ÖREBRO UNIVERSITET
HANDELSHÖGSKOLAN

Omtentamen

INFORMATIK, OBJEKTORIENTERAD PROGRAMMERING MED C#, FORTSÄTTNINGSKURS

Provkod: A002, Skriftlig tentamen, 4 högskolepoäng

Lärare: Mevludin Memedi, Panagiota Chatzipetrou och Johan Petersson

Datum: 2022-08-10
Tid: 14:15-18:15

Antal uppgifter: 10

Antal poäng: 24

För betyget godkänt (G) krävs 12 poäng dvs 50% av poängen.
För betyget väl godkänt (VG) krävs 18 poäng dvs 75% av poängen.

Instruktioner:

- Alla svar **skall skrivas direkt i svarsdokumentet**
- För att besvara de uppgifter som innefattar programkod är det tillåtet att använda den **inbyggda kodeditorn** som verktyg. **För varje uppgift kopiera din programkod från kodredigeraren och klistra in den i svarsdokumentet.**
- Kodeditorn bör ställas in på ”C#” för att ge bäst stöd.
- Presentera dina uppgiftssvar i nummerordning i ditt svarsdokument.
- Är uppgiften oklar, gör rimliga antaganden och redovisa dessa antaganden.
- Även inte helt fullständiga lösningar kan ge poäng. Lös de delar du kan.

Betrakta följande klasser:

```
public abstract class Person
    {
        private string personalNumber;
        public string PersonalNumber
        {
            get { return personalNumber;}
            set
            {
                if (Regex.IsMatch(value, @"^[0-9]+$") && value.Length<=12){
                    personalNumber = value;
                }
                else
                {
                    throw new System.ArgumentException("Personal number should contain
only digits and" +
                        " should not exceed 12 characters", "personalNumber");
                }
            }
        }
        public string FirstName { get; set; }
        public Person(string pn, string fn)
        {
            PersonalNumber = pn;
            FirstName = fn;
        }

        public virtual void ShowPersonInfo()
        {
            Console.WriteLine("--- Person info ---");
            Console.WriteLine("Personal Number: " + PersonalNumber);
            Console.WriteLine("First name: " + FirstName);
        }
    }


public class Student:Person
    {
        public List<string> Courses { get; set; }
        public Student(string pn, string fn, List<string> courses) // Svar på frågan 3
här
        {
            PersonalNumber = pn;
            FirstName = fn;
            Courses = courses;
        }

        public void PrintCoursesTaken()
        {
            foreach (var item in Courses)
            {
                Console.WriteLine(item);
            }
        }

        public override void ShowPersonInfo()
        {
            // Svar på frågan 4 här
        }
    }


public class Astronaut:Person
    {
        public List<string> PlanetsVisited { get; set; }
```

```
        public Astronaut(string pn, string fn, List<string> planetsVisited) // Svar på
frågan 3 här
        {
            PersonalNumber = pn;
            FirstName = fn;
            PlanetsVisited = planetsVisited;
        }

        public void PrintPlanetsVisisted()
        {
            foreach (var item in PlanetsVisited)
            {
                Console.WriteLine(item);
            }
        }

        public override void ShowPersonInfo()
        {
            // Svar på frågan 4 här
        }
    }
```

Uppgift 1 (2p)

Vad är användningen av nyckelordet `virtual` i metoden `ShowPersonInfo()` i klassen `Person`?

Uppgift 2 (2p)

Vad innebär det att definiera metoden `ShowPersonInfo()` med nyckelordet "`override`" i subklasserna?

Uppgift 3 (2p)

Uppdatera konstruktörerna i en av subklasserna för att möjliggöra korrekt initiering av deras instanser. (2p)

Uppgift 4 (2p)

Betrakta följande kod

```
Person person1 = new Student("198801011234", "Anders", new
List<string> { "C#", "IT Security", "Machine Learning"});

Person person2 = new Astronaut("194002025678", "Leif", new
List<string> { "Mars", "Mercury" });
```

Uppdatera metoden `ShowPersonInfo()` i en av subklasserna så att när de anropas på underklassernas objekt enligt följande

```
person1.ShowPersonInfo();
person2.ShowPersonInfo();
```

Följande resultat produceras i terminalen

```
--- Person info ---
Personal Number: 198801011234
First name: Anders
Courses taken:
C#
IT Security
Machine Learning
--- Person info ---
Personal Number: 194002025678
First name: Leif
Planets visited:
Mars
Mercury
```

Uppgift 5 (2p)

Klassen Person är en abstrakt klass. Vad betyder det?

Uppgift 6 (2p)

Tänk på följande tillägg till programmet:

```
    public interface IWakable
    {
        public string WalkingType();
    }


public class Person: IWakable
    {
        …

        public string WalkingType()
        {
            return "I walk with two legs";
        }
```

Förklara varför det kan vara en bra idé att använda ett interface så som ovan. Ge exempel och motivera.

Uppgift 7 (2p)

```
public class Persons<T>: List<T> where T:Person
```

Vad innebär det att definiera en lista enligt ovan?

Uppgift 8 (4p)

Skapa en metod `GetPeopleByNameDescending()` i klassen `Persons` som kommer att sortera personerna efter deras förnamn i fallande ordning. Om vi skulle gå igenom den nya listan med de två objekten skapade enligt ovan skulle först Leif visas följt av Anders.
Lösningen måste göras med LINQ. Inga lösningar med loops accepteras.

Uppgift 9 (2p)
Vilka är fördelarna med att använda en trelagersarkitektur när man utvecklar C#-applikationer?

Uppgift 10 (4p)

```
Func<int, int, int> del1 …
Action<int, int> del2 …
```

Förklara skillnaderna mellan dessa två generiska delegater.