JEGYZŐKÖNYV

Alkalmazások Fejlesztése

Projekt Labor I.

Féléves feladat

Készítette: **Orosz Kristóf**

Neptunkód: **EYZWG9**

Dátum: 2025. május 10.

Sárospatak, 2025

Tartalomjegyzék:

Bevezetés

A féléves beadandóm során egy összetett Neptun rendszert készítettem több osztály segítségével. Az osztályok között szerepel például a hallgatók tantárgyfelvétele, vizsgára jelentkezés, jegyrögzítés, valamint az oktatók és adminisztrátorok közreműködése a tanulmányi adatok kezelésében.

A program egy absztrakt Szemely alaposztályra épül, amelyből három szereplőtípus öröklődik: Hallgato, Oktato és Adminisztrator. A rendszer további osztályai a Tantargy és Vizsga osztályok, amelyek a tanulmányi egységeket és a vizsgákat tartalmazzák.

A főosztály (Main) segítségével létrehozhatók és tesztelhetők a különböző típusú szerepkörök (hallgató, oktató, adminisztrátor), illetve a tantárgyak és vizsgák.

A fájlok tartalmának rövid leírása

- Jegyzőkönyv [Jegyzőkönyv a féléves beadandóról.]
- Beadando-alkalmazasok fejlesztese projekt lab [Tartalmazza a feladat leírást.]
- Szemely.java [A Személy osztályt tartalmazza.]
- Hallgato.java [A Hallgató osztályt tartalmazza.]
- Oktato.java [Az Oktató osztályt tartalmazza.]
- Adminisztrator.java [Az Adminisztrátor osztályt tartalmazza.]
- Tantargy.java [A Tantárgy osztályt tartalmazza.]
- Vizsga.java [A Vizsga osztályt tartalmazza.]
- Main. java [A főosztályt tartalmazza.]

1. Osztályok létrehozása

Az alábbi alfejezetekben a programom osztályainak magyarázatai láthatóak.

1.1 Személy osztály létrehozása

A Szemely osztály célja, hogy alapot biztosítson a többi fő felhasználói szerepköröknek.

```
package Beadando;

// Absztrakt osztály, amely egy általános személyt ír le.
// Ebből az osztályból nem lehet példányt létrehozni közvetlenül,
// de más osztályok örökölhetnek belőle.
public abstract class Szemely {

    // Privát adattagok (mezők), amelyek egy személy adatait tárolják:
    protected String nev;
    protected String azonosito;
    protected String cim;

    // Konstruktor: Létrehozza a Szemely példányt a megadott adatokkal
    public Szemely(String nev, String azonosito, String cim) {
        this.nev = nev;
        this.azonosito = azonosito;
        this.cim = cim;
    }

    // Getter metódus a 'nev' mező eléréséhez
    public String getNev() { return nev; }

    // Setter metódus a 'nev' mező módosításához
    public void setNev(String nev) { this.nev = nev; }

    // Getter metódus az 'azonosito' mező eléréséhez
    public String getAzonosito() { return azonosito; }

    // Setter metódus az 'azonosito' mező módosításához
    public void setAzonosito(String azonosito) { this.azonosito = azonosito; }

    // Getter metódus a 'cim' mező eléréséhez
    public String getCim() { return cim; }

    // Setter metódus a 'cim' mező módosításához
    public void setCim(String cim) { this.cim = cim; }
}
```

1.2 Hallgató osztály létrehozása

Ez a Java-osztály a Neptun rendszer hallgatói szereplőjét modellezi, és a Szemely absztrakt osztályból öröklődik.

```
package Beadando;

import java.util.ArrayList;
import java.util.List;

// A Hallgato osztály örökli a Szemely absztrakt osztályt
public class Hallgato extends Szemely {

    private String neptunKod;
    private boolean aktiv;

    private List<Tantargy> tantargyak = new ArrayList<>(); // A regisztrált
tantárgyak listája
    private int maxKredit = 30;             // Maximálisan felvehető kredit

    // Konstruktor: a Szemely adatok mellett a hallgató saját adatait is beállítja
    public Hallgato(String nev, String azonosito, String cim, String neptunKod) {
        super(nev, azonosito, cim);         // A szülőosztály konstruktorának
meghívása
        this.neptunKod = neptunKod;
        this.aktiv = true;                  // Alapértelmezett státusz: aktív
    }

    // Tantárgy regisztrálása, csak akkor, ha a maximális kreditkeretet nem lépi
túl
    public void regisztralTantargy(Tantargy t) {
        int osszKredit = tantargyak.stream()
                                   .mapToInt(Tantargy::getKredit)
                                   .sum();
        if (osszKredit + t.getKredit() <= maxKredit) {
            tantargyak.add(t);
            t.regisztralHallgato(this);
        } else {
            System.out.println("Nem lehet regisztralni, túl sok kredit!");
        }
    }

    // Vizsgára jelentkezés metódus
    public void jelentkezikVizsgara(Vizsga v) {
        v.jelentkezikVizsgara(this); // A Vizsga osztály fogja kezelni a
jelentkezést
    }

    // Státusz beállítása (pl. admin által)
    public void setAktiv(boolean aktiv) {
        this.aktiv = aktiv;
    }

    // Státusz lekérdezése
    public boolean isAktiv() {
        return aktiv;
    }
```

```
    @Override
    public String toString() {
        return "Hallgato {\n" +
                "\tNeptun kód: '" + neptunKod + "',\n" +
                "\tAktív: " + aktiv + ",\n" +
                "\tTantárgyak: " + tantargyak + ",\n" +
                "\tMax kredit: " + maxKredit + "\n" +
                "}";
    }
}
```

1.3 Oktató osztály létrehozása

Ez a Oktato osztály a Neptun rendszer oktató szereplőjét valósítja meg. Örökli a Szemely absztrakt osztályt, és tartalmaz oktató-specifikus adatokat és funkciókat.

```
package Beadando;

import java.util.ArrayList;
import java.util.List;

// Az Oktato osztály a Szemely osztályból öröklődik
public class Oktato extends Szemely {

    private String intezet;
    private String beosztas;
    private List<Tantargy> tantargyak = new ArrayList<>();

    // Konstruktor: beállítja az oktató alapadatait és saját mezőit
    public Oktato(String nev, String azonosito, String cim, String intezet, String
beosztas) {
        super(nev, azonosito, cim); // A szülőosztály adattagjainak beállítása
        this.intezet = intezet;
        this.beosztas = beosztas;
    }

    // Tantárgy hozzáadása az oktatóhoz
    public void hozzaadTantargy(Tantargy t) {
        tantargyak.add(t);
    }

    // Jegy rögzítése egy vizsgán: a Vizsga osztályon keresztül történik
    public void rogzitesJegy(Vizsga v, Hallgato h, int jegy) {
        v.rogzitesJegy(h, jegy);
    }


    @Override
    public String toString() {
        return "Oktato {\n" +
                "\tNév: " + nev + ",\n" +
                "\tAzonosító: " + azonosito + ",\n" +
                "\tCím: " + cim + ",\n" +
                "\tIntézet: " + intezet + ",\n" +
                "\tBeosztás: " + beosztas + ",\n" +
                "\tTantárgyak: " + tantargyak + "\n" +
                "}";
```

```
    }
}
```

1.4 Adminisztrátor osztály létrehozása

Ez az Adminisztrator osztály a Neptun rendszerben szereplő adminisztrátor szerepkört valósítja meg. Szintén a Szemely absztrakt osztályból öröklődik, és olyan metódusokat tartalmaz, amelyek a rendszer adminisztratív feladatait kezelik, például a hallgatói státusz módosítását és vizsgajegyek rögzítését.

```
package Beadando;

// Az Adminisztrator osztály a Szemely osztályból öröklődik
public class Adminisztrator extends Szemely {

    // Konstruktor: az adminisztrátor adatait állítja be (szülőosztálytól örökölt
mezők)
    public Adminisztrator(String nev, String azonosito, String cim) {
        super(nev, azonosito, cim);
    }

    // A hallgató státuszát módosítja (aktív vagy inaktív)
    public void modositStatusz(Hallgato h, boolean aktiv) {
        h.setAktiv(aktiv);
    }

    // Vizsgajegy rögzítése egy hallgatónak a vizsgához
    public void rogzitesJegy(Vizsga v, Hallgato h, int jegy) {
        v.rogzitesJegy(h, jegy); // A Vizsga osztály végzi a tényleges rögzítést
    }


    @Override
    public String toString() {
        return "Adminisztrátor {\n" +
               "\tNév: " + nev + ",\n" +
               "\tAzonosító: " + azonosito + ",\n" +
               "\tCím: " + cim + "\n" +
               "}";
    }
}
```

1.5 Tantárgy osztály létrehozása

A Tantargy osztály a Neptun rendszer tantárgyait modellezi. A tantárgy egy névvel, kóddal, kredittel listával rendelkezik.

```
package Beadando;

import java.util.ArrayList;
import java.util.List;

// Tantargy osztály – egy tantárgyat reprezentál a rendszerben
public class Tantargy {

    private String nev;
    private String kod;
    private int kredit;
    private List<Hallgato> regisztraltHallgatok = new ArrayList<>(); // A
tantárgyra regisztrált hallgatók listája

    // Konstruktor: a tantárgy fő adatait állítja be
    public Tantargy(String nev, String kod, int kredit) {
        this.nev = nev;
        this.kod = kod;
        this.kredit = kredit;
    }

    // Getterek az adattagokhoz:
    public String getNev() { return nev; }
    public String getKod() { return kod; }
    public int getKredit() { return kredit; }

    // Hallgató regisztrálása a tantárgyra
    public void regisztralHallgato(Hallgato h) {
        regisztraltHallgatok.add(h);
    }

    // Visszaadja a regisztrált hallgatók listáját
    public List<Hallgato> getRegisztraltHallgatok() {
        return regisztraltHallgatok;
    }
    @Override
    public String toString() {
        return nev + " (" + kod + ", " + kredit + " kredit)";
    }
}
```

1.6 Vizsga osztály létrehozása

Ez a Vizsga osztály a Neptun rendszerben a vizsgák kezelésére szolgál. Egy vizsga egy adott tantárgyhoz és dátumhoz kapcsolódik, és tárolja, hogy mely hallgatók jelentkeztek, illetve milyen jegyet kaptak.

```
package Beadando;

import java.util.Date;
import java.util.HashMap;
import java.util.Map;

// A Vizsga osztály egy adott tantárgyhoz és időponthoz kapcsolódó vizsgát ír le
public class Vizsga {

    private Tantargy tantargy; // A vizsgához tartozó tantárgy
    private Date datum; // A vizsga időpontja
    private Map<Hallgato, Integer> jegyek = new HashMap<>(); // Hallgatók és
vizsgajegyeik (null = még nincs jegy)

    // Konstruktor: tantárgy és vizsga dátumának beállítása
    public Vizsga(Tantargy tantargy, Date datum) {
        this.tantargy = tantargy;
        this.datum = datum;
    }

    // Vizsgára jelentkezés: a hallgatót felveszi a jegyek listájába, még jegy
nélkül (null)
    public void jelentkezikVizsgara(Hallgato h) {
        jegyek.putIfAbsent(h, null);
    }

    // Jegy rögzítése a hallgatónak, csak ha előtte jelentkezett
    public void rogzitesJegy(Hallgato h, int jegy) {
        if (jegyek.containsKey(h)) {
            jegyek.put(h, jegy);
    }


    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("Vizsga {\n");
        sb.append("\tTantárgy: ").ap
```

2. Főosztály létrehozása

Ez a Main osztály a teljes Neptun-rendszer tesztelésére és bemutatására szolgál. A main() metóduson belül különböző objektumokat hoztam létre: tantárgyakat, hallgatókat, oktatókat, adminisztrátorokat és vizsgákat. Teszteli a rendszer működését.

```
package Beadando;

import java.util.Date;

public class Main {
    public static void main(String[] args) {

        // --- Tantárgyak létrehozása ---
        Tantargy prog = new Tantargy("Programozás Alapjai", "PROG100", 5);
        Tantargy adatbazis = new Tantargy("Adatbáziskezelés", "DB101", 3);
        Tantargy op = new Tantargy("Operációs rendszerek", "OP102", 3);
        Tantargy adatszerkezet = new Tantargy("Adatszerkezetek és algoritmusok",
"DB103", 5);
        Tantargy matek = new Tantargy("Numerikus matematika", "DB104", 5);

        // --- Hallgatók létrehozása ---
        Hallgato h1 = new Hallgato("Kiss Péter", "", "Tokaj Hegyalja Egyetem",
"HAEEW23"); // nincs tantárgy felvéve

        Hallgato h2 = new Hallgato("Orosz Kristóf", "ABC123", "Tokaj Hegyalja
Egyetem", "BEAGH55");
        // Tantárgyak regisztrálása h2-nek
        h2.regisztralTantargy(prog);
        h2.regisztralTantargy(adatbazis);
        h2.regisztralTantargy(adatszerkezet);
        h2.regisztralTantargy(matek);

        Hallgato h3 = new Hallgato(" Kovács Tamás", "ADC123", "Tokaj Hegyalja
Egyetem", "AQDVT5");
        h3.regisztralTantargy(prog);
        h3.regisztralTantargy(adatbazis);
        h3.regisztralTantargy(adatszerkezet);
        h3.regisztralTantargy(matek);

        Hallgato h4 = new Hallgato("Nagy István", "ADF123", "Tokaj Hegyalja
Egyetem", "BEHKH41");
        h4.regisztralTantargy(prog);
        h4.regisztralTantargy(matek);

        // --- Oktatók létrehozása és tantárgyak hozzárendelése ---
        Oktato okt1 = new Oktato("Dr. Nagy Ákos", "O456", "Tokaj-Hegyalja
Egyetem", "Comenius Intézet", "Egyetemi adjunktus");
        okt1.hozzaadTantargy(prog);
        okt1.hozzaadTantargy(adatszerkezet);
        okt1.hozzaadTantargy(matek);

        Oktato okt2 = new Oktato("Dr. Kiss János", "O456", "Tokaj-Hegyalja
Egyetem", "Comenius Intézet", "Egyetemi docens");
        okt2.hozzaadTantargy(op);
        okt2.hozzaadTantargy(adatbazis);
```

```
        // --- Adminisztrátorok létrehozása és státusz módosítás ---
        Adminisztrator admin = new Adminisztrator("Molnár Anna", "A789", "Tokaj -
Hegyalja Egyetem");
        admin.modositStatusz(h1, false); // h1 inaktiválása

        Adminisztrator adminhely = new Adminisztrator("Kiss Noel", "A789", "Tokaj
- Hegyalja Egyetem");
        admin.modositStatusz(h1, false);


        // --- Vizsga létrehozása, jelentkezés és jegyrögzítés ---
        Vizsga vizsga1 = new Vizsga(prog, new Date()); // aktuális dátum

        h1.jelentkezikVizsgara(vizsga1);
        h2.jelentkezikVizsgara(vizsga1);

        admin.rogzitesJegy(vizsga1, h1, 4);
        admin.rogzitesJegy(vizsga1, h2, 4);

        // --- Kiíratás minden fontos szereplő és objektum esetén ---
        System.out.println(h1);
        System.out.println(h2);
        System.out.println(h3);
        System.out.println(h4);

        System.out.println(vizsga1);              // vizsga és eredmények
        System.out.println(prog);                 // tantárgyak
        System.out.println(adatbazis);
        System.out.println(op);
        System.out.println(adatszerkezet);
        System.out.println(matek);

        System.out.println(okt1);                 // oktatók
        System.out.println(okt2);

        System.out.println(admin);                // adminisztrátorok
        System.out.println(adminhely);
    }
}
```

3. Program futtatásának eredménye

```
Hallgato {
        Neptun kód: 'HAEEW23',
        Aktív: false,
        Tantárgyak: [],
        Max kredit: 30
}
Hallgato {
        Neptun kód: 'BEAGH55',
        Aktív: true,
        Tantárgyak: [Programozás Alapjai (PROG100, 5 kredit), Adatbáziskezelés (DB101, 3 kredit), Adatszerkezetek és algoritmusok (DB1
        Max kredit: 30
}
Hallgato {
        Neptun kód: 'AQDVT5',
        Aktív: true,
        Tantárgyak: [Programozás Alapjai (PROG100, 5 kredit), Adatbáziskezelés (DB101, 3 kredit), Adatszerkezetek és algoritmusok (DB1
        Max kredit: 30
}
Hallgato {
        Neptun kód: 'BEHKH41',
        Aktív: true,
        Tantárgyak: [Programozás Alapjai (PROG100, 5 kredit), Numerikus matematika (DB104, 5 kredit)],
        Max kredit: 30
}
Vizsga {
        Tantárgy: Programozás Alapjai (PROG100, 5 kredit),
        Dátum: Tue May 13 21:42:38 CEST 2025,
        Jegyek:
                Kiss Péter: 4
                Orosz Kristóf: 4
```

```
Programozás Alapjai (PROG100, 5 kredit)
Adatbáziskezelés (DB101, 3 kredit)
Operációs rendszerek (OP102, 3 kredit)
Adatszerkezetek és algoritmusok (DB103, 5 kredit)
Numerikus matematika (DB104, 5 kredit)
Oktato {
        Név: Dr. Nagy Ákos,
        Azonosító: O456,
        Cím: Tokaj-Hegyalja Egyetem,
        Intézet: Comenius Intézet,
        Beosztás:  Egyetemi adjunktus,
        Tantárgyak: [Programozás Alapjai (PROG100, 5 kredit), Adatszerkezetek és algoritmusok (DB103, 5 kredit), Numerikus matematika
}
Oktato {
        Név: Dr. Kiss János,
        Azonosító: O456,
        Cím: Tokaj-Hegyalja Egyetem,
        Intézet: Comenius Intézet,
        Beosztás:  Egyetemi docens,
        Tantárgyak: [Operációs rendszerek (OP102, 3 kredit), Adatbáziskezelés (DB101, 3 kredit)]
}
Adminisztrátor {
        Név: Molnár Anna,
        Azonosító: A789,
        Cím: Tokaj - Hegyalja Egyetem
}
Adminisztrátor {
        Név: Kiss Noel,
        Azonosító: A789,
        Cím: Tokaj - Hegyalja Egyetem
```