

Tantárgyi tematika és félévi követelményrendszer

BPI1113L Programozási technológiák

A foglalkozásokon történő részvétel:

- A gyakorlati foglalkozásokon a részvétel kötelező. A félévi hiányzás megengedhető mértéke teljes idejű képzésben a tantárgy heti kontaktóraszámának háromszorosa. Ennek túllépése esetén a félév nem értékelhető (TVSz 8.§ 1.)
- A kurzus gyakorlati konzultáció jellegű, *az órákon való részvétel megköveteli, hogy a kurzus – nappalisokkal közös -- Teams-csoportjában addig elérhető órafelvételi videókat a hallgatók az órai részvétel előtt, előzetesen megtekintsék és a kiadott gyakorlatokat megoldják. A kurzus úgy indul, hogy a videón elmondottakat ismertnek tételezi föl.*

Félévi követelmény: gyakorlati jegy

Az értékelés módja, ütemezése:

- Beadandó otthoni készítése és 2 ütemben történő megvédése. A védések időpontja is az alábbi óraütemezésben kiolvashatók

A félévközi ellenőrzések követelményei:

- A beadandó tartalma: egy asztali (desktop) játékprogram, amely a kurzuson átvett témák nagy részét használja, előre definiált mélységben.

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.	Bemutakozás, Eszközök telepítése	<ul style="list-style-type: none">• JDK telepítés (11)• IDE (IntelliJ)• Verzió kezelés / Git alapok<ul style="list-style-type: none">• Git telepítés• IDE / Tortoise Git• GitHub regisztráció• repo létrehozás• clone, pull, commit, push• Maven telepítés
1.	Maven alapok	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• Java ismétlés<ul style="list-style-type: none">• Exception handling• Collection API• Generikusok• OO alapok<ul style="list-style-type: none">• Absztrakció, Polimorfizmus, Öröklődés, Enkapszuláció• OO irányelvek<ul style="list-style-type: none">• magas kohézió, alacsony kötés,• SOLID

		<ul style="list-style-type: none"> • KISS • YAGNI • DRY • Clean Code <ul style="list-style-type: none"> • elnevezési konvenciók, beszédes változónevek, kommentek (inline comments, JavaDoc) • Checkstyle Maven plugin • Gyakorlati feladat: egy szándékosan rossz (állatorvosi ló) forráskódot bemutatni és megkeresni a hibáit
1.	Tesztelés	<ul style="list-style-type: none"> • Tesztelés céljának bemutatása <ul style="list-style-type: none"> • tesztelési módszerek és szintek • testing pyramid • Elméleti alapoázás: <ul style="list-style-type: none"> • Egységtesztelés (fontosságának kihangsúlyozása, fejlesztés alatt már írni kell, TDD) • Mockolás • FIRST • Gyakorlat <ul style="list-style-type: none"> • JUnit5 • Mockito • Given-When-Then struktúra • teszt metódusok elnevezési konvenciója (testMethodShouldDoSomethingWhenCondition) • Jacoco, Surefire Maven plugins (80% coverage check felkonfigurálása)
1.	Komplexebb közös programozási feladat	<ul style="list-style-type: none"> • Command Line játék (például torpedó) alkalmazás vázának elkezdése • Logolás • VO objektumok (Object Methods fontossága: equals, hashCode, toString) • Java Packaging (alap megközelítés model, ui, service, persistance package-ek)
1.	Komplexebb közös programozási feladat	<ul style="list-style-type: none"> • Design Pattern (például: Singleton, Observer, Builder) • Immutability
2.	Beadandó védés part 1	<ul style="list-style-type: none"> • Elvárások az első beadandó védésre: <ul style="list-style-type: none"> • Maven project összerakása (pluginokkal) • Model osztályok leimplementálása • Command line interakciók kezelése (tesztekkel)
1.	IoC, DI (Spring Core)	
1.	IoC, DI (Spring Core)	
2.	JDBC	
2.	JDBC	
3.	XML Feldolgozás	
2.	Komplexebb közös programozási	

	feladat	
3.	Beadandó védés, 2. rész	