Magasszintű Programozási Nyelvek II. – Vizsgatematika

2015

1. OOP története. OOP alapfogalmak és alapelvek.

- Az OOP nyelvek és a procedurális nyelvek viszonya.
- OOP alapfogalmak: osztály, mező, metódus, property, konstruktor, példányosítás, objektum stb.
- Az OOP három alapelve és azok jelentősége.

2. Adatrejtés. Property.

- A mező fogalma.
- Védelmi szintek, azok kapcsolata a hatáskör fogalmával.
- Az adatrejtés értelme, jelentősége.
- Property fogalma. Property készítésének szintaktikai és szemantikai szabályai.
- Az elrejtett mező kívülről történő írási és olvasási lehetőségei, azok szokásos megoldása property-ken és metódusokon keresztül.
- Hasonlítsa össze a property-n keresztüli megoldást a metóduson keresztüli megoldással!

3. Osztályszintű és példányszintű mezők. Konstansok.

- A különbség az osztályszintű és példányszintű mezők között, különös tekintettel az élettartami kérdésekre.
- A példányszintű mezők deklarálásának és felhasználásának szintaktikája osztályon belül és kívül.
- Az osztályszintű mezők deklarálásának és felhasználásának szintaktikája osztályon belül és kívül.
- A konstans fogalma, deklarációja, hasonlósága az osztályszintű mezőkkel.

4. Osztályszintű és példányszintű metódusok.

- A példányszintű metódusok létrehozásának és felhasználásának szintaktikai és szemantikai szabályai.

- Az osztályszintű metódusok létrehozásának és felhasználásának szintaktikai és szemantikai szabályai.
- Ismertesse a metódusok belsejéből más metódusok meghívhatóságának problémakörét!
- Ismertesse a metódusok belsejéből más osztály belsejében deklarált mezők elérhetőségének problémakörét!
- Milyen feltételek mellett készítünk osztályszintű, és milyenek mellett példányszintű metódusokat?
- A 'this' kulcsszó és a metódusok kapcsolata.

5. Öröklődés szabályai és alkalmazása mezőkre.

- Az öröklődés szabályai.
- A mezők öröklődési szabályai példányszintű mezők esetén, a különböző védelmi szintek esetén.
- A mezők öröklődési szabályai osztályszintű mezők esetén, a különböző védelmi szintek esetén.
- A mezők újradefiniálási lehetőségei a különböző védelmi szintek esetén.

6. Öröklődés szabályai és alkalmazása metódusokra.

- Az öröklődés szabályai.
- A metódusok öröklődési szabályai példányszintű metódusok esetén, a különböző védelmi szintek esetén.
- A metódusok öröklődési szabályai osztályszintű metódusok esetén, a különböző védelmi szintek esetén.
- A metódusok újradefiniálási lehetőségei a különböző védelmi szintek esetén, a paraméterlista változása és nem változása esetén (overloading szabály, virtuális metódusok, egyszerű felüldefiniálás esete).
- A metódusok hívása esetén hogyan dől el, melyik metódus kerül végrehajtásra?

7. Korai és késői kötés.

- A korai kötés lényege, alkalmazása az OOP kódban. Adjon példát, amikor a korai kötés 'hibás' működésűvé válik. Ismertesse ennek körülményeit!
- A késői kötés lényege, alkalmazása az OOP kódban.
- Ismertesse, hogy dől el, hogy melyik kötés lesz korai, melyik késői!
- Ismertesse a késői kötés működésének támogatását VMT és DMT táblák esetén! Hasonlítsa össze a két technikát (előnyök, hátrányok)!

8. Konstruktorok.

- A konstruktor fogalma, feladatai.
- Hogyan lehet konstruktor írása nélkül is alaphelyzetbe állítani az objektum-mezőket, illetve milyen esetekben nem működik ez az alternatív módszer?
- A konstruktor írás szintaktikai szabályai.
- Alapértelmezett konstruktor.
- Hogyan hívhat meg egy konstruktorból másik saját osztálybeli, illetve ős osztálybeli konstruktort?
- Hogyan és miként hívjuk meg a konstruktort példányosításkor, és írja le mely mechanizmus szerint hajtódnak végre a konstruktorok az adott osztályból, illetve annak ősosztályaiban!
- Az osztályszintű konstruktor fogalma, írása, használata.
- Ismertesse az object factory fogalmát!

9. Típuskompatibilitás.

- Ismertesse az OOP területén értelmezett típuskompatibilitás fogalmát!
- Az Object típus kapcsolata a típuskompatibilitással, ennek használata kollekciók (listák, verem, sor, stb.) fejlesztésekor.
- Ismertesse a korai kötés során fellépő problémás eseteket!
- Dinamikus típus azonosítási lehetőség (’is’ operátor).
- Dinamikus típuskényszerítés lehetőségei (’as’ operátor), ennek lehetséges veszélyei.

10. Sealed, static és absztrakt osztályok.

- A sealed kulcsszó jelentősége, használatának szintaktikai, szemantikai szabályai.
- A static osztályok használatának szintaktikai és szemantikai szabályai.
- Az absztrakt osztályok fogalma, jellemzői.
- Az absztrakt osztályok és a virtuális metódusok előnyei, hátrányai, hasonlósága, különbsége egymással szemben.
- Az absztrakt metódusok használatának szintaktikai és szemantikai szabályai.
- Ismertessed az absztrakt osztályokra és gyermekosztályaikra vonatkozó további szabályokat!

11. A ’this’ kulcsszó. Indexelők.

- A this kulcsszó fogalma.
- A this kulcsszó használatának lehetőségei metódus, property, konstruktor belsejében.
- Ismertesse az indexelő fogalmát!

- Az indexelő használatának szintaktikai és szemantikai szabályai.
- Indexelő jelentősége.

12. Interface.

- Az interface fogalma.
- Milyen elemekből épülhet fel egy interface, mely elemeknek mik a szintaktikai szabályai?
- Ismertesse az interface felhasználását, jelentőségét az OOP stílusú programokban!
- Ismertesse a 'interface implementálás' fogalmát, szabályait!
- Az interface és az absztrakt osztály között különbözőségek.
- Milyen fontos interface-eket ismer a .NET Framework-ben?

13. Kivételkezelés.

- Milyen egyéb (hagyományos) hibajelzési technikákat ismer, melyiknek mik a korlátai, hátrányai?
- Ismertesse a kivétel mint hibajelzési technika jellemzőit!
- A kivétel programozott módon történő előidézésének (kivétel feldobása) technikája, annak szintaktikai szabálya, szemantikai következményei.
- A kivétel kezelésének módszere, annak szintaktikai és szemantikai szabályai.

14. Boxing és unboxing. Generikus típusok.

- Ismertesse a boxing művelet értelmét, hol és hogyan használja ezt a rendszer!
- Ismertesse az unboxing művelet értelmét, hol és hogyan használja ezt a rendszer!
- A generikus típus fogalma, készítésének szabályai.
- A generikus típus felhasználása a programozás során.
- Milyen generikus típusokat ismer a .NET Framework-ből? Hasonlítsa össze ezek használhatóságát a régebbi megoldásokkal!

15. Delegate.

- A delegate fogalma, ennek jelentősége, felhasználhatósága OOP környezetben.
- Callback függvények.
- Ismertesse a delegate-ek kezelésével kapcsolatos ismerteket!
- Hogy lehet delegate-et kezelni mint paraméter, mint ilyen típus mező?
- Hogyan tud több delegate-et kezelni egy időben lista, illetve esemény (event) segítségével?

16. Lambda kifejezés.

- A lambda kifejezés fogalma, jelentősége, felhasználhatósága OOP környezetben.
- A lambda kifejezések használatának szintaktikai és szemantikai szabályai.
- Generikus delegate-ek a .NET Framework-ben, a lambda kifejezések ezekkel való kapcsolata.
- LINQ bővítő metódusok fogalma, jelentősége, a lambda kifejezések ezekkel való kapcsolata.