Mälardalens Högskola TEN1 DVA222 2019
Akademin för Innovation, Design och Teknik

Objektorienterad programmering

Torsdag 15e augusti, 2019, 14:10–18:30
Lärare: Daniel Hedin, 021-107052 (15:00–17:00)
Två blad handskrivna anteckningar (båda sidor får användas)

Tentamen består av 42 poäng fördelat på 11 uppgifter. Betygsgränser: 3: 21p, 4: 28p, 5: 35p.

- Skriv tydligt; svårläsliga lösningar kan underkännas. Du kan skriva på engelska eller svenska.
- *Sortera sidorna* så uppgifterna kommer i ordning och *skriv inte på baksidan* av blad. Det minskar risken att en lösning missas.
- Om du inte förstår beskrivningen av en uppgift måste du fråga.
- Var noga med att redogöra för hur du resonerat och vilka antaganden du har gjort. Detta kan bidra till att du får poäng på din lösning även om den inte är helt korrekt.
- Planera ditt arbete; läs igenom hela tentan och gör sen de uppgifter du tycker verkar lätta först!

Allmänt (8p)

1) Vilka fördelar finns det med att använda accessor properties istället för att exponera ett fält. **(2p)**

2) Visa i kod hur man kan skapa ett fält som är read-only utanför klassen, men read-write inom klassen. **(2p)**

3) Exemplifiera i kod användandet av nyckelordet *abstract*. **(2p)**

4) Vad är en rotklass? **(2p)**

Klasser och arv (12p)

5) Visa hur man, med arv eller delegering, kan skapa en klass SizedStack som på konstant tid kan ge svar på hur många element som ligger på stacken. **(5p)**

```
public class Stack
{
    LinkedList<object> Data = new LinkedList<object>();

    public object Pop()
    {
        var result = Data.First;
        Data.RemoveFirst();
        return result;
    }

    public void Push(object element)
    {
        Data.AddFirst(element);
    }
}
```

6) Skapa en klass som håller reda på hur många objekt av klassen som skapats. **(3p)**

7) Vad händer när `(new C(14)).M(new C()` körs? Förklara i detalj hur körningen går till! **(4p)**

```
  class C
    {
        int F;
        public C() : this(10)
        {
        }
        public C(int f)
        {
            F = f;
        }
        public void M(C c)
        {
            Console.WriteLine(F + c.F);
        }
    }
```

Interfaces och generics (8p)

8) Implementera ett generiskt binärt sökträd som stöder insättning, uttagning, sökning och antalet insatta element. För att kunna jämföra elementen är det rimligt att kräva att de implementerar *IComparable*. **(8p)**

```
public interface IComparable<in T> {
  public int CompareTo(T other);

}
```

Bindning och överlagring (14p)

9) Vad skriver följande program ut vid körning? För poäng krävs att du förklarar varför för varje utskrift! **(6p)**

```
class MainClass
{
  public static void Main(string[] args)
  {
    Derived d = new Derived();
    Base b = d;

    b.Method1();
    b.Method1(3);
    b.Method2();
    b.Method2(5);

    d.Method1();
    d.Method1(3);
    d.Method2();
    d.Method2(5);
  }
}

class Base
{
  public void Method1()
    { Console.WriteLine("Base.Method1()"); }
  public virtual void Method1(int x)
    { Console.WriteLine("Base.Method1({0})", x); }

  public void Method2()
    { Console.WriteLine("Base.Method2()"); }
  public virtual void Method2(int x)
    { Console.WriteLine("Base.Method2({0})", x); }
}

class Derived : Base
{
  new public void Method1()
    { Console.WriteLine("Derived.Method1()"); }
  public override void Method1(int x)
    { Console.WriteLine("Derived.Method1({0})", x); }
}
```

10) Hur bör följande två metoder implementeras för att undvika kodduplicering? **(2+2p)**

```
class IntList
  {
    int[] data;

    // Create a new IntList that contains the items between
    // low index 0 and high index highIndex
    public IntList Slice(int highIndex)
    {
      // implementera!
    }

    // Create a new IntList that contains the items between
    // low index lowIndex and high index highIndex
    public IntList Slice(int lowIndex, int highIndex)
    {
      // implementera!
    }
  }
```

11) Följande operatoröverlagring innehåller ett fundamentalt fel. Beskriv felet och ge ett program som inte fungerar som man förväntar sig på grund av det. Ge även en korrekt implementation av additionsoperatorn. **(2+2p)**

```
class Vector
{
  int x, y;

  public Vector(int x, int y)
  {
    this.x = x;
    this.y = y;
  }

  public static Vector operator+(Vector v1, Vector v2)
  {
    v1.x += v2.x;
    v1.y += v2.y;
    return v1;
  }

  public override string ToString()
  {
    return string.Format("<{0},{1}>", x, y);
  }
}
```