlépésének meghívása.

**void regenerate():**

Életerő és mana regenerációja (mindkettő max értékig tölt, vagy 50%-kal növel, ha nem teljes).

**void attack(Character& target):**

Támadás varázsfegyverrel: mana költség levonása, majd az űs támadási logikájának meghívása.

**void wonTheBattle(const Character&) override:**

Harc győzelme után a mana is maximálisra töltődik, majd az űs (Character) győzelmi logikája fut.

*Protected (örökölt Character metódusok)*

---

## Archer

### Felelőssége:

Az Archer osztály a Character osztályból származó speciális karaktertípus, amely az íjász (Archer) szerepét testesíti meg a játékban. Fő jellemzője a távolsági fegyver (ranged weapon) használata, amelyet alapértelmezetten birtokol. Kezeli a fegyver újratöltését regeneráció során, valamint szintlépéskor az életerő növelését. Az íjász nem rendelkezik extra erőforrással (mint pajzs vagy mana), hanem a fegyverhasználatra fókuszál.

### Ősosztályok:

- Character

### Attribútumok:

*Protected (örökölt attribútumok a Character-ből)*

### Metódusok:

*Public*

**Archer(const std::string& name):**

Íjász példányosítása névvel, 1-es szinten, alap távolsági fegyverrel (Bow).

**Archer(const std::string& name, unsigned level):**

Íjász példányosítása névvel és megadott szinttel, szinthez igazított fegyverrel (Bow).

**~Archer() override:**

Destruktor, inventory (fegyverek) felszabadítása.

**void regenerate() override:**

Életerő regeneráció, valamint a kézben lévő távolsági fegyver újratöltése.

**void levelUp() override:**

Szintlépés logika: szint és maximális életerő növelése, majd az ős (Character) szintlépésének meghívása.

*Protected(örökölt Character metódusok)*

---

## Weapon

### Felelőssége:

A Weapon osztály egy absztrakt alaposztály, amely a játékban szereplő fegyverek közös tulajdonságait és viselkedését írja le. Felelős a fegyverek nevének, sebzésének kezeléséért, valamint egy egységes interfészen keresztül biztosítja a fegyverhasználat (use) és sebzés lekérdezés (getDamage) lehetőségét. Konkrét fegyvertípusok (pl. közelharci, varázs, távolsági) ebből származnak le, és implementálják az absztrakt műveleteket.

### Ősosztályok:

Nincs.

### Attribútumok:

*Protected*

**std::string name:** a fegyver neve (std::string)

**unsigned damage:** a fegyver sebzése (unsigned)

### Metódusok:

*Public*

**Weapon(std::string name, int damage = 20):**

Fegyver példányosítása névvel és opcionális sebzéssel, alapértelmezetten 20-szal. Hibás értékek esetén alapértelmezett értéket állít be.

**virtual ~Weapon():**

Virtuális destruktor, leszármazottaknál alapértelmezett.

**std::string getName() const:**

A fegyver nevének lekérése.

**virtual unsigned getDamage() const:**

A fegyver sebzésének lekérése.

**virtual void use() = 0:**

Absztrakt metódus, a fegyver használatának logikáját a leszármazottak valósítják meg.

---

## Melee

### Felelőssége:

A Melee osztály a Weapon osztályból származó közelharci fegyvereket reprezentálja. Kezeli a fegyver tartósságát (durability), élességét (sharpness), maximális tartósságát, valamint biztosítja a fegyver használatát, javítását és az aktuális sebzés lekérdezését. A közelharci fegyverek sebzése élességfüggő, használatból csökkenhet, illetve törhet, ekkor javítás szükséges.

### Ősosztályok:

- Weapon

### Attribútumok:

*Protected (örökölt attribútumok a Weapon-ből)*

*Private*

**unsigned durability:** a fegyver aktuális tartóssága

**unsigned maxDurability:** a fegyver maximális tartóssága

**unsigned sharpness:** a fegyver élessége (0-100 között)

### Metódusok:

*Public*

**Melee(std::string name, unsigned damage=25, unsigned durability=6):**

Közelharci fegyver példányosítása névvel, alap vagy megadott sebzéssel és tartóssággal. Az élesség alapból 100.

**~Melee():** Destruktor.

**void use() override:**

A fegyver használata; csökkenti a tartósságot és az élességet. Ha eltörött, nem használható.

**unsigned getDamage() const override:**

Aktuális sebzés lekérése, amely az élességtől függ.

**unsigned getDurability() const:**

Aktuális tartósság lekérése.

**unsigned getMaxDurability() const:**

Maximális tartósság lekérése.

**unsigned getSharpness() const:**

Aktuális élesség lekérése.

**bool isBroken() const:**

Igaz, ha a fegyver tartóssága 0 (törött állapot).

**bool isFullyRepaired() const:**

Igaz, ha a fegyver teljesen megjavított állapotban van.

**void repair():**

A fegyver teljes javítása (maximális tartósság és élesség visszaállítása).

## Magic

### Felelőssége:

A Magic osztály a Weapon osztályból származó varázsfegyvereket reprezentálja. Kezeli a fegyverhez tartozó mana költséget, valamint biztosítja a fegyver használatának logikáját és a mana szükséglet lekérdezését. A varázsfegyverek sebzést okoznak és használatukhoz mana szükséges.

### Össztályok:

- Weapon

### Attribútumok:

*Protected (örökölt attribútumok a Weapon-ből)*

*Private*

**unsigned manaCost:** a fegyver használatához szükséges mana mennyisége

### Metódusok:

*Public*

**Magic(std::string name, unsigned damage=20, unsigned manaCost=20):**

Varázsfegyver példányosítása névvel, alap vagy megadott sebzéssel és mana költséggel.

**~Magic():**

Destruktor.

**unsigned getManaCost() const:**

A fegyver mana költségének lekérdezése.

**void use() override:**

A fegyver használata; kiírja a sebzést és a mana költséget.

---

## Ranged

### Felelőssége:

A Ranged osztály a Weapon osztályból származó távolsági fegyvereket reprezentálja. Kezeli a fegyver lőszer mennyiségét (ammo), maximális lőszerszámát (maxAmmo), valamint biztosítja a fegyver használatának, újratöltésének (reload) és lőszerállapot lekérdezésének logikáját. A távolsági fegyverek csak akkor használhatók, ha van bennük lőszer, ellenkező esetben újra kell tölteni őket.

### Ősosztályok:

- Weapon

### Attribútumok:

*Protected (örökölt attribútumok a Weapon-ből)*

*Private*

**unsigned ammo:** a fegyver aktuális lőszermennyisége

**unsigned maxAmmo:** a fegyver maximális lőszermennyisége

### Metódusok:

*Public*

**Ranged(std::string name, unsigned damage=20, unsigned maxAmmo=4):**

Távolsági fegyver példányosítása névvel, alap vagy megadott sebzéssel és maximális lőszermennyiséggel. Az aktuális lőszer induláskor maximális értéken van.

**~Ranged() override = default:**

Destruktor (alapértelmezett).

**void use() override:**

A fegyver használata; csökkenti a lőszert, ha van. Ha elfogyott a lőszer, nem használható.

**unsigned getAmmo() const:**

Az aktuális lőszermennyiség lekérése.

**unsigned getMaxAmmo() const:**

A maximális lőszermennyiség lekérése.

**bool isOutOfAmmo() const:**

Igaz, ha a fegyverben nincs lőszer.

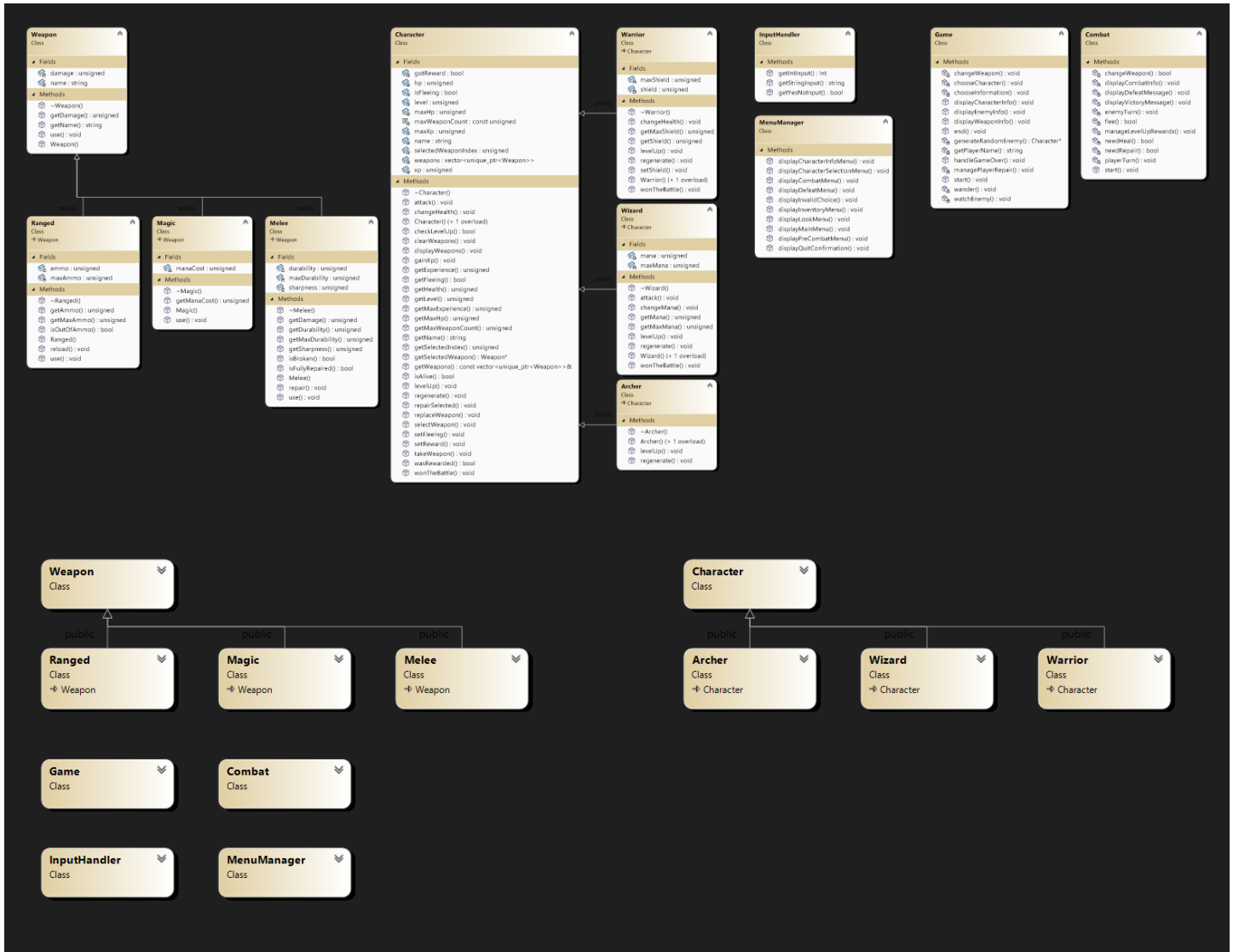
**void reload():**

Fegyver újratöltése maximális lőszerre, ha nem volt már teljesen feltöltve.

---



# UML osztálydiagram:



## Összegzés:

Többet sikerült megvalósítani a specifikációnál, mert a játék jutalmazza a játékos adott szintlépésenként, inputHandler és menuManager osztályokat használtam a könnyebb formázás és átláthatóság érdekében, külön osztályokban kezeltem a harc és a játék metódusait (Game, Combat).

Sok C++ nyelvi elemet és objektum orientált programozási elvet értettem meg mélyebben, jó gyakorlás és érdekes tanulási folyamat volt, illetve kellemes érzés, ahogy egy projekt a semmiből felépül és működik.

Továbbfejleszteni különböző nehézségi szintekkel (könnyű, normál, nehéz), többféle játékmóddal (PvP, Dungeon), a játék vizuális megjelenítés fejlesztésével (UI) lehetne, másrészt új fegyverek, karakterek és az eddigiek specifikus tulajdonságainak fejlesztésével, Item osztály illetve inventory létrehozásával, amiben fegyverek és „dolgoz” vannak (bájjital, amulett és armor slotba tehető: páncél), különböző effektek, módosítók bevezetése (pl.: kritikus sebzés esélyét növelje valamilyen tárgy), a mágikus és fizikai sebzés megkülönböztetése (pl.: adott páncél egyik ellen jobban véd mint a másik ellen) amilyen megoldásokkal sokat fejlődhetne a játék.

## Képernyőképek:

```
You are wandering in the wilderness...
An evil Shadow Archer (Level 1) has appeared!
```

```
DEEP DARK
1. Take a closer look
2. Check your character information
3. Back to menu
Choice: |
```

```
Your current weapon:
Name: Wand
Damage: 20
Mana cost: 20
```

```
INFO
Choose an option:
1. Check character
2. Check weapon
3. Display inventory
4. Back
Choice: |
```

```
Your inventory:
1. Wand
Selected weapon: Wand (1.)
```

```
INFO
Choose an option:
1. Check character
2. Check weapon
3. Display inventory
4. Back
Choice: |
```

```
The battle begins!

Player: Balázs | Health: 100/100
Enemy: Shadow Archer | Health: 100/100
```

```
Your turn!
1. Attack
2. Regenerate
3. Repair or reload
4. Flee
5. Change weapon
6. Display Info
7. Skip turn
Select action: |
```

```
Do you want to flee the battle? (yes/no): |
```

```
Are you sure you want to quit? (yes/no): no|
```

```
Your character:
Name: Balázs
Health: 100/100
Level: 1
Experience: 0/100
Class: Wizard
Mana: 100/100
Current weapon: Wand
Damage: 20
```

```
INFO
Choose an option:
1. Check character
2. Check weapon
3. Display inventory
4. Back
Choice: |
```

```
You chose to take a closer look at the enemy!
```

```
Enemy:
Name: Shadow Archer
Level: 1
Health: 100/100
Weapon: Bow
Ammo: 4/4
```

```
DEEP DARK
1. Fight!
2. Go deeper
Choice: |
```

```
-----
Your character:
Name: Balázs
Health: 80/100
Level: 1
Experience: 0/100
Class: Wizard
Mana: 80/100
Current weapon: Wand
Damage: 20
```

```
Your current weapon:
Name: Wand
Damage: 20
Mana cost: 20
```

```
Enemy:
Name: Dark Wizard
Level: 1
Health: 80/100
Mana: 80/100
Weapon: Wand
Mana cost: 20
-----
```