Lab2 - bibliotek för hantering av geometiska figurer.

I denna lab ska ni skapa en Visual Studio solution som innehåller ett **class library** enligt nedan specifikation. Ni kan arbeta självständigt eller i par, vilket ni själva föredrar. Senaste tid för inlämning är ...

Börja med skapa följande klass i biblioteket:

```
public abstract class Shape
 {
      public abstract Vector3 Center { get; }
      public abstract float Area { get; }
}
```

Skapa därefter dessa två *publika abstrakta* klasser **Shape2D** och **Shape3D** som båda ärver av Shape. I Shape2D ska du lägga till en public abstract property **Circumference** av typ float, och i Shape3D ska du lägga till en public abstract property **Volume** av typ float. För båda dessa properties gäller att man endast ska kunna läsa deras värden (dvs man ska inte kunna skriva till dem).

Fem klasser av geometriska figurer:

Nu ska ni skapa klasser som ärver av antingen Shape2D eller Shape3D. Alla klasserna ska implementera de abstakta properties som de ärver av respektive klass på ett sådant sätt att de ger korrekta värden för mittpunkt(Center), omkrets, area och volym. Om ni inte vet hur man räknar ut vissa saker, t.ex volym på ett klot, så försök hitta formlerna på internet och implementera dem i er kod.
Alla klasser ska även override:a ToString() metoden för att skriva ut namnet på formen tillsammans med centerpunktens position, följt av övriga värden (tex radie, eller höjd/bredd).

De fem klasserna som ska implementeras är **Circle**, **Rectangle** och **Triangle** som ska ärva av **Shape2D**, samt **Sphere** och **Cuboid** som ska ärva av **Shape3D.**

Circle (Shape2D)

Klassen Circle ska ha en (1) konstruktor som tar en Vector2 center som första parameter, och en float radius som andra parameter. Den ska override:a ToString() på så sätt att t.ex:

Console.WriteLine(new Circle(new Vector2(3.0f, 4.0f), 2.0f));
=> "**circle @(3.0, 4.0): r = 2.0**"

Rectangle (Shape2D)

Denna klass ska ha en konstruktor som tar parametrar: Vector2 center, Vector2 size (dvs. höjd/bredd), samt en alternativ konstruktor: Vector2 center, float width (som sätter både höjd och bredd till samma värde).

Den ska även implementera en property IsSquare som returnerar true om höjd och bredd är lika (annars false).

ToString() => "**rectangle @(3.0, 4.0): w = 4.0, h = 5.0**" (square om w == h).

Triangle (Shape2D)

Konstruktorn tar tre parameterar p1, p2, p3 av typ Vector2, som beskriver de tre punkter som utgör triangeln. Tänk på att Center också behöver beräknas.

ToString() => "**triangle @(3.0, 1.0): p1(0.0, 0.0), p2(3.0, 3.0), p3(6.0. 0.0)**"

Cuboid (Shape3D)

Denna klass ska ha en konstruktor som tar parametrar: Vector3 center, Vector3 size (dvs. höjd/bredd/djup), samt en alternativ konstruktor: Vector3 center, float width (som sätter höjd, bredd djup till samma värde).

Den ska även implementera en property IsCube som returnerar true om höjd, bredd och djup är lika (annars false).

ToString() => "**cuboid @(3.0, 4.0, 5.0): w = 4.0, h = 5.0, l=2.0**" (cube om w == h == l).

Sphere (Shape3D)

Konstruktor med parametrarna Vector3 center, float radius.

ToString() => “**sphere @(0.0, 1.0, 0.0): r = 3.2**”

Lägg till metod för att generera random shapes

När ni är klara med alla klasser ska ni lägga till en publik statisk metod **GenerateShape** i basklassen **Shape**. Denna metod ska slumpa ett tal 0-6 för att instantiera en av 7 figurer (circle, rectangle, square, triangle, cuboid, cube, sphere) och returnera i form av en Shape.

De olika figurernas värden ska också slumpas. GenerateShape kommer alltså behöva anropa konstruktorn för de olika klasserna med randomvärden på alla parameterar.

Men, vi vill också att GenerateShape ska finnas i två utföranden, en som inte tar några parametrar (dvs alla värden slumpas), och en där man kan skicka in en Vector3 position för att ange mittpunkten på figuren (då slumpas endast övriga värden såsom bredd, radie etc). Tänk på att till klasser som ärver av Shape3D kan ni skicka in position som den är, medan ni får skapa en Vector2 av X och Y, för de klasser som ärver av Shape2D (och alltså tar en Vector2 för mittpunkt).

Tänk också på att för Triangle kan ni bara slumpa två av punkterna. Den tredje punkten måste räknas ut för att få mittpunkten på rätt ställe.

Skapa ett projekt i samma solution som använder sig av ert class library

Detta projekt ska skapa en lista med 20 random shapes, genom att anropa GenerateShape(). Loopa sedan igenom listan och skriv ut alla shapes till konsollen.

Räkna samtidigt ut summan av omkretserna på alla trianglar i listan,
den genomsnittliga arean av alla Shapes i listan,
samt hitta den Shape3D som har störst volym av alla i listan.

Presentera detta snyggt och tydligt på konsollen när loopen är klar.

Tips & Hjälp

Vector2 och Vector3 är 2- respektive 3-dimensionella vektorer som har properties X, Y, respektive X, Y, Z (floats).

Längden på en vektor är avståndet mellan punkten vektorn beskriver och origo (nollpunkten). Använd metoden Length() för att får längden ex myVec3.Length();

Vector2.Zero ger en 2D vektor som motsvarar new Vector2(0.0f, 0.0f);
Vector3.Zero ger en 3D vektor som motsvarar new Vector3(0.0f, 0.0f, 0.0f);

Vector2.One => new Vector2(1.0f, 1.0f);
Vector3.One => new Vector3(1.0f, 1.0f, 1.0f);

Om man multiplicerar en vektor med en float så är det samma som att multiplicera varje ingående del i vektorn med samma värde.
ex. Vector3.One * 5.0f => new Vector3(5.0f, 5.0f, 5.0f);

Redovisning

Lämna in uppgiften på ithsdistans med en kommentar med github-länken.

Om ni jobbat i par måste både skriva länken till github, samt vem ni jobbat med.

Betygskriterier

För godkänt:

- Alla klasser ska vara publika.
- Koden ska fungera enligt ovan beskrivning.
- Man ska kunna få ut korrekt mittpunkt, area, omkrets (2D) och volym (3D) på samtliga geometriska figurer (enligt ovan).
- .IsSquare och .IsCube ska korrekt ange om det är en kvadrat eller kub.
- Båda versionerna av GenereteShape ska fungera korrekt.
- ToString() ska vara implementerad så att alla shapes skrivs ut korrekt.
- Projektet som använder klassen ska ta fram en lista med 20 random objekt, skriva ut alla objekten, samt trianglarnas total omkrets, alla objekts genomsnittliga area, och ange det objekt med störst volym.

För väl godkänt krävs även:

- Koden ska vara väl strukturerad och lätt att förstå
- Lösningen ska inte innehålla massa onödig kod.
- Allting ska fungera vid inlämning. Gör det inte det så kommer ni få tillbaks uppgiften med möjlighet att fixa det som krävs för godkänt. Men ni har alltså bara en chans på er att få VG. Så var noga med testning.
- Inlämning sker före deadline.
- Man ska även implementera extrauppgiften enligt nedan (IEnumerable).

OBS! Extra uppgift som krävs för VG!

Implementera **IEnumerator** och **IEnumerable** interfacen på klassen **Triangle** på ett sådant sätt att man kan använda en foreach på instanser av Triangle för att loopa igenom de tre vektorer som beskriver triangeln.

Vi ska alltså kunna få ut/loopa igenom triangelns punkter genom att skriva t.ex:

```
Triangle t = new Triangle(Vector2.Zero, Vector2.One, new Vector2(2.0f, .5f));

foreach (Vector2 v in t)
{
   Console.WriteLine(v);
}
```