Alkalmazások fejlesztése projekt labor I.

(Levelező)

Készítette: Végh Vencel

Neptunkód: BQ5HU5

Kishuta, 2025.05.15.

Tartalom

1.Feladat leírása

A beadandó feladat célja a Neptun rendszer megvalósítása volt Java nyelven. A programnak képesnek kellett lennie több különböző típusú felhasználó (hallgató, oktató, adminisztrátor) kezelésére, valamint az ezekhez tartozó funkciók biztosítására. A fejlesztés során igyekeztem az objektumorientált szemléletet alkalmazni. A program háromféle felhasználói szerepet különböztet meg: hallgatót, oktatót és adminisztrátort. A belépés után minden szereplő egy saját menürendszert kap, ahol a neki megfelelő műveleteket hajthatja végre. A menük kezelése egyszerű, szöveges formában történik, a konzolon keresztül.

Hallgatóként a felhasználó megadhatja a Neptun-kódját, és ha az szerepel a rendszerben, akkor be tud lépni. A hallgatók jelentkezhetnek vizsgára, illetve lekérdezhetik, hogy aktív vagy passzív státuszban vannak-e.

Oktatóként a felhasználó megtekintheti a hallgatók listáját, és jegyet is adhat nekik egy megadott tantárgyból.

Az adminisztrátor külön azonosítóval léphet be, és hozzáférhet többféle kezelési funkcióhoz: új hallgató felvétele, meglévő hallgató törlése, illetve a státuszuk módosítása.

2.Program működésének a bemutatása

A program működését képekkel fogom prezentálni amelyekhez részletes leírást társítok, és kifejtem az adatok osztály, vagy metódus szerepét a programon belül.

2.1Program kódok és azok kifejtése

Main:

A program ezen részében történik meg az egyének példányosítása. Először két hallgatót hoztam létre, akik bekerülnek egy dinamikus listába (Array List). Minden hallgatónál megadom a nevét, a Neptun-kódját és azt is, hogy aktív státuszban van-e. Ezután létrehoztam egy oktatót, akinek a neve, Neptun-kódja és tanszéke is meg van adva, majd egy adminisztrátort is, aki szintén kap nevet és Neptun-kódot. Ezek az objektumok lesznek fognak szerepelni a programban. Minden példányosítás new kulcsszóval történik, és egy konstruktor segítségével történik meg az adatok beállítása.

```
public class Main {
    static ArrayList<Hallgato> hallgatok = new ArrayList<>();  7 usages
    static Scanner scanner = new Scanner(System.in);  15 usages

    public static void main(String[] args) {
        // Példányosított szereplők
        hallgatok.add(new Hallgato( nev: "Végh Csaba",  neptunKod: "ASZ694",  aktiv: true));
        hallgatok.add(new Hallgato( nev: "Kiss Anna",  neptunKod: "BDF123",  aktiv: true));

        Oktato oktato = new Oktato( nev: "Dr. Nagy Éva",  neptunKod: "XYZ789",  tanszek: "Informatika");
        Adminisztrator admin = new Adminisztrator( nev: "Tóth Anna",  neptunKod: "ADM456");
```

Főmenü ciklus:

A főmenü egy végtlen ciklus ami. A menü háromféle belépési lehetőséget kínál: hallgatói, oktatói és adminisztrátori. A felhasználó a szám beírásával választ, amit a program scanner.nextLine() segítségével beolvas, majd egy switch szerkezet értelmez.

Minden választható menüpont egy külön metódust indít el, amely az adott szerepkörhöz tartozó funkciókat biztosítja. Például, ha a felhasználó az "1"-est választja, akkor a hallgatoiMenu() metódus fut le. Ha a felhasználó a "0"-át írja be, a program leáll. Bármilyen más, érvénytelen választás esetén egy hibaüzenet jelenik meg.

```
20          while (true) {
21              System.out.println("\n--- Neptun rendszer ---");
22              System.out.println("1. Hallgatói bejelentkezés");
23              System.out.println("2. Oktatói bejelentkezés");
24              System.out.println("3. Adminisztrátori bejelentkezés");
25              System.out.println("0. Kilépés");
26              System.out.print("Válasszon: ");
27              String valasztas = scanner.nextLine();
28
29              switch (valasztas) {
30                  case "1":
31                      hallgatoiMenu();
32                      break;
33                  case "2":
34                      oktatoiMenu(oktato);
35                      break;
36                  case "3":
37                      adminisztratoriMenu(admin);
38                      break;
39                  case "0":
40                      System.out.println("Kilépés...");
41                      return;
42                  default:
43                      System.out.println("Érvénytelen választás.");
44              }
```

hallgatoiMenu() metódus:

A hallgatoiMenu() metódus akkor fut le, ha a felhasználó hallgatóként jelentkezik be. A metódus első lépése, hogy bekéri a Neptun-kódot, és megpróbálja megkeresni a megfelelő hallgatót a listában. Ha nem találja, hibaüzenetet ír ki és visszatér. Ha sikeres a bejelentkezés, egy újabb menü jelenik meg, amely kizárólag a hallgatói funkciókat kínálja.

A hallgató ezután három dolgot tehet: jelentkezhet vizsgára egy tantárgy megadásával, megnézheti az aktuális státuszát (aktív vagy passzív), vagy kiléphet a menüből. Minden választási lehetőséget a program egy switch szerkezettel kezel.

```
47
48      static void hallgatoiMenu() {  1 usage
49          System.out.print("Neptun-kód: ");
50          String kod = scanner.nextLine();
51          Hallgato h = keresHallgatot(kod);
52          if (h == null) {
53              System.out.println("Hallgató nem található.");
54              return;
55          }
56
57          System.out.println("Üdv, " + h.getNev() + "!");
58          while (true) {
59              System.out.println("\n--- Hallgatói menü ---");
60              System.out.println("1. Vizsgára jelentkezés");
61              System.out.println("2. Státusz megtekintése");
62              System.out.println("0. Kilépés");
63              System.out.print("Választás: ");
64              String val = scanner.nextLine();
65              switch (val) {
66                  case "1":
67                      System.out.print("Tantárgy neve: ");
68                      String tantargy = scanner.nextLine();
69                      h.vizsgaraJelentkezes(tantargy);
70                      break;
71                  case "2":
72                      System.out.println("Státusz: " + (h.isAktiv() ? "aktív" : "passzív"));
73                      break;
74                  case "0":
75                      return;
76                  default:
77                      System.out.println("Érvénytelen opció.");
78              }
79          }
80      }
```

oktatoiMenu() metódus:

Ez a metódus az oktatók belépése után jelenik meg. Kilistáz egy egyszerű menüt, amely lehetőséget ad a hallgatók kilistázására és arra, hogy az oktató jegyet adjon egy hallgatónak egy megadott tantárgyból. A bevitelt a program switch szerkezettel dolgozza fel.

Amikor a felhasználó a jegy adása opciót választja, először meg kell adnia a hallgató Neptun-kódját. A program ez alapján megkeresi a hallgatót, és ha megtalálja, elkéri a tantárgy nevét és a jegyet (1 és 5 között). Ezután az Oktato osztály jegyetAd() metódusával kiírja, hogy a hallgató milyen jegyet kapott.

```
    static void oktatoiMenu(Oktato oktato) {  1 usage
        System.out.println("Üdv, " + oktato.getNev() + " (Oktató)!");
        while (true) {
            System.out.println("\n--- Oktatói menü ---");
            System.out.println("1. Hallgatók listázása");
            System.out.println("2. Jegy adása hallgatónak");
            System.out.println("0. Kilépés");
            System.out.print("Választás: ");
            String val = scanner.nextLine();
            switch (val) {
                case "1":
                    listazHallgatok();
                    break;
                case "2":
                    System.out.print("Hallgató Neptun-kódja: ");
                    String kod = scanner.nextLine();
                    Hallgato h = keresHallgatot(kod);
                    if (h == null) {
                        System.out.println("Nem található.");
                        break;
                    }
                    System.out.print("Tantárgy neve: ");
                    String targy = scanner.nextLine();
                    System.out.print("Jegy (1-5): ");
                    int jegy = Integer.parseInt(scanner.nextLine());
                    oktato.jegyetAd(h, targy, jegy);
                    break;
                case "0":
                    return;
                default:
                    System.out.println("Érvénytelen opció.");
            }
        }
    }
```

Admin menü:

Az adminisztrátori menü akkor érhető el, ha a felhasználó helyesen adja meg az admin azonosítót (ADM456). Ez egy biztonsági lépés, amely kizárólag az adminisztrátor jogosultságait engedélyezi.

Sikeres belépés után a menü több opciót kínál: hallgatók listázása, új hallgató felvétele, meglévő törlése, valamint egy hallgató státuszának módosítása. Mindezeket egy switch szerkezet kezeli, a felhasználó választása alapján.

A hallgató felvételénél a program bekéri a nevet és a Neptun-kódot, majd új Hallgato objektumot hoz létre és hozzáadja a listához. A törlésnél a program a megadott Neptun-kód alapján keresi meg a hallgatót, és ha megtalálja, eltávolítja a listából. A státusz módosításnál a program először beolvassa a hallgató azonosítóját, majd egy logikai értéket (true vagy false), amely alapján a hallgató aktív vagy passzív lesz. Ez a menü biztosítja az adminisztrátori felhasználó számára a rendszer karbantartását: új adatok felvitelét, hibás adatok törlését és státuszkezelést.

```
static void adminisztratoriMenu(Adminisztrator admin) {  1 usage
    System.out.print("Admin azonosító (ADM456): ");
    String kod = scanner.nextLine();
    if (!admin.getNeptunKod().equalsIgnoreCase(kod)) {
        System.out.println("Hibás admin belépés.");
        return;
    }

    System.out.println("Üdv, " + admin.getNev() + " (Adminisztrátor)!");
    while (true) {
        System.out.println("\n--- Admin menü ---");
        System.out.println("1. Hallgatók listázása");
        System.out.println("2. Hallgató felvétele");
        System.out.println("3. Hallgató törlése");
        System.out.println("4. Státusz módosítása");
        System.out.println("0. Kilépés");
        System.out.print("Választás: ");
        String val = scanner.nextLine();
```

```
    switch (val) {
        case "1":
            listazHallgatok();
            break;
        case "2":
            System.out.print("Hallgató neve: ");
            String nev = scanner.nextLine();
            System.out.print("Neptun-kód: ");
            String neptun = scanner.nextLine();
            hallgatok.add(new Hallgato(nev, neptun,  aktiv: true));
            System.out.println("Hallgató felvéve.");
            break;
        case "3":
            System.out.print("Neptun-kód: ");
            String torlendo = scanner.nextLine();
            Hallgato torles = keresHallgatot(torlendo);
            if (torles != null) {
                hallgatok.remove(torles);
                System.out.println("Hallgató törölve.");
            } else {
                System.out.println("Nem található.");
            }
            break;
```

```
            case "4":
                System.out.print("Neptun-kód: ");
                String modosit = scanner.nextLine();
                Hallgato mod = keresHallgatot(modosit);
                if (mod != null) {
                    System.out.print("Új státusz (true/false): ");
                    boolean uj = Boolean.parseBoolean(scanner.nextLine());
                    admin.statuszModositas(mod, uj);
                } else {
                    System.out.println("Nem található.");
                }
                break;
            case "0":
                return;
            default:
                System.out.println("Érvénytelen opció.");
        }
    }
}
```

keresHalgatot() metódus:

Ez a metódus arra szolgál, hogy egy hallgatót megkeressen a rendszerben a Neptun-kódja alapján. A metódus végigmegy az összes hallgatón, és összehasonlítja a megadott kódot a listában szereplő hallgatók Neptun-kódjával. Az equalsIgnoreCase() metódus biztosítja, hogy a kis- és nagybetűk ne számítsanak, így a felhasználó nem köteles pontos karakterazonosságot megadni. Ha talál egyezést, visszaadja az adott hallgató objektumát, különben null értéket ad vissza, jelezve, hogy nincs ilyen hallgató a rendszerben. Ez a metódus kulcsfontosságú a hallgatók azonosításához több menüpontban is.

```
static Hallgato keresHallgatot(String kod) {  4 usages
    for (Hallgato h : hallgatok) {
        if (h.getNeptunKod().equalsIgnoreCase(kod)) {
            return h;
        }
    }
    return null;
}
```

listazHallgatok() metódus:

Ez a metódus arra szolgál, hogy az összes hallgatót kilistázza. Először ellenőrzi, hogy a hallgatók listája üres-e. Ha nincs egyetlen hallgató sem a rendszerben, akkor ezt kiírja a felhasználónak. Ha viszont vannak hallgatók, akkor mindegyiket végig járja, és kiírja a nevét, Neptun-kódját, valamint az aktuális státuszát (aktív vagy passzív). Ez a metódus az adminisztrátor és az oktató menüben használható, ahol fontos látni, kik szerepelnek a rendszerben.

```
static void listazHallgatok() {  2 usages
    if (hallgatok.isEmpty()) {
        System.out.println("Nincs hallgató a rendszerben.");
    } else {
        for (Hallgato h : hallgatok) {
            System.out.println(h.getNev() + " (" + h.getNeptunKod() + ") - " + (h.isAktiv() ? "aktív" : "passzív"));
        }
    }
}
```

Szemely osztály:

A Szemely osztály a program alapvető szülőosztálya. Ez az osztály tartalmazza a közös adatokat: a nev és a neptunKod mezőket, amelyek minden felhasználóra jellemzők. A konstruktor segítségével ezek az értékek az objektum létrehozásakor azonnal beállíthatók. Az osztály tartalmaz getter metódusokat is, amelyekkel kívülről biztonságosan lekérdezhető a név és a Neptun-kód. Ez az osztály biztosítja a közös alapot az öröklődéshez, és elősegíti a kód újra felhasználhatóságát.

```
class Szemely {  3 usages  3 inheritors
    protected String nev;  3 usages
    protected String neptunKod;  2 usages

    public Szemely(String nev, String neptunKod) {  3 usages
        this.nev = nev;
        this.neptunKod = neptunKod;
    }

    public String getNev() {  6 usages
        return nev;
    }

    public String getNeptunKod() {  3 usages
        return neptunKod;
    }
}
```

Hallgato osztaly:

A Hallgato osztály a Szemely osztály gyermekosztálya, azaz tőle örökli a nevet és a Neptun-kódot. Ez az osztály kifejezetten a hallgatói szerepkört valósítja meg, és tartalmaz egy aktiv nevű logikai változót, amely azt mutatja meg, hogy a hallgató aktív vagy passzív státuszban van-e. A konstruktor a szülőosztály adatait a super() hívással továbbítja, majd beállítja az aktiv értékét. Az isAktiv() és setAktiv() metódusok lehetővé teszik a státusz lekérdezését és módosítását. Emellett szerepel egy vizsgaraJelentkezes() metódus is, amellyel a hallgató jelezni tudja, hogy melyik tantárgy vizsgájára jelentkezik. Ez az osztály tehát a hallgatókkal kapcsolatos összes alapvető működést lefedi.

```
class Hallgato extends Szemely {  13 usages
    private boolean aktiv;  3 usages

    public Hallgato(String nev, String neptunKod, boolean aktiv) {  3 usages
        super(nev, neptunKod);
        this.aktiv = aktiv;
    }

    public boolean isAktiv() {  2 usages
        return aktiv;
    }

    public void setAktiv(boolean aktiv) {  1 usage
        this.aktiv = aktiv;
    }

    public void vizsgaraJelentkezes(String tantargy) {  1 usage
        System.out.println(nev + " jelentkezett a(z) " + tantargy + " vizsgára.");
    }
}
```

Oktato osztaly:

Az Oktato osztály szintén a Szemely osztály gyermekosztálya, és kifejezetten az oktatók adatait és feladatait képviseli a programban. Tartalmaz egy tanszek nevű adattagot, amely az oktató szakmai hátterét írja le. A konstruktor a név, Neptun-kód és tanszék megadásával hoz létre egy oktató objektumot. A jegyetAd() metódus egy hallgatónak ad jegyet egy megadott tantárgyból, és ezt kiírja a képernyőre.

```
class Oktato extends Szemely {  3 usages
    private String tanszek;  1 usage

    public Oktato(String nev, String neptunKod, String tanszek) {  1 usage
        super(nev, neptunKod);
        this.tanszek = tanszek;
    }

    public void jegyetAd(Hallgato hallgato, String tantargy, int jegy) {  1 usage
        System.out.println(hallgato.getNev() + " kapott egy " + jegy + "-öst a(z) " + tantargy + " tantárgyból.");
    }
}
```

Admin osztály:

Az Adminisztrator osztály a Szemely osztály gyermekosztálya, amely az adminisztrátori szerepkört testesíti meg a rendszerben. A konstruktor a szokásos módon megkapja a nevet és a Neptun-kódot, és ezeket a Szemely osztályon keresztül tárolja. Az osztály tartalmaz egy statuszModositas() nevű metódust, amely lehetővé teszi egy hallgató státuszának megváltoztatását. A metódus paraméterként kapja a hallgatót és az új státuszt (true = aktív, false = passzív), majd ezt beállítja, és visszajelzést ad a felhasználónak.

```
class Adminisztrator extends Szemely {  3 usages
    public Adminisztrator(String nev, String neptunKod) {  1 usage
        super(nev, neptunKod);
    }

    public void statuszModositas(Hallgato hallgato, boolean ujStatusz) {  1 usage
        hallgato.setAktiv(ujStatusz);
        System.out.println(hallgato.getNev() + " státusza módosítva: " + (ujStatusz ? "aktív" : "passzív"));
    }
}
```

3. Program indítása és használata

A Programot IntelliJ-ben készítettem el és 11-es JDK-t használtam hozzá. Az IntelliJ-ben a kódot a fent enyhén jobbra igazítva egy zöld kerettel ellátott belseje kitöltetlen háromszöggel lehet elindítani (személy szerint dark témát használok az OS rendszer színével megegyezőt így a futtató háromszög színe eltérhet).

3.1 Program futás közben

Főmenü:

A program futásakor elsőként a főmenü jelenik meg, amely lehetőséget biztosít a felhasználó számára, hogy kiválassza, milyen szerepkörben akar belépni a rendszerbe. A lehetőségek között szerepel a hallgatói, oktatói és adminisztrátori bejelentkezés, valamint a kilépés opció is. A menü szöveges formában jelenik meg, és a felhasználó a szám beírásával navigálhat.

```
--- Neptun rendszer ---
1. Hallgatói bejelentkezés
2. Oktatói bejelentkezés
3. Adminisztrátori bejelentkezés
0. Kilépés
Válasszon:
```

Hallgató Belépés:

Miután a felhasználó hallgatóként jelentkezik be a rendszerbe a Neptun-kódja megadásával, a program ellenőrzi, hogy az adott hallgató szerepel-e az adatbázisban. Ha igen, üdvözli név szerint, majd megjeleníti a hallgatói menüt. Ebben a menüben három lehetőség közül választhat a hallgató: vizsgára jelentkezhet, lekérdezheti az aktuális státuszát (aktív vagy passzív), illetve kiléphet a menüből.

A vizsgára jelentkezés során a program bekéri a tantárgy nevét, és megerősíti, hogy a hallgató jelentkezett a vizsgára. A státusz megtekintése esetén egyszerűen kiírja, hogy a hallgató jelenleg aktív vagy passzív jogviszonnyal rendelkezik.

```
Válasszon: 1
Neptun-kód: ASZ694
Üdv, Végh Csaba!

--- Hallgatói menü ---
1. Vizsgára jelentkezés
2. Státusz megtekintése
0. Kilépés
Választás:
```

```
Választás: 1
Tantárgy neve: Informatika
Végh Csaba jelentkezett a(z) Informatika vizsgára.

--- Hallgatói menü ---
1. Vizsgára jelentkezés
```

```
--- Hallgatói menü ---
1. Vizsgára jelentkezés
2. Státusz megtekintése
0. Kilépés
Választás: 2
Státusz: aktív
```

Okatató Belépés:

Az oktatói menübe belépve a rendszer köszönti az oktatót név szerint, majd felkínálja számára a lehetséges műveleteket: a hallgatók listázását, illetve jegy adását. A listázás opcióval az oktató átláthatja, kik szerepelnek a rendszerben, és milyen státuszban vannak.

Amikor jegyet ad, meg kell adnia a hallgató Neptun-kódját, a tantárgy nevét és a jegy értékét. Ezt követően a rendszer visszajelzést ad arról, hogy az adott hallgató milyen jegyet kapott az adott tantárgyból.

```
Válasszon: 2
Üdv, Dr. Nagy Éva (Oktató)!

--- Oktatói menü ---
1. Hallgatók listázása
2. Jegy adása hallgatónak
0. Kilépés
Választás:
```

```
--- Oktatói menü ---
1. Hallgatók listázása
2. Jegy adása hallgatónak
0. Kilépés
Választás: 1
Végh Csaba (ASZ694) - aktív
Kiss Anna (BDF123) - aktív
```

```
--- Oktatói menü ---
1. Hallgatók listázása
2. Jegy adása hallgatónak
0. Kilépés
Választás: 2
Hallgató Neptun-kódja: ASZ694
Tantárgy neve: Informatika
Jegy (1-5): 5
Végh Csaba kapott egy 5-öst a(z) Informatika tantárgyból.
```

Admin Belépés:

Az adminisztrátor a bejelentkezés után megkapja a saját menüjét, amely többféle, opciót kínál. A belépés biztonsági ellenőrzéssel kezdődik, amely során a felhasználónak meg kell adnia a helyes adminisztrátori azonosítót. Sikeres azonosítás után a program üdvözli az adminisztrátort, majd megjeleníti a lehetséges opciókat.

Az adminisztrátor listázhatja a jelenlegi hallgatókat, új hallgatót vehet fel a rendszerbe (név és Neptun-kód megadásával), törölhet hallgatót a Neptun-kód alapján, valamint módosíthatja egy hallgató státuszát aktívról passzívra vagy fordítva. A program minden művelet után visszajelzést ad, így a felhasználó pontosan látja, hogy a művelet sikeres volt-e.

```
--- Neptun rendszer ---
1. Hallgatói bejelentkezés
2. Oktatói bejelentkezés
3. Adminisztrátori bejelentkezés
0. Kilépés
Válasszon: 3
Admin azonosító (ADM456): ADM456
Üdv, Tóth Anna (Adminisztrátor)!
```

```
--- Admin menü ---
1. Hallgatók listázása
2. Hallgató felvétele
3. Hallgató törlése
4. Státusz módosítása
0. Kilépés
Választás:
```

```
Választás: 1
Végh Csaba (ASZ694) - aktív
Kiss Anna (BDF123) - aktív
```

```
2. Hallgató felvétele
3. Hallgató törlése
4. Státusz módosítása
0. Kilépés
Választás: 2
Hallgató neve: Végh vencel
Neptun-kód: BQ5HU5
Hallgató felvéve.
```

```
1. Hallgatók listázása
2. Hallgató felvétele
3. Hallgató törlése
4. Státusz módosítása
0. Kilépés
Választás: 3
Neptun-kód: BQ5HU5
Hallgató törölve.
```

```
Választás: 4
Neptun-kód: ASZ694
Új státusz (true/false): false
Végh Csaba státusza módosítva: passzív
```

Itt kicsit sűrű lett a képek elhelyezése, viszont szemlélteti az admin funkciók lehetőségeit például egy el írt keresztnév után törölni kellett az újonnan felvett hallgatót a Neptun rendszerből.