Tantárgyi tematika és félévi követelményrendszer
BPI1113L/BMI1302L Programozási technológiák

Oktató: Vályi Sándor Zoltán (PhD)
Oktatás:
Az órák hibrid formában mennek. Jelenlétiben is be lehet jönni a megadott terembe, s a Teams-en közvetítve is részt lehet venni, a Progtech2024 csoportban, 3 alkalom lesz,

okt 19, szo, 10:00--14:50, 6 tanóra 2 x 10 perc szünetet tartalmazva, B183
nov 9, szo, 10:00--14:50, 6 tanóra 2 x 10 perc szünetet tartalmazva, B182
dec 7, szo, 8:20 -- 12:25, 5 tanóra 10 perc szünettel, B182

Ezen kívül a tananyag átvételéhez kötelező a hasonló nappalis kurzuson felvett online előadások–demonstrációs gyakorlatok feldolgozása, amik minden hétfőn 7:30–10:30 között zajlanak, ugyanabban a Teams csoportban. Ezen előadások külső partner bevonásával történnek, az Epam Systems debreceni irodájának programozói tartják az online órák. A forráskódokat, prezentációkat a külsős Epam-oktatók a https://github.com/epam-nye-cooperation/epam-nye-progtech webhelyre helyezik el.

Vályi Sándor saját tananyagát a https://classroom.google.com/c/NzIzODcwNTI1ODU4?cjc=3luzvbk linken felvehető google classroom kurzuson helyezi el.

A foglalkozásokon történő részvétel:
- A gyakorlati foglalkozásokon a részvétel kötelező, legalább utólagos videó-feldolgozás formájában.
- ***A tárgy ismeret-átadásának módszere: a tárgy nappalis kurzusán menő meetingek felvételeinek – lehetőség híján offline – megtekintése, anyagának követése, házi feladatok megoldása. Szóval ez még az első órák előtt is teendőket ró a hallgatókra. Utána a levelezős órákon is – kevésbé részletesen – még átveszik az anyagot.***

Félévi követelmény: gyakorlati jegy

A félévközi ellenőrzések követelményei:
- Beadandó otthoni készítése és megvédése. A beadandó tartalma: egy asztali (desktop) parancssoros (command line) játékprogram, amely a kurzuson átvett témák nagy részét használja, előre definiált mélységben.
- Valamint egy elméletibb jellegű beadandó készítése, ami a programtervezési mintákról szól, és kritériumkövetelmény, pontszám nem jár hozzá, csak sikeres/nem sikeres értékelést kap. Ezt nem kell védeni. dolgozat cime ez*: Tervezési minták egy OO programozási nyelvben. MVC, mint modell-nézet-vezérlő minta és néhány másik tervezési minta.*

Az értékelés módja, ütemezése:
- Beadandó otthoni készítése és online megvédése. A beadandó-védés december 7-én, szombaton, az órán történik. A beadandó-védéshez a jelentkezési határidő december 4 éjfél, egy github URL megadásával. A védés projektjéhez az utolsó commit határideje szintén december 4, éjfél. A programtervezési mintás beadandó beadási határideje a 2023. december 4, éjfél..

Az érdemjegy kialakításának módja:
A félévi gyakorlati jegyet az összes pontszám alapján állapítjuk meg: 70% -- elégséges, 77% -- közepes, 85% -- jó, 93% -- jeles. Elégtelen gyakorlati jegy javítása a TVSz szerint lehetséges, csak egyszer a vizsgaidőszakban, a beadandó-védés pótlásával.

Féléves tematika:

Alkalom	Témakör	Megjegyzés

1.	Bemutatkozás, Eszközök telepítése	• JDK telepítés (11)
		• IDE (IntelliJ)
		• Verzió kezelés / Git alapok
		  • Git telepítés
		  • IDE / Tortoise Git

		● GitHub regisztráció ● repo létrehozás ● clone, pull, commit, push ● Maven telepítés
1.	Maven alapok	● Megmutatni hogy build tool nélkül nehéz a fejlesztés (javac, JAR) ● Maven életciklusok (clean, package, test, install) ● Alap maven projekt létrehozása ● pom.xml ● Függőség kezelés ● Pluginek
1.	Prog2 (Java / OO alapok) ismétlés	● Java ismétlés ● Exception handling ● Collection API ● Generikusok ● OO alapok ● Absztrakció, Polimorfizmus, Öröklődés, Enkapszuláció ● OO irányelvek ● magas kohézió, alacsony kötés, ● SOLID ● KISS ● YAGNI ● DRY ● Clean Code ● elnevezési konvenciók, beszédes változónevek, kommentek (inline comments, JavaDoc) ● Checkstyle Maven plugin
1.	Tesztelés	● Tesztelés céljának bemutatása ● tesztelési módszerek és szintek ● ● Elméleti alapozás: ● Egységtesztelés (fontosságának kihangsúlyozása, fejlesztés alatt már írni kell, TDD) ● Mockolás ● ● Gyakorlat ● JUnit5 ●
1.	Komplexebb közös programozási feladat	● Command Line játék (például Sudoku) alkalmazás vázának elkezdése ● Logolás ● VO objektumok (Object Methods fontossága: equals, hashCode, toString) ● Java Packaging (alap megközelítés model, ui, service, persistance package-ek)
1.	Nyomkövetés,Naplózás, dokumentáció	●
2.	Java Stream API	● funkcionális programozás, stream-ek kezelése
2.	JDBC	● JDBC a beadandó példában

2.	OO Tervezési minták	● Design Pattern (például: Singletone, Observer, Builder) ● Immutability
3.	Beadandó védés	

A beadandó projektről

Connect 4 parancssoros játék implementáció

- A félév során a hallgatóknak önállóan kell lefejleszteni egy Java parancssoros Connect-4 játékot
- Ennek leírása itt olvasható angolul: https://hu.wikipedia.org/wiki/Connect_four
    - A connect4 kétszemélyes stratégiai táblajáték, mely 1 db NxM-es (N és M pozitiv egész szám, 4 <=M <= N <=12), tipikusan 7x6-as táblán játszható. N -- sorok, M -- oszlopok száma.
    - Az oszlopok számozása tipikusan a,b,c, ... betűkkel történik, a soroké 1,2,3,..,N sorszámokkal -- de ettől nem függ a játékprogram.
    - Induláskor a tábla üres.
    - A két játékos közül az egyik a sárga szinű korongokat, a másik a pirosakat vezeti. A sárga szin lesz a humán játékosé, a piros a gépi játékosé. A sárga kezd.
    - A játékosok felülről csúsztatják bele a táblába a saját színeiket, így a jelrakások lehetőségei is korlátozottak a többi amőba típusú játékhoz képest. Tehát egy lépés megadásához mindig elég megadni, mely oszlopba fogjuk csúsztatni a korongunkat.
    - A gépi ellenfél ebben a félévben még rém egyszerű, csak random generál egy lehetséges oszlopot, mindegyiket egyenlő valószinűséggel.
    - Az a játékos nyer, amelyik függőlegesen, vízszintesen, vagy átlósan kirakott négyet a saját színéből.
- A védések az óra időpontjában fognak történni (a 3. alkalommal)
- Elvárások
    - Egy publikus GitHub repository létrehozása
    - A létrehozott Git repository tartalmazza a beadandó forráskódját
    - A repository tartalmaz egy megfelelő .gitignore fájlt annak érdekébe, hogy IDE vagy Maven specifikus ideiglenes fájlok ne kerüljenek fel a repository-ba
    - Egy Java 21-es Maven projekt létrehozása (pom.xml és Maven folder struktúra)
    - A Maven projekt az alábbi konfigurációkat tartalmazza:
        - Plugin-ek:
            - org.apache.maven.plugins.maven-jar-plugin - annak érdekében, hogy felkonfiguráljuk az alkalmazásunk belépési pontját (Main Class)
            - org.apache.maven.plugins.maven-assembly-plugin - annak érdekében, hogy egy függőségeket tartalmazó, futtatható JAR fájl jöjjön létre az alkalmazás build-elése eredményeként
            - org.jacoco.jacoco-maven-plugin - annak érdekében, hogy a megírt Egység tesztek kód lefedettségét tudjuk mérni

            - org.apache.maven.plugins.maven-checkstyle-plugin - annak érdekében, hogy a projekten elkövetett kód formázási hibákat és egyéb rossz praktikák automatikus detektáljunk
        - Függőségek:
            - JUnit5
            - Mockito
            - Logback
    - Az alkalmazás objektum-orientált modellezése
        - Az alkalmazásunkhoz szükséges VO (Value Object) osztályok létrehozása (ügyelve és figyelembe véve a "best practice"-eket: Object methods overriding, Immutability, stb)
        - A teljes Connect4 játék funkcionalitás lefejlesztésre került (lehetséges egy játékot végig játszani elejétől a végéig)
        - A játékmenet sem a main metódusban történjen, hanem egy Game vagy hasonló nevű objektum start)Ö metódushiivása történjen
    - Az induláskor egy szövegfájlból beolvas egy játékállást, ha nincs meg az input fájl, akkor üres pályáról indulunk
    - Az alkalmazás képes kezdetleges felhasználói interakciókat fogadni
        - egy szövegfájlból betölteni egy pályát
        - egy szövegfájlba kiÍrni egy pályát
        - Például: Játékos nevének bekérése, Játék elindítása, a játéktér kiiratása, Egy lépés fogadása a parancssoron, a lépés vizsgálata abból a szempontból, hogy alkalmazható-e; a lépés alkalmazása és az eredmény kiírása, stb
        - Itt nem határozunk meg kötelező elvárásokat, tetszőleges interakciók elegendőek
    - Egység tesztek 80% lefedettséget biztosítanak üzleti logikát tartalmazó osztályokra (tehát például VO osztályokra nem szükséges egységteszteket írni)
-
    - A projekt a `mvn clean install` parancs futtatására hiba nélkül fordul
    - Az alkalmazás egy adatbázisba lementi a játékosok nevét és azt, hogy hányszor nyertek
        - Az alkalmazás képes megjeleníteni parancssorban egy high score táblázatot (melyik játékos hány meccset nyert)
    - Opcionális (plusz pontért): egy aktuálisan folyamatban lévő játék állást az alkalmazás képes egy XML fájlba kimenteni és később visszatölteni (tehát a játékos onnan folytathatja a játékot, ahol korábban abba hagyta)
    - Egység tesztek 80% lefedettséget biztosítanak üzleti logikát tartalmazó osztályokra