Mälardalens Högskola
Akademin för Innovation, Design och Teknik

Objektorienterad programmering

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

Tentamen består av 43 poäng fördelat på 8 uppgifter.
Betygsgränser: 3: 21p, 4: 30p, 5: 36p.

- Skriv tydligt; svårläsliga lösningar underkänns. Du kan skriva på engelska eller svenska.
- 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.
- All kod på tentan är skriven i C#, men utnyttjar inga språkkonstruktioner som är specifika för C#.
- Om inget annat sägs är det fritt att välja mellan C# och C++ och att använda standardbibliotek i svaren.

Allmänt (4p)

1) Förklara och exemplifiera följande reserverade ord. **(4p)**

- interface
- virtual
- abstract
- protected

Klasser (8p)

2) Skapa klasserna `Cart` och `Item` som tillsammans skall implementera en enkel varukorg. En `Item` representerar en vara och skall innehålla varans *namn* och *pris*. En `Cart` representerar en varukorg och innehåller ett antal varor. Man skall kunna lägga till varor, räkna ut totalbeloppet och skriva ut ett kvitto (se exempel nedan - *uppmärksamma att samma vara grupperas på kvittot!*). Var noga med att följa principerna för god objektorienterad design. Följande program **(8p)**

```
public static void Main(string[] args)
{
  Cart c = new Cart();
  c.Add(new Item("Banan", 5));
  c.Add(new Item("Banan", 5));
  c.Add(new Item("Mango", 10));

  Console.WriteLine(c.GetTotal());
  c.PrintReceipt();
}
```

skall ge följande resultat vid körning där 20 är utskrift från `Console.WriteLine(c.GetTotal());` och resten utskrift från `c.PrintReceipt();`

```
20
Banan : 2st a 5 = 10
Mango : 1st a 10 = 10
--------------------
Totalt : 20
```

Notera att ni skall ge en fullständig implementation som ger resultatet ovan tillsammans med *Main*. Det räcker inte att ge klassernas interface.

Arv (10p)

3) Givet följande två klasser för rektanglar (Rectangle) och fyrkanter (Square). **(3+3p)**

```
class Rectangle
{
    int width;
    int height;

    public Rectangle(int width, int height)
    {
        SetSize(width, height);
    }

    public void SetSize(int width, int height)
    {
        this.width = width;
        this.height = height;
    }
}

class Square
{
    int width;
    int height;

    public Square(int size) {
        SetSize(size);
    }

    public void SetSize(int size) {
        this.width = size;
        this.height = size;
    }
}
```

Är det lämpligt att

1. Rectangle ärver av Square?
2. Square ärver av Rectangle?

Diskutera med utgångspunkt i *code reuse*, *Is-A*, och *public interface*.

4) Vad innebär begreppet (subtype) polymorphism och exemplifiera dess användning. **(4p)**

Dynamisk och statisk bindning (6p)

5) Följande program **(6p)**

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

    b.Method1();
    b.Method2();
    p.Method1();
    p.Method2();
    d.Method1();
    d.Method2();
  }
}

class Base
{
  public void Method1()
  { Console.WriteLine("Base.Method1"); }


  public virtual void Method2()
  { Console.WriteLine("Base.Method2"); }
}

class Derived : Base
{
  public new void Method1()
  { Console.WriteLine("Derived.Method1"); }


  public override void Method2()
  { Console.WriteLine("Derived.Method2"); }
}
```

skriver ut

```
Base.Method1
Base.Method2
Base.Method1
Derived.Method2
Derived.Method1
Derived.Method2
```

Förklara varje rad i utskriften baserat på hur statisk och dynamisk bindning fungerar så att det tydligt framgår varför varje metod kördes.

Generics och templates (6p)

6) Skapa en generisk klass, Pair, som lagrar par av värden. Implementationen måste uppfylla följande krav: **(4+2p)**

- det skall gå att skapa par där värdena har olika typer, samt
- man skall kunna läsa och uppdatera värdena individuellt.

Exemplifiera användningen av klassen genom att skapa ett par som innehåller en *int* och en *double*.

Överlagring (9p)

7) När och hur avgörs överlagring? **(3p)**

8) Utöka följande vektorklass (Vector) **(6p)**

```
class Vector
{
    double x, y;

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

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

med operatorer så att följande startklass (MainClass) fungerar

```
class MainClass
{
    public static void Main(string[] args)
    {
        Vector v1 = new Vector(2, 2);
        Vector v2 = new Vector(1, 3);
        Console.WriteLine(v1 + v2);
        Console.WriteLine(10 * v1);
        Console.WriteLine(v2 * 5);
    }
}
```

och ger följande utskrift

```
<3,5>
<20,20>
<5,15>
```