Mälardalens Högskola
Akademin för Innovation, Design och Teknik

Objektorienterad programmering

Torsdag 16e 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 44 poäng fördelat på 9 uppgifter.
Betygsgränser: 3: 22p, 4: 29p, 5: 36p.

- 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) Förklara begreppet inkapsling och varför det är ett centralt begrepp i objektorienterad programmering. **(3p)**

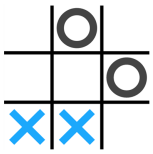
2) Namnge tre språkkonstruktioner i C# som möjliggör inkapsling och förklara hur de fungerar. **(4p)**

Klasser (8p)

3) Skapa en modellklass för spelet tre-i-rad (TicTacToe). **(4+4p)**

Klassen måste stödja två spelare, utplacering av spelarnas drag och kontroll om någon av spelarna vunnit. Beskriv hur klassen är tänkt att passa in en full implementation av spelet med ett grafiskt interface: hur tas dragen emot, hur renderas spelet, etc.

Var noga med att din implementation uppfyller de krav vi ställt på god objektorienterad design, ex.v., med avseende på inkapsling.



Figur 1: TicTacToe

Interface och arv (8p)

4) Är det någon skillnad på en fullt abstrakt klass och ett interface? **(2p)**

5) I programmet nedan definieras Square som en underklass till Rectangle. Beskriv för- och nackdelar med denna lösning. **(3p)**

```
class Rectangle {
  int width, height;

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

class Square : Rectangle {
  public void SetSize(int size) {
    SetSize(size,size);
  }
}
```

6) Under vilka förutsättningar är det lämpligt att använda sig av arv? **(3p)**

Generics (10p)

7) Skapa en generisk enkellänkad lista som tillåter insättning av nya element först och sist, genomlöpning av alla element på linjär tid samt utläsning av antal element som finns i listan. Exemplifiera användningen av din lista genom att skapa ett program som sätter in fyra valfria element och därefter går igenom listan och skriver ut dem. **(6+4p)**

Var noga med att din implementation uppfyller de krav vi ställt på god objektorienterad design, ex.v., med avseende på inkapsling.

Bindning och överlagring (10p)

8) Förklara hur dynamisk binding fungerar. **(3p)**

9) Hur och när avgörs det vilken metod som ska anropas om det finns fler än en metod med samma namn? **(3p)**

10) Vad skrivs ut av följade program **(4p)**

```
class Base {
  public void Method1() {
    Console.WriteLine("Base.Method1()");
  }
  public virtual void Method2() {
    Console.WriteLine("Base.Method2()");
  }
};

class Derived : Base {
  new public void Method1() {
    Console.WriteLine("Derived.Method1()");
  }
  public override void Method2() {
    Console.WriteLine("Derived.Method2()");
  }
};

public class MainClass {
  public static void Main(String[] args) {
    Derived o1 = new Derived();
    Base o2 = o1;
    o1.Method1();
    o1.Method2();
    o2.Method1();
    o2.Method2();
  }
}
```