Drugi međuispit iz Programskih paradigmi i jezika (verzija za vježbu)

7. svibnja 2008.

Važne napomene:

- Ispit se piše 60 minuta
- Odgovori na pitanja moraju biti koncizni, jasni i precizni. Ukoliko koncept treba objasniti, očekuje se što krade i jednostavnije objašnjenje.
- Zabranjeno je prepisivanje. Isto de se kazniti za to na Fakultetu predviđenim mjerama.

1. (1 bod) U nekoliko rečenica opišite što označavaju kratice CTS i BCL unutar .NET Frameworka.

Common Type System u .Net Frameworku je standard koji specificira kako su kako su tipovi podataka i njihove specificne vrijednosti pohranjene u memoriji racunala. Namjena mu je da pojednostavi dijeljenje informacija medu programima koji su napisani razlicitim programskim jezicima. (http://en.wikipedia.org/wiki/Common_Type_System)

Base Class Library je standardna biblioteka koja je dostupna svim programskim jezicima koji koriste .NET Framework. Ona sadrzi veliki broj razlicitih funkcija koje se cesto koriste, npr. citanje i pisanje datoteka, rad sa bazama podataka, funkcija za crtanje, ispisivanja podtaka u konzolu... (To je ono sto navodite na pocetku kad pistete program, npr. „using System.IO“ namespace sadrzi klasu StreamReader koja sluzi za citanje podataka iz datoteke)
(http://en.wikipedia.org/wiki/Base_Class_Library)

2. (1 boda) Navedite barem 5 sličnosti programskih jezika C++ i C.

1) Isti tipovi podataka
2) Isti operatori za primitivne tipove podataka (+,-,/,*...)
3) Iste naredbe za kontrolne strukture (if, for, while,...)
4) Ulazak u program definiran sa „main“
5) Funkcije definirane na isti nacin
6) Program se moze razdijeliti na vise datoteka

3. (2 boda) Objasnite što je to višeobličje u objektno orijentiranom programiranju. Ukratko opišite kako ga implementiramo u programskom jeziku C#.

Viseoblicje (polimorfizam) u OPP-u se ocituje tako sto omogucava da se slicna ponasanja razlicitih klasa izvedu u jednu zajednicku metodu, s tim da svaka klasa za sebe precizno definira zajednicko ponasanje. Polimorfizam se ostvaruje nasljedivanjem i nadjacavanjem.

```
namespace CommonClasses
{
    public interface IAnimal
    {
        string Name { get; }
        string Talk();
    }
}
namespace Animals
{
    public abstract class AnimalBase
    {
```

```
        public string Name { get; private set; }

        protected AnimalBase(string name) {
            Name = name;
        }
    }

    public class Cat : AnimalBase, IAnimal
    {
        public Cat(string name) : base(name) {
        }

        public string Talk() {
            return "Meowww!";
        }
    }

    public class Dog : AnimalBase, IAnimal
    {
        public Dog(string name) : base(name) {
        }

        public string Talk() {
            return "Arf! Arf!";
        }
    }
}

namespace Program
{
    public class TestAnimals
    {
        public static void Main(string[] args)
        {
            var animals = new List<IAnimal>() {
                new Cat("Missy"),
                new Cat("Mr. Mistoffelees"),
                new Dog("Lassie")
            };

            foreach (var animal in animals) {
                Console.WriteLine(animal.Name + ": " + animal.Talk());
            }
        }
    }
}
```

4. (2 bod) Navedite barem dvije stvari po čemu se razlikuju i barem dvije stvari po kojima su slične apstraktne klase i sučelja (ključna riječ `interface`).

Razlike:
Sucelje podrzava visestruko naslijedivanje, apstraktna klasa ne.
Sucelje nemoze sadrzavati konstruktore i destruktore, apstraktna klasa moze.
Sucelje moze naslijediti samo drugo sucelje, apstraktna klasa moze naslijediti drugu klasu i jedno ili vise sucelja.

Slicnosti:
I u suceljima i u apstraktnim klasama metode se deklariraju ali se ne definiraju (implementiraju).
I sucelje i apstraktna klasa se moraju naslijediti i implementirati metode istih da bi se koristile.

5. (2 bod) Detaljno objasnite princip objektnog dizajna po kojem je uputno više koristiti sadržavanje, a manje nasljeđivanje. Kršenje tog principa ilustrirajte prikladnim (proizvoljnim) odsječkom koda.

Sadrzavanje (kompozicija) je uputnije koristiti nego naslijedivanje jer kod potonjeg izmjene osnovne klase utjecu na izvedenu klasu te su moguci problemi kod ucahurivanja (enkapsulacije)
Za primjer vidi predavanje „Osoba, Putnik, Pilot“

6. (4 boda)
http://www.fer2.net/showpost.php?p=882363&postcount=8