Några C# strukturer

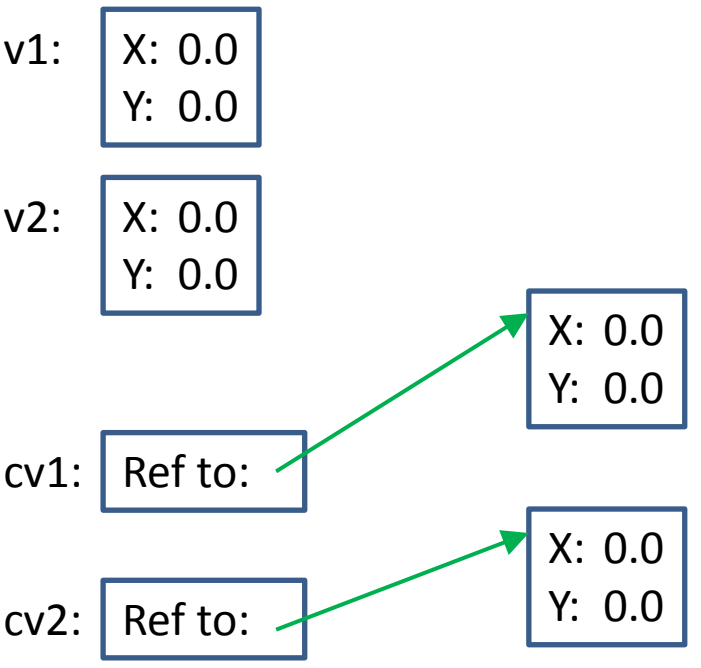
Struct (jämfört med Class)
Problemet med struct som property
Interface

struct resp. class

```
struct Vec2 {
    public float X, Y;
}
class CVec2 {
    public float X, Y;
}
//Grundläggande skillnad!
static public void TestStructClass() {
    Vec2 v1 = new Vec2(), v2 = new Vec2();
    CVec2 cv1=new CVec2(), cv2=new CVec2();
    //alla v1.X, v2.Y, cv2.X etc. är nu =0
    v1.X = cv1.X = 1;
    v1 = v2;
    cv1 = cv2;
    //Vad är nu v1.Y resp. cv1.Y?
    v2.X = 2;
    cv2.X = 2;
    //vad är v1.X och cv1.X?
}
```

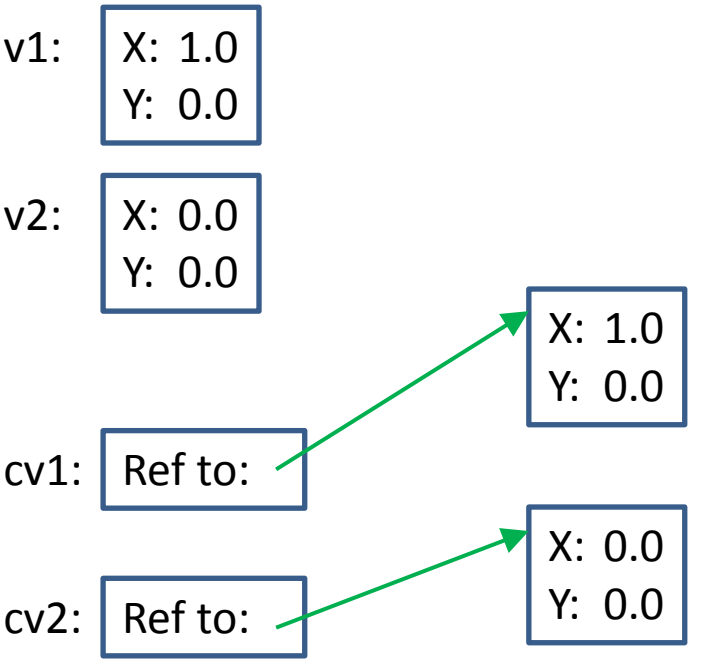
struct resp. class

```
struct Vec2 {
    public float X, Y;
}
class CVec2 {
    public float X, Y;
}
//Grundläggande skillnad!
static public void TestStructClass() {
    Vec2 v1 = new Vec2(), v2 = new Vec2();
    CVec2 cv1=new CVec2(), cv2=new CVec2();
    //alla v1.X, v2.Y, cv2.X etc. är nu =0
```



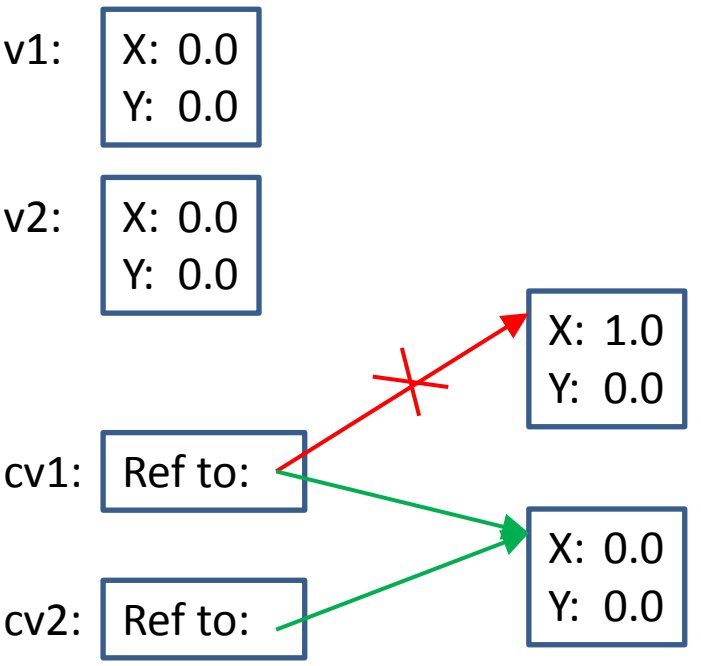
struct resp. class

```
struct Vec2 {
    public float X, Y;
}
class CVec2 {
    public float X, Y;
}
//Grundläggande skillnad!
static public void TestStructClass() {
    Vec2 v1 = new Vec2(), v2 = new Vec2();
    CVec2 cv1=new CVec2(), cv2=new CVec2();
    //alla v1.X, v2.Y, cv2.X etc. är nu =0
    v1.X = cv1.X = 1;
```



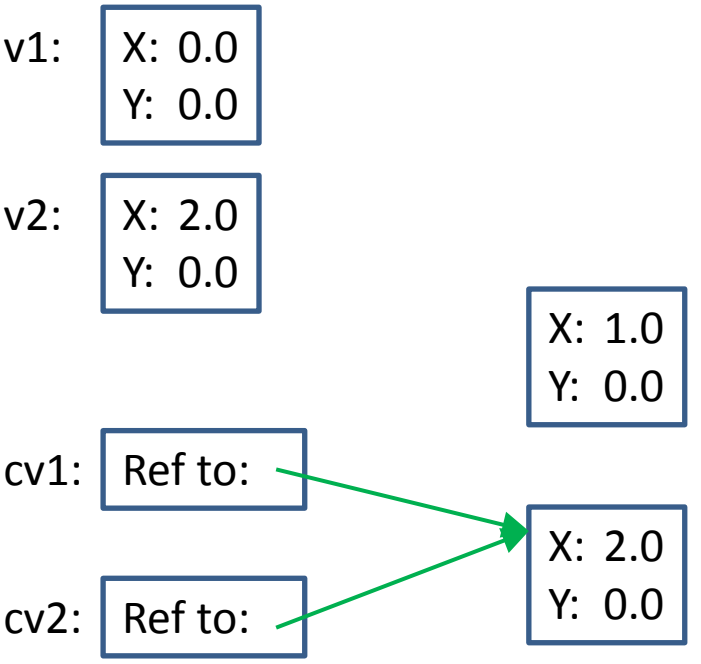
struct resp. class

```
struct Vec2 {
    public float X, Y;
}
class CVec2 {
    public float X, Y;
}
//Grundläggande skillnad!
static public void TestStructClass() {
    Vec2 v1 = new Vec2(), v2 = new Vec2();
    CVec2 cv1=new CVec2(), cv2=new CVec2();
    //alla v1.X, v2.Y, cv2.X etc. är nu =0
    v1.X = cv1.X = 1;
    v1 = v2;
    cv1 = cv2;
    //Vad är nu v1.Y resp. cv1.Y?
```



struct resp. class

```
struct Vec2 {
    public float X, Y;
}
class CVec2 {
    public float X, Y;
}
//Grundläggande skillnad!
static public void TestStructClass() {
    Vec2 v1 = new Vec2(), v2 = new Vec2();
    CVec2 cv1=new CVec2(), cv2=new CVec2();
    //alla v1.X, v2.Y, cv2.X etc. är nu =0
    v1.X = cv1.X = 1;
    v1 = v2;
    cv1 = cv2;
    //Vad är nu v1.Y resp. cv1.Y?
    v2.X = 2;
    cv2.X = 2;
    //vad är v1.X och cv1.X?
}
```



Skillnader struct jämfört med class

- v1=v2 betyder att värdena kopieras
- cv1=cv2 betyder att de pekar på samma sak
- struct kan varken ärvas eller ärva
- struct kan implementera interface
- struct parameterar kopieras vid anropet

Struct property problemet

```
class TestIt {
    public Vec2 v;
    public CVec2 cv = new CVec2();
    public Vec2 Vprop { get; set; }
    public CVec2 CVprop { get; set; }
}

static public void StructProperty() {
    TestIt ti = new TestIt();
    ti.cv.X = 4;
    ti.CVprop.X = 5;
    ti.v.X = 2;
    ti.Vprop.X = 3;  //Error: Cannot modify the return value of
                     // 'TestIt.Vprop' because it is not a variable
}
```

- ti.cv, ti.CVprop, ti.v är alla referenser till en variabel (v eller cv)
- ti.Vprop är en kopia av v.
- Undvik att ha en Property som är en struct!

Interface, Exempel

- Vi vill kunna ha alla objekten i ett spel i samma lista för att kunna göra vissa saker med dem på ett enhetligt sätt.
  - Update och Draw samt ta bort de objekt som är ”döda”.
- Som alternativ till att låta dem alla ärva från ett gemensamt basobjekt kan vi ha ett interface som de alla ska uppfylla.

Interface exampel

```
interface IGameObject {
    void Update(GameTime gametime);
    void Draw(SpriteBatch sb);
    bool IsDead { get; }
}

class UnMutable : IGameObject {
    Point pos;
    Texture2D tex;
    public bool IsDead { get; set; }
    public void Update(GameTime gt) { }
    public void Draw(SpriteBatch sb) {
        sb.Draw(tex
            , new Vector2(pos.X, pos.Y)
            , Color.White);
    }
    //Constructor is needed also
}
```

```
class Moving : IGameObject {
    Vector2 pos, speed;
    Texture2D tex;

    public bool IsDead { get; set; }
    public void Update(GameTime gt) {
        pos += speed *
(float)gt.ElapsedGameTime.TotalSeconds;
        if (pos.X + tex.Width < 0
              || pos.X > 800
              || pos.Y + tex.Height < 0
              || pos.Y > 600)
            IsDead = true;
    }
    public void Draw(SpriteBatch sb) {
        sb.Draw(tex, pos, Color.White);
    }
    //Constructor is needed also
}
```

Interface användning:

```
List<IGameObject> objectList =
            new List<IGameObject>();


protected override void
Update(GameTime gameTime) {
    foreach (IGameObject o
                    in objectList)
        o.Update(gameTime);

    for (int i = objectList.Count-1;
           i >= 0;
           i--)
        if (objectList[i].IsDead)
            objectList.RemoveAt(i);

    base.Update(gameTime);
}
```

```
protected override void
Draw(GameTime gameTime) {
    GraphicsDevice.Clear(
            Color.CornflowerBlue);
    spriteBatch.Begin();

    foreach (IGameObject o
                    in objectList)
        o. Draw(spriteBatch);
    spriteBatch.End();

    base.Draw(gameTime);
}
```

Interface

- Liknar en helt abstrakt klass.
  Men metoderna i ett interface är inte virtual
- Ett interface implementerar inget utan talar bara om vad som ska finnas.
- Ett interface kan ärva från andra interface
- En klass kan ärva från en klass och många interface.
- Klassen måste implementera allt i Interfacet

Interface

- Ett interface kan innehålla
  - methods
  - properties
  - events
  - Indexers

```
Moving myMoving = new Moving();
IGameObject myIGameObject;
```

- En referens till en klass konverteras automatiskt till sina interface

```
myIGameObject = myMoving;
```

- Man måste göra “cast” om man har en interfacereferens och vill ha tag på en klassreferens.

```
myMoving = (Moving)myIGameObject;
```