Mälardalens Högskola
Akademin för Innovation, Design och Teknik

Tentamen objektorienterad programmering

Torsdag 9e juni, 2022, 14:30 – 18:30

Tentamen består av 44 poäng fördelat på 8 uppgifter.
Betygsgränser: 3: 50%, 4: 70%, 5: 90%

Tänk på

- Du kan skriva på svenska eller engelska.
- 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!

1) (3+3p) Hur fungerar följande access modifiers? Förklara med ord och ge kodexempel som illustrerar skillnaderna.

- public
- protected
- private

2) (1+2p) Vilken access modifier används om man inte uttryckligen anger en? Varför har man gjort det valet?

3) (4p) Vad innebär begreppet inkapsling och varför är det ett centralt begrepp i objektorienterad programmering?

4) (4p) I kursen har vi diskuterat fyra krav som måste vara uppfyllda för att det skall vara lämpligt att använda arv. Beskriv två av dessa (vilka två är valfritt).

5) (9p) Implementera ett generiskt tertiärt sökträd som stöder insättning, sökning och antalet insatta element. Ett tertiärt sökträd har tre barn där det vänstra delträdet lagrar värden som är mindre än föräldern, det mittersta barnet lagrar element som är samma som föräldern och det högra barnet lagrar element som är större än föräldern. Sökningsmetoden skall returnera antalet förekomster av det eftersökta i trädet.

För att kunna jämföra elementen är det rimligt att kräva att de implementerar *IComparable*.

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

}
```

6) (6p) Skapa en klass Vector som representerar en flyttalsvektor. Klassen skall stödja operatorer (inte metoder!) för addition av två vektorer, skalning samt beräkning av längden. Längdberäkningen skall göras som en property. För att påminna er om dessa operationer antag att $v_1 = (x_1, y_1)$ och $v_2 = (x_2, y_2)$ är vektorer. Då skall gälla:

- Addition: $v_1 + v_2 = (x_1, y_1) + (x_2, y_2) = (x_1 + x_2, y_1 + y_2)$.
- Skalning: $v * k = k * v = (k * x, k * y)$, för $v = (x, y)$.
- Längd: $|v| = |(x, y)| = \sqrt{x^2 + y^2}$

7) (6p) Vad skrivs ut av följade program och varför? Förklara i termer av statisk och dynamisk binding och statiska och dynamiska typer. Att bara ge utskriften ger inga poäng.

```
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); }
}
```

8) (6p) 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). Var noga med att följa principerna för god objektorienterad design.

```
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 : 5
Banan : 5
Mango : 10
Total : 20
```