



Linnéuniversitetet

Kalmar Vaxjö

Laborationsanvisning

Geometriska figurer

Steg 2, laborationsuppgift 3



Författare: Mats Looch

Kurs: Inledande programmering med C#

Kurskod: 1DV402

Upphovsrätt för detta verk

Detta verk är framtaget i anslutning till kursen Inledande programmering med C# vid Linnéuniversitetet.

Du får använda detta verk så här:

Allt innehåll i verket Geometriska figurer av Mats Loock, förutom Linnéuniversitetets logotyp, symbol och kopparstick, är licensierad under:



Creative Commons Erkännande-IckeKommersiell-DelaLika 2.5 Sverige licens.
<http://creativecommons.org/licenses/by-nc-sa/2.5/se/>

Det betyder att du i icke-kommersiella syften får:

- kopiera hela eller delar av innehållet
- sprida hela eller delar av innehållet
- visa hela eller delar av innehållet offentligt och digitalt
- konvertera innehållet till annat format
- du får även göra om innehållet

Om du förändrar innehållet så ta inte med Linnéuniversitetets logotyp, symbol och/eller kopparstick i din nya version!

Vid all användning måste du ange källan: "Linnéuniversitetet – Inledande programmering med C#" och en länk till <https://coursepress.lnu.se/kurs/inledande-programmering-med-csharp> och till Creative Common-licensen här ovan.

Innehåll

A. Uppgift	5
Problem	5
Formelsamling för area och omkrets	5
Klasshierarki	6
Klassen Shape	6
Klassen Ellipse	7
Klassen Rectangle	7
Klassdiagram över Program och ShapeType	8
Klassen Program	8
Den uppräkningsbara typen ShapeType	10
A-krav	10
Läsvärt	10
B. Uppgift	13
Problem	13
Formelsamling för area och omkrets	13
Klasshierarki	14
Klassen Shape	14
Klassen Ellipse	15
Klassen Rectangle	16
Klassdiagram över Program och ShapeType	16
Klassen Program	16
Den uppräkningsbara typen ShapeType	17
B-krav	17
Läsvärt	18
C. Uppgift	19
Problem	19
Formelsamling	19
Klasshierarki	20
Klassen Shape	21
Klassen Shape2D	21
Klassen Ellipse	23
Klassen Rectangle	23
Klassen Skape3D	23
Klassen Cuboid	25
Klassen Cylinder	25
Klassen Sphere	26
Klassdiagram över Program, ShapeType och Extensions	26
Klassen Program	27
Den uppräkningsbara typen ShapeType	30
A-krav	31
Läsvärt	31

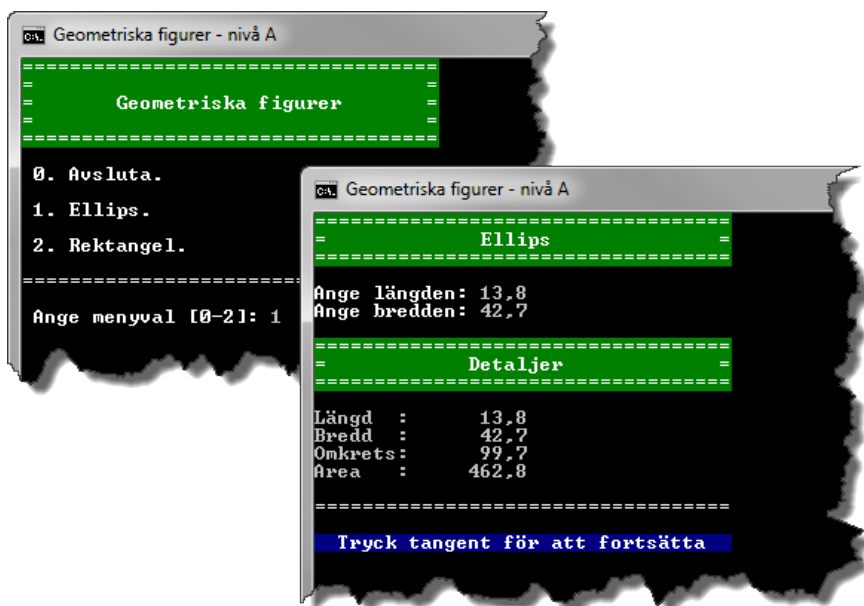
A. Uppgift

Problem

Geometriska figurer har flera egenskaper, som area och omkrets, som kan vara intressant att beräkna. Gemensamt för en ellips och rektangel är att de kan beskrivas med hjälp av längd och bredd. Med utgångspunkt från längden och bredden kan area och omkrets beräknas för respektive figur.


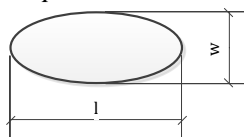
Du ska skriva en applikation som med hjälp av klasser grupperar beräkningarna av olika figurers area och omkrets. Beräkningar rörande en ellips area och omkrets ska placeras i klassen *Ellipse*. I klassen *Rectangle* ska beräkningar rörande en rektangels area och omkrets placeras.

Via en meny ska typ av figur väljas för vilken beräkning av area och omkrets ska göras. Efter att figur valts ska figurens längd och bredd anges varefter figurens detaljer ska presenteras.



Figur A.1.

Formelsamling för area och omkrets

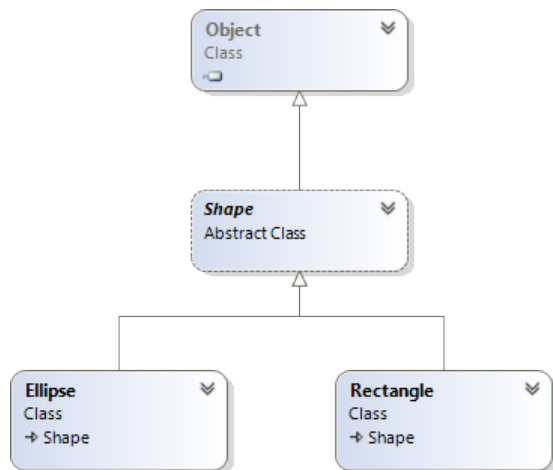
Figur	Formler och uttryck i C#
Rektangel 	$Area = l \cdot w$ <code>l * w</code> $Omkrets = 2l + 2w$ <code>2 * l + 2 * w</code>
Ellips 	$a = \frac{l}{2}$ $b = \frac{w}{2}$ $Area = \pi \cdot a \cdot b$ <code>Math.PI * a * b</code> $Omkrets = \pi \sqrt{2a^2 + 2b^2}$ <code>Math.PI * Math.Sqrt(2 * a * a + 2 * b * b)</code>

Formeln för en ellips omkrets är bara en approximation och är tillräckligt bra i detta sammanhang.

Klasshierarki

En ellips och rektangel har flera gemensamma egenskaper. De har båda en längd och bredd. De har även båda en area och en omkrets. Istället för att dessa egenskaper hanteras i respektive klass kan dessa placeras i en generell klass, en abstrakt basklass som det inte går att instansiera objekt av.

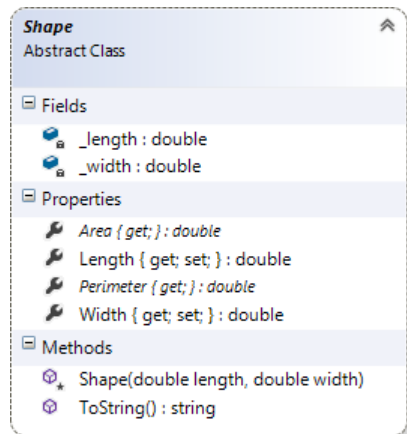
Den abstrakta klassen Shape ärver från Object och innehåller medlemmar som är gemensamma för de konkreta klasserna Ellipse och Rectangle, som ärver från klassen Shape.



Figur A.2.

Klassen Shape

Den abstrakta klassen Shape innehåller såväl konkreta som abstrakta medlemmar gemensamma för figurer som ellips och rektangel. I Figur A.3 presenteras de abstrakta medlemmarna med kursiv text.



Figur A.3. Egenskaperna Area och Perimeter är abstrakta. Konstruktorns åtkomstmodifierare är av typen protected.

Fältet *_length*

Privat fält av typen double representerande en figurs längd.

Fältet *_width*

Privat fält av typen double representerande en figurs bredd.

Egenskapen *Area*

Publik abstrakt egenskap av typen double representerande en figurs area.

Egenskapen *Length*

Publik egenskap av typen double som kapslar in fältet *_length*.

set-metoden ska validera värdet som tilldelas egenskapen. Är värdet inte större än 0 ska ett undantag av typen ArgumentException kastas.

Egenskapen *Perimeter*

Publik abstrakt egenskap av typen double representerande en figurs omkrets.

Egenskapen Width

Publik egenskap av typen `double` som kapslar in fältet `_width`.

`set`-metoden ska validera värdet som tilldelas egenskapen. Är värdet inte större än 0 ska ett undantag av typen `ArgumentException` kastas.

Konstruktorn

Konstruktorn, som ska vara *"protected"*, ansvara för att fälten, via egenskaperna, tilldelas de värden konstruktorns parametrar har.

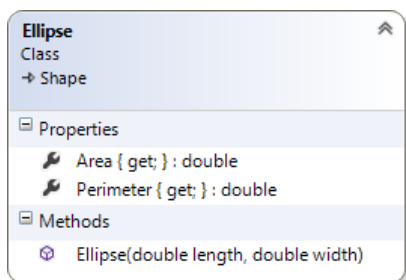
Metoden ToString

Publik metod som överskuggar metoden `ToString()` i basklassen `Object`. Metoden ska returnera en sträng representerande värdet av en instans. Strängen ska vara lite speciellt formaterad och på separata rader innehålla ledtext och värde för figurens längd, bredd, omkrets och area.

Längd : Length
Bredd : Width
Omkrets: Perimeter
Area : Area

Klassen Ellipse

Klassen `Ellipse` ärver från den abstrakta basklassen `Shape`. I och med att det ska gå att instansiera objekt av klassen, d.v.s. den ska vara konkret, måste den implementera de abstrakta egenskaperna `Area` och `Perimeter` i basklassen.



Figur A.4.

Egenskapen Area

Publik egenskapen av typen `double` som ska ge en ellips area.

Egenskapen Perimeter

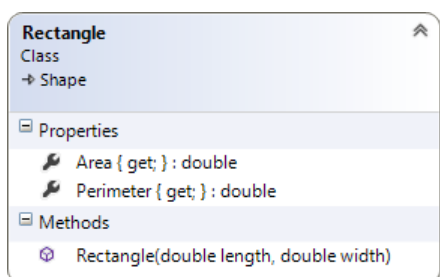
Publik egenskapen av typen `double` som ska ge en ellips omkrets.

Konstruktorn

Publik konstruktör som genom anrop av basklassens konstruktör ser till att det nya objektets längd och bredd sätts.

Klassen Rectangle

Klassen `Rectangle` ärver från den abstrakta basklassen `Shape`. I och med att det ska gå att instansiera objekt av klassen, d.v.s. den ska vara konkret, måste den implementera de abstrakta egenskaperna `Area` och `Perimeter` i basklassen.



Figur A.5.

Egenskapen Area

Publik egenskapen av typen `double` som ska ge en rektangels area.

Egenskapen Perimeter

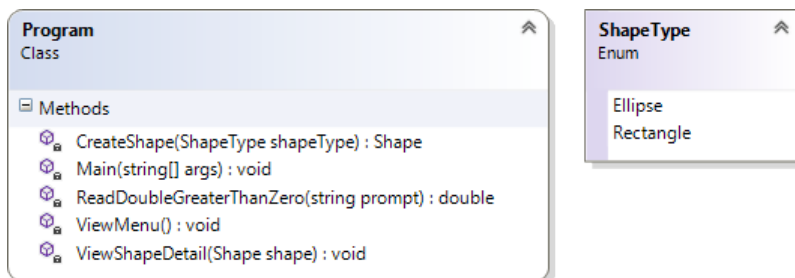
Publik egenskapen av typen `double` som ska ge en rektangels omkrets.

Konstruktorn

Publik konstruktor som genom anrop av basklassens konstruktor ser till att det nya objektets längd och bredd sätts.

Klassdiagram över Program och ShapeType

Klassen `Program` och den uppräkningsbara typen `ShapeType` ska användas för att skriva den menystyrda delen av applikationen där användaren väljer vilken figur för vilken längd och bredd ska matas in, area och omkrets beräknas samt figurens detaljer presenteras.

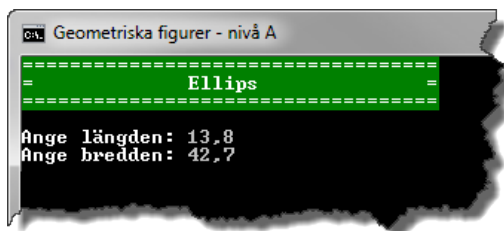


Figur A.6.

Klassen Program

Metoden CreateShape

Den privata statiska metoden `CreateShape` ska läsa in en figurs längd och bredd, skapa objektet och returnera en referens till det. Metoden ska ha en parameter av typen `ShapeType` vars värde bestämmer om en ellips eller rektangel ska skapas.

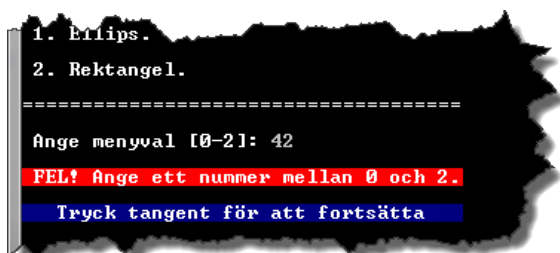


Figur A.7. Inläsning av en ellips längd och bredd.

Metoden Main

Metoden `Main` ska anropa metoden `ViewMenu()` för att visa en meny. Väljer användaren att inte avsluta applikationen ska med metoden `CreateShape()` anropas som skapar och returnerar en referens till ett `Ellipse`- eller `Rectangle`-objekt. Referensen till objektet används sedan vid anrop av `ViewDetail()` som presenterar figurens detaljer. Då en beräkning är gjord ska menyn visas på nytt.

Meny alternativen är numrerade från 0 till 2 och väljer inte användaren ett värde i det slutna intervallet mellan 0 och 2 ska ett felmeddelande visas och användaren uppmanas att trycka på en tangent för att få en ny chans att ange ett korrekt menyalternativ.

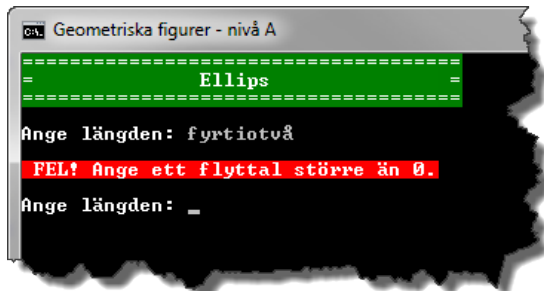


Figur A.8. Felmeddelande då användaren inte angett 0, 1 eller 2.

Metoden *ReadDoubleGreaterThanZero*

Den privata statiska metoden `ReadDoubleGreaterThanZero()` ska returnera ett värde av typen `double` som är större än 0. Till metoden ska det vara möjligt att skicka med ett argument. Argument ska vara en sträng med information som ska visas i anslutning till där inmatningen av värdet sker. I Figur A.9 har argumenten "Ange längden: " skickats med som argument vid anropet av metoden.

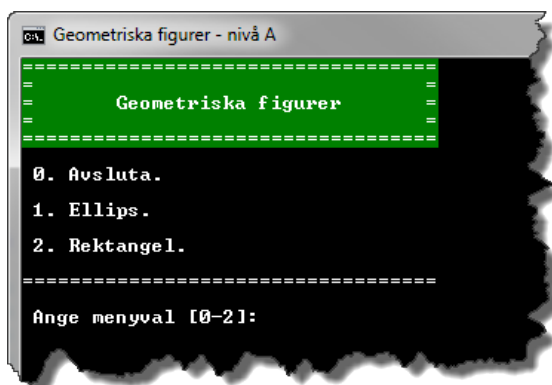
Om det inmatade inte kan tolkas som ett korrekt värde ska användaren få en chans att göra en ny inmatning efter att ett tydligt felmeddelande presenterats (se Figur A.9).



Figur A.9. Felmeddelande efter inmatning av sträng som inte kan tolkas som ett flyttal större än 0.

Metoden *ViewMenu*

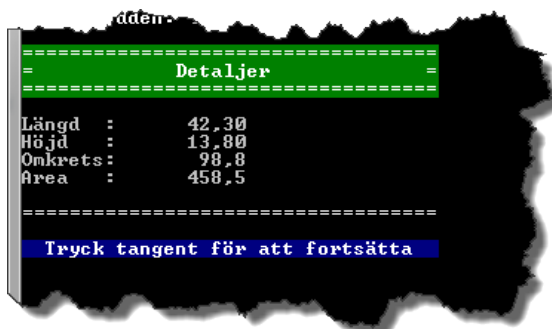
Den privata statiska metoden `ViewMenu()` ska bara presentera en meny. Någon inläsning ska inte göras av metoden.



Figur A.10. Applikationen efter att menyn presenterats.

Metoden *ViewShapeDetail*

Den privata statiska metoden `ViewShapeDetail()` ska presentera en figurs detaljer. Vid anrop av metoden skickas ett argument med som refererar till figurens vars detaljer ska presenteras. Parametern `shape` av typen `Shape` refererar till figuren. Genom att utnyttja att basklassen `Shape` överskuggar metoden `ToString()` förenklas koden väsentligt då en figurs längd, bredd, omkrets och area ska presenteras.

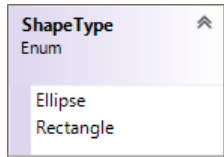


Figur A.11. Presentation av en figurs detaljer.

Den uppräkningsbara typen ShapeType

Den uppräkningsbara typen ShapeType används för att definiera vilka typer av figurer applikationen kan hantera. Typen används då metoden Main() anropar metoden CreateShape() för att informera vilken typ av figur som ska skapas.

Typen definieras lämpligen i filen Shape.cs, men då utanför klassdefinitionen så att den inte blir en del av typen Shape.



Figur A.12.

A-krav

1. Applikationen ska kunna beräkna och presentera omkrets och area för en ellips samt rektangel.
2. Klasserna Shape, Ellips och Rectangle ska implementeras enligt klassdiagrammen i Figur A.2 till Figur A.5.
3. Klasserna Ellipse och Rectangle ska implementera de abstrakta "read-only"-egenskaperna Area och Perimeter i basklassen Shape. Area och omkrets ska för respektive typ av figur beräknas enligt uttrycken under rubriken "Formelsamling för area och omkrets" på sidan 5.
4. Egenskaper som har en set-metod ska validera datat innan det tilldelas fältet egenskapen kapslar in.
5. Klassen Program och den uppräkningsbara typen ShapeType ska implementeras enligt klassdiagrammet i Figur A.6.
6. Metoden Main() i klassen Program ska se till att en meny visas där användaren kan välja för vilken typ av figur beräkningar av omkrets och area ska göras. Användaren ska via ett menyalternativ kunna välja att avsluta applikationen.
7. Metoden ViewShapeDetail() i klassen Program ska använda metoden ToString() i klassen Shape för att presentera en figurs längd, bredd, omkrets och area.
8. Fel som inträffar i applikationen ska tas om hand och relevanta felmeddelanden ska visas.

Läsvärt

- Klasser
 - Essential C# 5.0, 209-220.
 - <http://msdn.microsoft.com/en-us/library/0b0thckt.aspx>.
- Åtkomstmodifierare ("Access Modifiers")
 - Essential C# 5.0, 227-229.
 - <http://msdn.microsoft.com/en-us/library/ms173121.aspx>.
- Egenskaper
 - Essential C# 5.0, 229-235.
 - <http://msdn.microsoft.com/en-us/library/x9fsa0sw.aspx>.

- Konstruktörer
 - Essential C# 5.0, 244-248, 250-253.
 - <http://msdn.microsoft.com/en-us/library/k6sa6h87.aspx>.
- Arv
 - Essential C# 5.0, 277-311.
 - <http://msdn.microsoft.com/en-us/library/ms173149.aspx>.
- Uppräkningsbara typer (enum)
 - Essential C# 5.0, 358-366.
 - <http://msdn.microsoft.com/en-us/library/sbbt4032.aspx>.

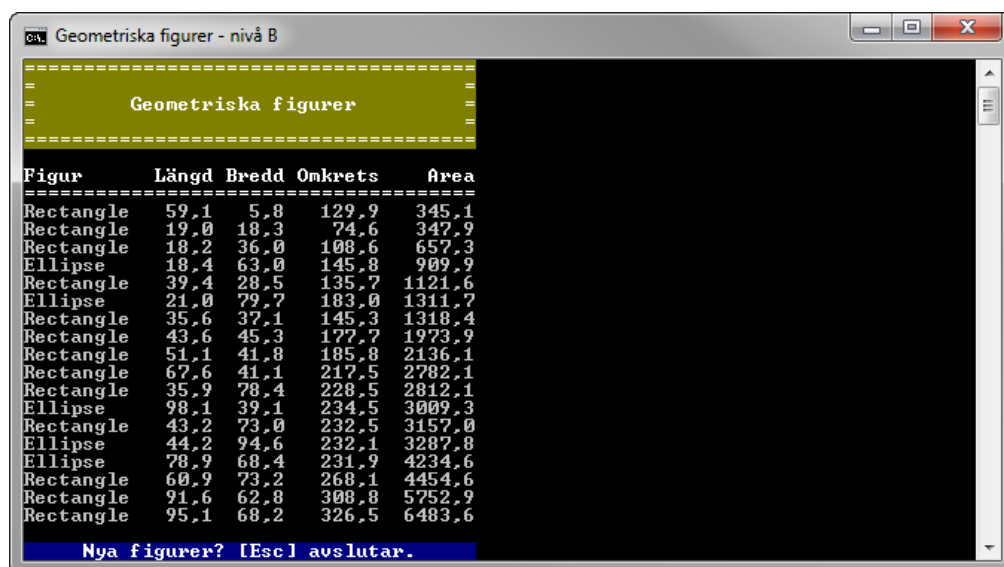
B. Uppgift

Problem

Geometrisk figur har flera egenskaper, som area och omkrets, som kan vara intressant att beräkna. Gemensamt för en ellips och rektangel är att de kan beskrivas med hjälp av längd och bredd. Med utgångspunkt från längden och bredden kan area och omkrets beräknas för respektive figur.

Du ska skriva en applikation som med hjälp av klasser grupperar beräkningarna av figurers area och omkrets. Beräkningar rörande en ellips area och omkrets ska placeras i klassen *Ellipse*. I klassen *Rectangle* ska beräkningar rörande en rektangels area och omkrets placeras.


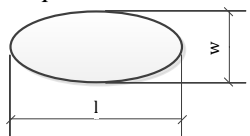
Applikationen ska inte kräva någon inmatning av något slag utan objekt av typerna *Ellipse* och *Rectangle* ska slumpas fram och referenser till objektet ska lagras i en array. Arrayen ska sedan sorteras med avseende på figurernas area varefter figurernas detaljer presenteras i form av en enkel tabell.



Figur	Längd	Bredd	Omkrets	Area
Rectangle	59.1	5.8	129.9	345.1
Rectangle	19.0	18.3	74.6	347.9
Rectangle	18.2	36.0	108.6	657.3
Ellipse	18.4	63.0	145.8	909.9
Rectangle	39.4	28.5	135.7	1121.6
Ellipse	21.0	79.7	183.0	1311.7
Rectangle	35.6	37.1	145.3	1310.4
Rectangle	43.6	45.3	177.7	1973.9
Rectangle	51.1	41.8	185.8	2136.1
Rectangle	67.6	41.1	217.5	2782.1
Rectangle	35.9	78.4	228.5	2812.1
Ellipse	98.1	39.1	234.5	3009.3
Rectangle	43.2	73.0	232.5	3157.0
Ellipse	44.2	94.6	232.1	3287.8
Ellipse	78.9	68.4	231.9	4234.6
Rectangle	60.9	73.2	260.1	4454.6
Rectangle	71.6	62.8	308.8	5752.9
Rectangle	95.1	68.2	326.5	6483.6

Figur B.1.

Formelsamling för area och omkrets

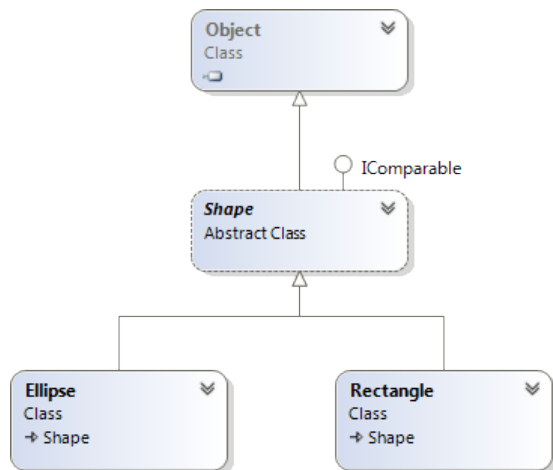
Figur	Formler och uttryck i C#
Rektangel 	$Area = l \cdot w$ <code>l * w</code> $Omkrets = 2l + 2w$ <code>2 * l + 2 * w</code>
Ellips 	$a = \frac{l}{2}$ $b = \frac{w}{2}$ $Area = \pi \cdot a \cdot b$ <code>Math.PI * a * b</code> $Omkrets = \pi \sqrt{2a^2 + 2b^2}$ <code>Math.PI * Math.Sqrt(2 * a * a + 2 * b * b)</code>

Formeln för en ellips omkrets är bara en approximation och är tillräckligt bra i detta sammanhang.

Klasshierarki

En ellips och rektangel har flera gemensamma egenskaper. De har båda en längd och bredd. De har även båda en area och en omkrets. Istället för att dessa egenskaper hanteras i respektive klass kan dessa placeras i en generell klass, en abstrakt basklass som det inte går att instansiera objekt av.

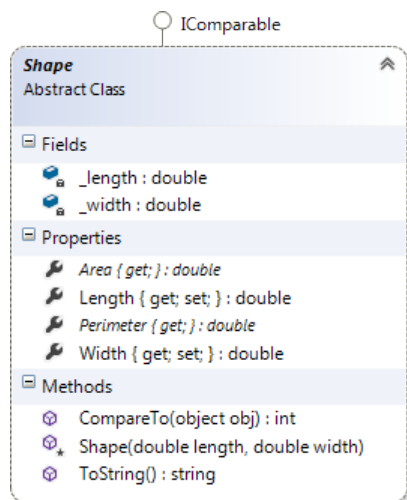
Den abstrakta klassen *Shape* ärver från *Object* och innehåller medlemmar som är gemensamma för de konkreta klasserna *Ellipse* och *Rectangle*, som ärver från klassen *Shape*.



Figur B.2.

Klassen *Shape*

Den abstrakta klassen *Shape* innehåller såväl konkreta som abstrakta medlemmar gemensamma för figurer som ellips och rektangel. I Figur B.3 presenteras de abstrakta medlemmarna med kursiv text. Klassen ska implementera interfacet *IComparable*, vilket innebär att klassen måste innehålla metoden *CompareTo()* som bland annat anropas av ramverket då en array med referenser av typen *Shape* ska sorteras.



Figur B.3. Egenskaperna *Area* och *Perimeter* är abstrakta. Konstruktorns åtkomstmodifierare är av typen *protected*.

Fältet *_length*

Privat fält av typen *double* representerande en figurs längd.

Fältet *_width*

Privat fält av typen *double* representerande en figurs bredd.

Egenskapen *Area*

Publik abstrakt egenskap av typen *double* representerande en figurs area.

Egenskapen *Length*

Publik egenskap av typen *double* som kapslar in fältet *_length*.

set-metoden ska validera värdet som tilldelas egenskapen. Är värdet inte större än 0 ska ett undantag av typen `ArgumentException` kastas.

Egenskapen Perimeter

Publik abstrakt egenskap av typen `double` representerande en figurs omkrets.

Egenskapen Width

Publik egenskap av typen `double` som kapslar in fältet `_width`.

set-metoden ska validera värdet som tilldelas egenskapen. Är värdet inte större än 0 ska ett undantag av typen `ArgumentException` kastas.

Konstruktorn

Konstruktorn, som ska vara *"protected"*, ansvara för att fälten, via egenskaperna, tilldelas de värden konstruktorns parametrar har.

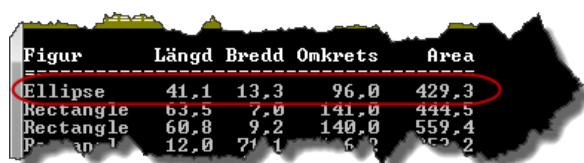
Metoden CompareTo

Metoden ska jämföra två objekt med avseende på deras areor.

- Refererar parametern till null ska ett heltal större än 0 returneras.
- Refererar parametern till ett objekt som inte är av typen Shape ska ett undantag av typen `ArgumentException` kastas.
- Refererar parametern till ett objekt vars area är större än det anropande objektet ska ett heltal mindre än 0 returneras.
- Refererar parametern till ett objekt vars area är mindre än det anropande objektet ska ett heltal större än 0 returneras.
- Refererar parametern till ett objekt vars area är lika med det anropande objektet ska heltalet 0 returneras.

Metoden ToString

Publik metod som överskuggar metoden `ToString()` i basklassen `Object`. Metoden ska returnera en sträng representerande värdet av en instans. Strängen ska vara lite speciellt formaterad och innehålla figurens typ och värden för längd, bredd, omkrets och area. Värdena ska vara högerjusterade med lämpliga fältbredder. Figurens typ ges av metoden `GetType()`, som ärvs från klassen `Object`, och egenskapen `Name`.

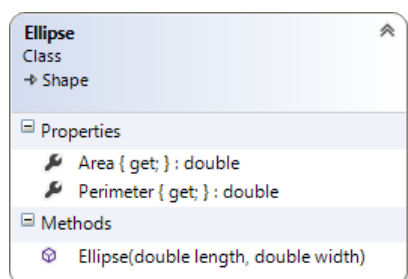


Figur	Längd	Bredd	Omkrets	Area
Ellipse	41.1	13.3	96.0	429.3
Rectangle	63.5	7.0	141.0	444.5
Rectangle	60.8	9.2	140.0	559.4
Rectangle	12.0	71.1	140.0	559.2

Figur B.4. Textbeskrivning av en figur.

Klassen Ellipse

Klassen `Ellipse` ärver från den abstrakta basklassen `Shape`. I och med att det ska gå att instansiera objekt av klassen, d.v.s. den ska vara konkret, måste den implementera de abstrakta egenskaperna `Area` och `Perimeter` i basklassen.



Figur B.5.

Egenskapen Area

Publik egenskapen av typen `double` som ska ge en ellips area.

Egenskapen Perimeter

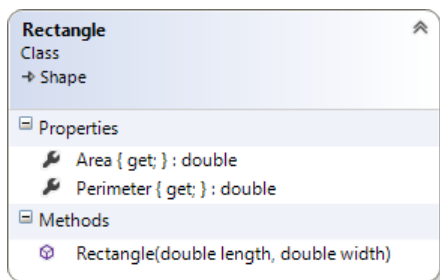
Publik egenskapen av typen `double` som ska ge en ellips omkrets.

Konstruktorn

Publik konstruktor som genom anrop av basklassens konstruktor ser till att det nya objektets längd och bredd sätts.

Klassen Rectangle

Klassen `Rectangle` ärver från den abstrakta basklassen `Shape`. I och med att det ska gå att instansiera objekt av klassen, d.v.s. den ska vara konkret, måste den implementera de abstrakta egenskaperna `Area` och `Perimeter` i basklassen.



Figur B.6.

Egenskapen Area

Publik egenskapen av typen `double` som ska ge en rektangels area.

Egenskapen Perimeter

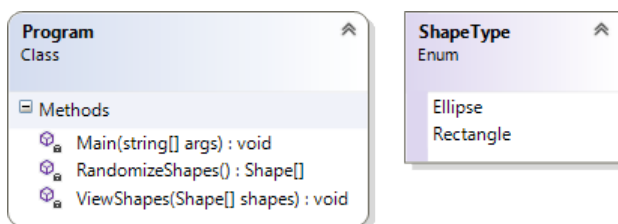
Publik egenskapen av typen `double` som ska ge en rektangels omkrets.

Konstruktorn

Publik konstruktor som genom anrop av basklassens konstruktor ser till att det nya objektets längd och bredd sätts.

Klassdiagram över Program och ShapeType

Klassen `Program` och den uppräkningsbara typen `ShapeType` ska användas för att skriva den del av applikationen som slumpar figurer och presenterar dem.



Figur B.7.

Klassen Program

Metoden Main

Metoden `Main` ska anropa metoden `RandomizeShape()` för att slumpa figurer. Figurerna ska därefter sorteras varefter metoden `ViewShapes()` anropas för att presentera figurerna.

Efter att figurerna presenterats ska användaren kunna välja att avsluta applikationen genom att trycka på Escape-tangenten; annan tangent ska slumpa och presentera nya figurer.

Metoden RandomizeShapes

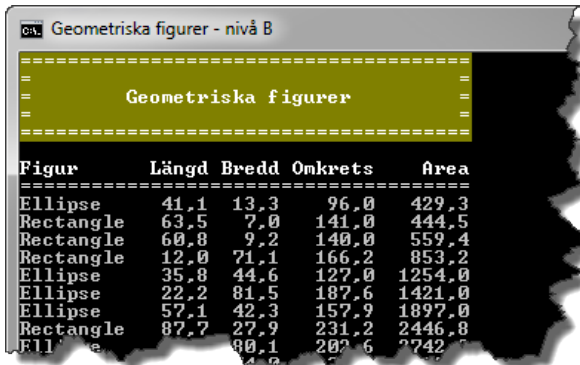
Den privata statistiska metoden `RandomizeShapes()` ska slumpa mellan 5 och 20 figurer vars längder och bredder slumpas till värden av typen `double` i det halvöppna intervallet mellan 5,0 och 100,0 (`[5, 100[`).

Typ av figur ska också slumpas och då måste en "switch"-sats användas tillsammans med uppräkningsbara typen `ShapeType`, som ska användas för att typomvandla heltalet 0 till `ShapeType.Ellipse` och heltalet 1 till `ShapeType.Rectangle`.

Referenserna till figurerna ska sparas i en array som metoden returnerar en referens till.

Metoden `ViewShapes`

Den privata statiska metoden `ViewShapes()` ska presentera figurerna som skickas till metoden i en enkel tabell. Metoden måste se till en figurs detaljer presenteras genom att använda textbeskrivningen för varje objekt, d.v.s. `ToString()` måste användas då ett objekts detaljer ska presenteras.



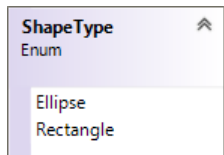
Figur	Längd	Bredd	Omkrets	Area
Ellipse	41.1	13.3	96.0	429.3
Rectangle	63.5	7.0	141.0	444.5
Rectangle	60.8	9.2	140.0	559.4
Rectangle	12.0	71.1	166.2	853.2
Ellipse	35.8	44.6	127.0	1254.0
Ellipse	22.2	81.5	187.6	1421.0
Ellipse	57.1	42.3	157.9	1897.0
Rectangle	87.7	27.9	231.2	2446.8
Ellipse	40.1	20.6	202.6	7742.0

Figur B.8. Tabell med figurers detaljer.

Den uppräkningsbara typen `ShapeType`

Den uppräkningsbara typen `ShapeType` används för att definiera vilka typer av figurer applikationen kan hantera. Typen används då metoden `RandomizeShapes()` slumpar vilken typ av figur som ska skapas.

Typen definieras lämpligen i filen `Shape.cs`, men då utanför klassdefinitionen så att den inte blir en del av typen `Shape`.



Figur B.9.

B-krav

1. Applikationen ska kunna beräkna och presentera omkrets och area för ellipser samt rektanglar.
2. Klasserna `Shape`, `Ellipse` och `Rectangle` ska implementeras enligt klassdiagrammen i Figur B.2, Figur B.3, Figur B.5 och Figur B.6.
3. Metoden `ToString()` i klassen `Shape` ska returnera en sträng där figurens namn är vänster justerat och längd, bredd, omkrets och area är högerjusterade.
4. Klasserna `Ellipse` och `Rectangle` ska implementera de abstrakta "read-only"-egenskaperna `Area` och `Perimeter` i basklassen `Shape`. Area och omkrets ska för respektive typ av figur beräknas enligt uttrycken under rubriken "Formelsamling för area och omkrets" på sidan 13.
5. Egenskaper som har en `set`-metod ska validera datat innan det tilldelas fältet egenskapen kapslar in. Klarar inte datat valideringen ska ett undantag av typen `ArgumentException` kastas.
6. Klassen `Program` och den uppräkningsbara typen `ShapeType` ska implementeras enligt klassdiagrammet i Figur B.7.

7. Metoden `Main()` i klassen `Program` ska se till att figurer av typerna `Ellipse` och `Rectangle` slumpas, sorteras och presenteras.
8. Metoden `RandomizeShapes()` ska slumpa mellan 5 och 20 objekt av typerna `Ellipse` och `Rectangle`. Figurernas längder och bredder ska slumpas till värden av typen `double` mellan inklusive 5,0 och exklusive 100,0.
9. Metoden `ViewShapes()` i klassen `Program` ska använda metoden `ToString()` i klassen `Shape` för att presentera en figurs läng, bredd, omkrets och area.
10. Fel som eventuellt inträffar i applikationen ska tas om hand och relevanta felmeddelanden ska visas.

Läsvärt

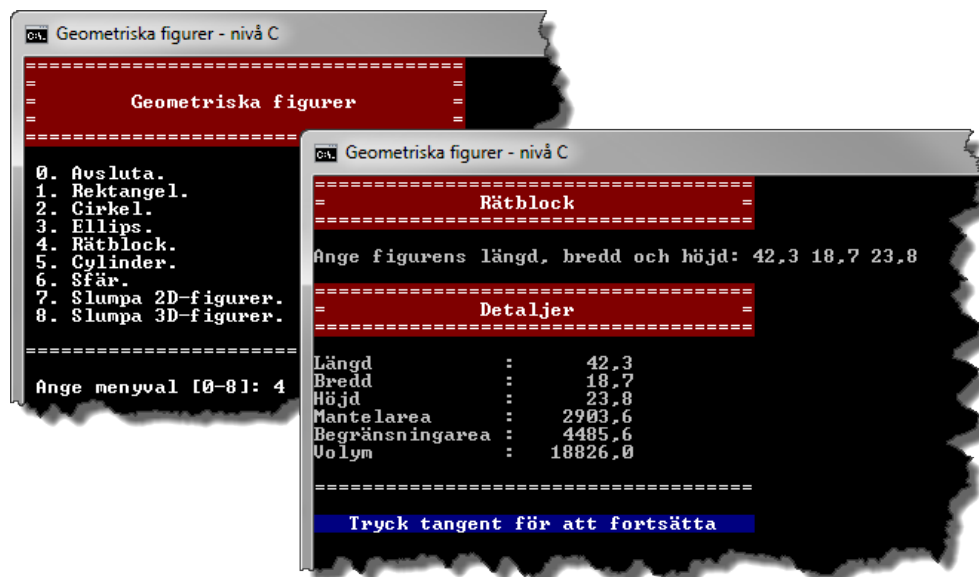
- Klasser
 - Essential C# 5.0, 209-220.
 - <http://msdn.microsoft.com/en-us/library/0b0thckt.aspx>.
- Åtkomstmodifierare ("*Access Modifiers*")
 - Essential C# 5.0, 227-229.
 - <http://msdn.microsoft.com/en-us/library/ms173121.aspx>.
- Egenskaper
 - Essential C# 5.0, 229-235.
 - <http://msdn.microsoft.com/en-us/library/x9fsa0sw.aspx>.
- Konstruktörer
 - Essential C# 5.0, 244-248, 250-253.
 - <http://msdn.microsoft.com/en-us/library/k6sa6h87.aspx>.
- Arv
 - Essential C# 5.0, 277-311.
 - <http://msdn.microsoft.com/en-us/library/ms173149.aspx>.
- `IComparable` (sortering av objekt)
 - Essential C# 5.0, 320-321.
 - <http://msdn.microsoft.com/en-us/library/ey2t2ys5.aspx>
- Uppräkningsbara typer (enum)
 - Essential C# 5.0, 358-366.
 - <http://msdn.microsoft.com/en-us/library/sbbt4032.aspx>.
- Klassen `Array`
 - <http://msdn.microsoft.com/en-us/library/czz5hkty.aspx>.
- Klassen `Random`
 - <http://msdn.microsoft.com/en-us/library/ts6se2ek.aspx>

C.Uppgift

Problem

Du ska skriva en applikation som med hjälp av klasser grupperar beräkningarna för plangeometrisk (2D) figurers area och omkrets. För rymdgeometrisk (3D) figurer ska volym, mantelarea och begränsningsarea beräknas.

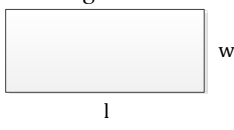
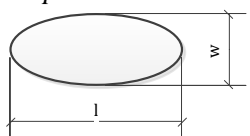
Via en meny ska typ av figur väljas för vilken beräkningar ska göras. Efter att figur valts ska figurens längd och bredd, och i förekommande fall djup, anges varefter figurens detaljer ska presenteras.

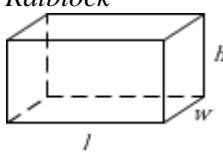
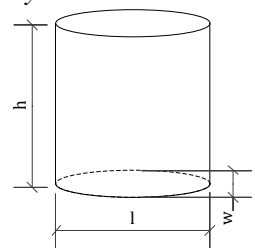
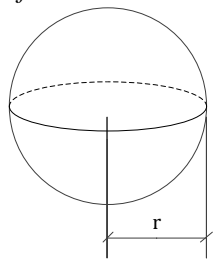


Figur C.1.

Användaren ska också kunna välja att slumpa 2D- eller 3D-figurer där referenser till objektet ska lagras i en array. Arrayen ska sedan sorteras varefter figurernas detaljer presenteras i form av en enkel tabell. 2D-figurer ska sorteras med avseende på area, 3D-figurer med avseende på volym.

Formelsamling

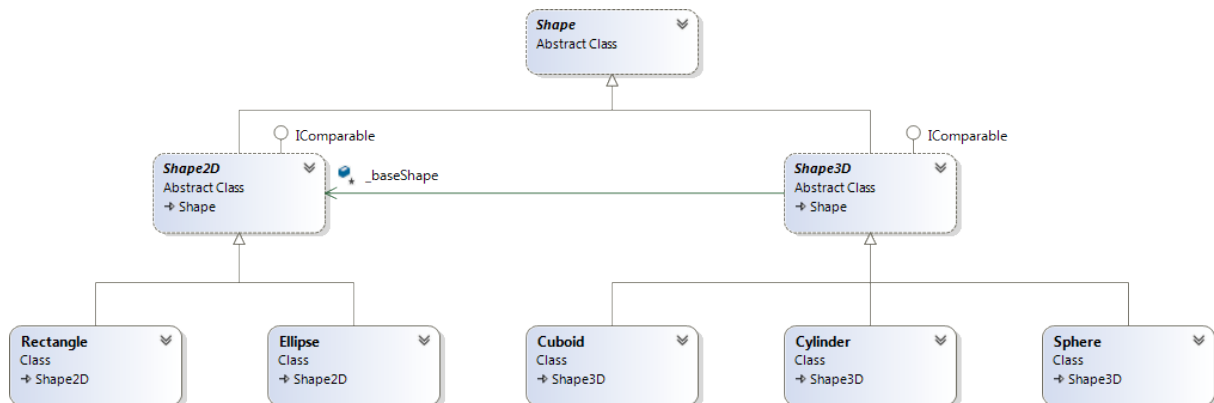
Figur	Formler och uttryck i C#
Rektangel 	$Area = l \cdot w$ <code>l * w</code> $Omkrets = 2l + 2w$ <code>2 * l + 2 * w</code>
Ellips 	$a = \frac{l}{2}$ $b = \frac{w}{2}$ $Area = \pi \cdot a \cdot b$ <code>Math.PI * a * b</code> $Omkrets = \pi \sqrt{2a^2 + 2b^2}$ <code>Math.PI * Math.Sqrt(2 * a * a + 2 * b * b)</code> (Formeln för en ellips omkrets är bara en approximation och är tillräckligt bra i detta sammanhang.)

<p><i>Rätblock</i></p> 	<p>Basfigur: rektangel</p> $\text{Mantelarea} = \text{basomkrets} \cdot h$ $\text{basomkrets} \cdot h$ $\text{Begränsningsarea} = \text{mantelarea} + 2 \cdot \text{basarea}$ $\text{mantelarea} + 2 \cdot \text{basarea}$ $\text{Volym} = \text{basarea} \cdot h$ $\text{basarea} \cdot h$
<p><i>Cylinder</i></p> 	<p>Basfigur: ellips</p> $\text{Mantelarea} = \text{basomkrets} \cdot h$ $\text{basomkrets} \cdot h$ $\text{Begränsningsarea} = \text{mantelarea} + 2 \cdot \text{basarea}$ $\text{mantelarea} + 2 \cdot \text{basarea}$ $\text{Volym} = \text{basarea} \cdot h$ $\text{basarea} \cdot h$
<p><i>Sfär</i></p> 	<p>Basfigur: ellips</p> $\text{Mantelarea} = \text{basarea} \cdot 4$ $\text{basarea} \cdot 4$ $\text{Begränsningsarea} = \text{basarea} \cdot 4$ $\text{basarea} \cdot 4$ $\text{Volym} = \frac{4}{3} \cdot \text{basarea} \cdot r$ $4 / 3 \cdot \text{basarea} \cdot r$

Klasshierarki

Figureerna har flera gemensamma egenskaper. 2D-figurer har t.ex. en längd, bredd, area och en omkrets. Istället för att dessa egenskaper hanteras i respektive klass kan dessa placeras i en generell klass, en abstrakt basklass som det inte går att instansiera objekt av.

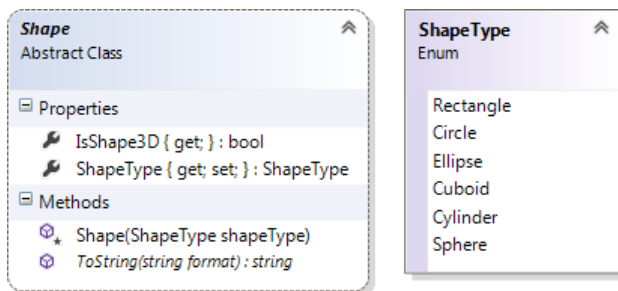
Den abstrakta klassen Shape2D ärver från Shape och innehåller medlemmar som är gemensamma för de konkreta klasserna Ellipse och Rectangle. Klassen Shape3D innehåller medlemmar som är gemensamma för de konkreta klasserna Cuboid, Cylinder och Sphere, som var och en baseras på en 2D-figur varför det är ett aggregat mellan Shape3D och Shape2D. Den abstrakta basklassen Shape innehåller medlemmar gemensamma för klasserna Shape2D och Shape3D.



Figur C.2.

Klassen Shape

Den abstrakta klassen Shape innehåller medlemmar gemensamma för klasserna Shape2D och Shape3D.



Figur C.3.

Egenskapen IsShape3D

Publik "read-only"-egenskap som ska returnera true om figuren är av någon av typerna Cuboid, Cylinder eller Sphere, annars false.

Egenskapen ShapeType

Autoimplementerad egenskap av typen ShapeType där get-metoden är publik och set-metoden är privat. Används för att definiera vilken typ av figur det är. ShapeType är en uppräkningsbar typ (enum) som lämpligen definieras i filen Shape.cs.

Konstruktorn

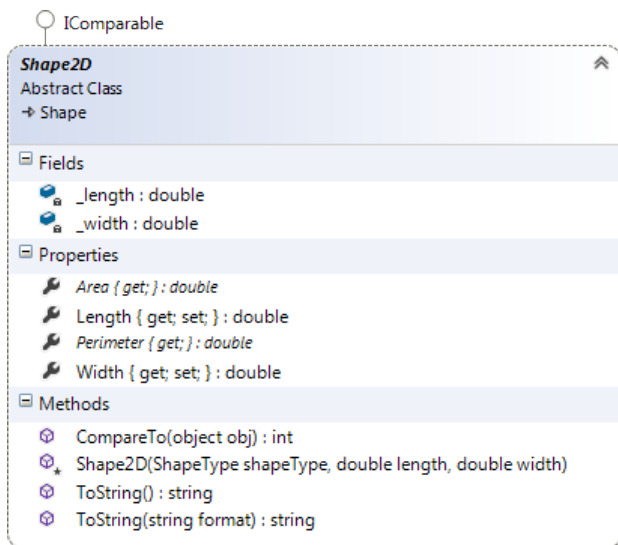
Konstruktorn, som ska vara "protected", ansvara för att objektet initieras med det värde konstruktorns parameter har.

Metoden ToString

Publik abstrakt metod som ska returnera en formaterad textbeskrivning av objektet.

Klassen Shape2D

Den abstrakta klassen Shape2D innehåller såväl konkreta som abstrakta medlemmar gemensamma för figurer som ellips och rektangel. I Figur C.4 presenteras de abstrakta medlemmarna med kursiv text.



Figur C.4. Egenskaperna Area och Perimeter är abstrakta. Konstruktorns åtkomstmodifierare är protected.

Fältet _length

Privat fält av typen double representerande en figurs längd.

Fältet _width

Privat fält av typen double representerande en figurs bredd.

Egenskapen Area

Publik abstrakt egenskap av typen `double` representerande en figurs area.

Egenskapen Length

Publik egenskap av typen `double` som kapslar in fältet `_length`.

`set`-metoden ska validera värdet som tilldelas egenskapen. Är värdet inte större än 0 ska ett undantag av typen `ArgumentException` kastas.

Egenskapen Perimeter

Publik abstrakt egenskap av typen `double` representerande en figurs omkrets.

Egenskapen Width

Publik egenskap av typen `double` som kapslar in fältet `_width`.

`set`-metoden ska validera värdet som tilldelas egenskapen. Är värdet inte större än 0 ska ett undantag av typen `ArgumentException` kastas.

Metoden CompareTo

Metoden ska jämföra två objekt med avseende på deras areor.

- Refererar parametern till `null` ska ett heltal större än 0 returneras.
- Refererar parametern till ett objekt som inte är av typen `Shape2D` ska ett undantag av typen `ArgumentException` kastas.
- Refererar parametern till ett objekt vars area är större än det anropande objektet ska ett heltal mindre än 0 returneras.
- Refererar parametern till ett objekt vars area är mindre än det anropande objektet ska ett heltal större än 0 returneras.
- Refererar parametern till ett objekt vars area är lika med det anropande objektet ska heltalet 0 returneras.

Konstruktorn

Konstruktorn, som ska vara *"protected"*, ansvara för att fälten, via egenskaperna, tilldelas de värden konstruktorns parametrar har.

Metoderna ToString

`ToString()` ska överlagras, d.v.s. det ska finnas två metoder med samma namn men med olika parameterlistor.

Den publika metoden `ToString(string format)` har som uppgift att returnera en sträng representerande värdet av en instans. Formatsträngen ska bestämma hur textbeskrivningen av instansen ska formateras.

Är formatsträngen "G", en tomsträng eller null, ska strängen formateras så separata rader innehåller ledtext och värden för figurens längd, bredd, omkrets och area.

```
Längd : 5.7
Bredd : 34.5
Omkrets: 77.7
Area : 154.4
```

Figur C.5.

Är formatsträngen "R" ska strängen formateras så en rad innehåller ledtext och värden för figurens längd, bredd, omkrets och area.

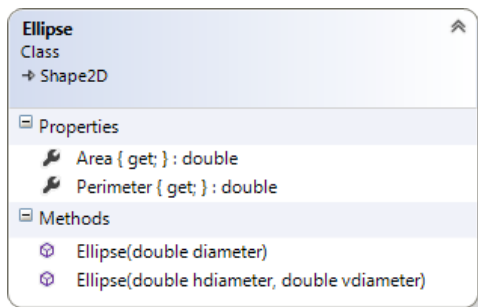
```
Figure    Längd  Bredd  Omkrets  Area
Ellipse   5.7    34.5    77.7    154.4
Rectangle 31.1    17.8    77.8    553.6
Rectangle 72.5     8.0   161.0   580.0
Rectangle 20.1    57.7    99.8    9.8
```

Figur C.6.

Alla övriga värden på formatsträngen ska leda till att ett undantag av typen `FormatException` kastas. Metoden `ToString()` ska returnera en sträng formaterad enligt kraven för formatsträngen "G".

Klassen `Ellipse`

Klassen `Ellipse` ärver från den abstrakta basklassen `Shape2D`. I och med att det ska gå att instansiera objekt av klassen, d.v.s. den ska vara konkret, måste den implementera de abstrakta egenskaperna `Area` och `Perimeter` i basklassen.



Figur C.7.

Egenskapen `Area`

Publik egenskapen av typen `double` som ska ge en ellips area.

Egenskapen `Perimeter`

Publik egenskapen av typen `double` som ska ge en ellips omkrets.

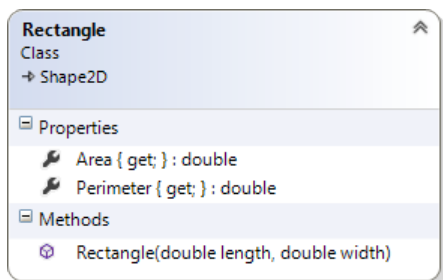
Konstruktorn

Publika konstruktorer som genom anrop av basklassens konstruktör ser till att det nya objektets längd och bredd sätts.

Konstruktorn som har en parameter används då en figur av typen cirkel ska skapas.

Klassen `Rectangle`

Klassen `Rectangle` ärver från den abstrakta basklassen `Shape2D`. I och med att det ska gå att instansiera objekt av klassen, d.v.s. den ska vara konkret, måste den implementera de abstrakta egenskaperna `Area` och `Perimeter` i basklassen.



Figur C.8.

Egenskapen `Area`

Publik egenskapen av typen `double` som ska ge en rektangels area.

Egenskapen `Perimeter`

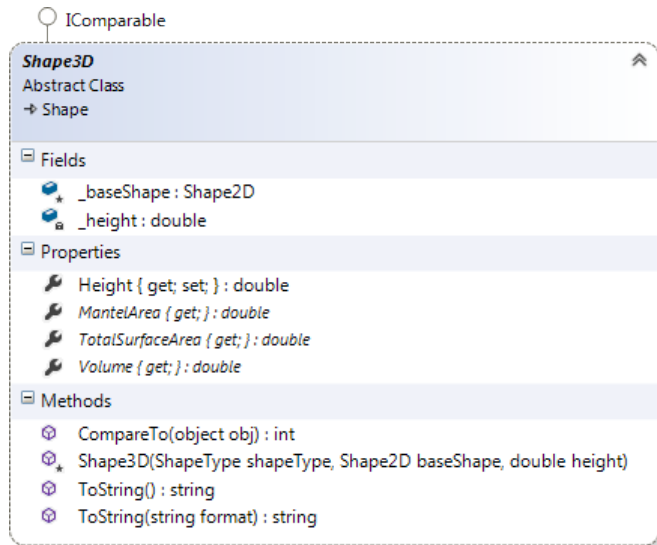
Publik egenskapen av typen `double` som ska ge en rektangels omkrets.

Konstruktorn

Publik konstruktör som genom anrop av basklassens konstruktör ser till att det nya objektets längd och bredd sätts.

Klassen `Skape3D`

Den abstrakta klassen `Shape3D` innehåller såväl konkreta som abstrakta medlemmar gemensamma för figurer som rätblock, cylinder och sfär. I Figur C.9 presenteras de abstrakta medlemmarna med kursiv text.



Figur C.9. Egenskaperna MantelArea, TotalSurfaceArea och Volume är abstrakta. Konstruktorns åtkomstmodifierare är av typen protected.

Fältet *_baseShape*

Skyddat ("protected") fält av typen Shape2D representerande en 3D-figurs basfigur (ellips eller rektangel).

Fältet *_height*

Privat fält av typen double representerande en figurs höjd.

Egenskapen *Height*

Publik egenskap av typen double som kapslar in fältet *_height*.

set-metoden ska validera värdet som tilldelas egenskapen. Är värdet inte större än 0 ska ett undantag av typen ArgumentException kastas.

Egenskapen *MantelArea*

Publik abstrakt egenskap av typen double representerande en figurs mantelarea.

Egenskapen *TotalSurfaceArea*

Publik abstrakt egenskap av typen double representerande en figurs begränsningsarea.

Egenskapen *Volume*

Publik abstrakt egenskap av typen double representerande en figurs volym.

Metoden *CompareTo*

Metoden ska jämföra två objekt med avseende på deras volymer.

- Refererar parametern till null ska ett heltal större än 0 returneras.
- Refererar parametern till ett objekt som inte är av typen Shape3D ska ett undantag av typen ArgumentException kastas.
- Refererar parametern till ett objekt vars volym är större än det anropande objektet ska ett heltal mindre än 0 returneras.
- Refererar parametern till ett objekt vars volym är mindre än det anropande objektet ska ett heltal större än 0 returneras.

Konstruktorn

Konstruktorn, som ska vara "protected", ansvara för att fälten, via egenskaper då sådana finns, tilldelas de värden konstruktorns parametrar har.

Metoderna *ToString*

ToString() ska överlagras, d.v.s. det ska finnas två metoder med samma namn men med olika parameterlistor.

Den publika metoden `ToString(string format)` har som uppgift att returnera en sträng representerande värdet av en instans. Formatsträngen ska bestämma hur textbeskrivningen av instansen ska formateras.

Är formatsträngen "G", en tomsträng eller null, ska strängen formateras så separata rader innehåller ledtext och värden för figures läng, bredd, höjd, mantelarea, begränsningsarea och volym.

```
Längd      :      29,6
Bredd      :      29,6
Höjd       :      29,6
Mantelarea :    2752,5
Begränsningsarea : 2752,5
Volym      :   13579,2
```

Figur C.10

Är formatsträngen "R" ska strängen formateras så en rad innehåller ledtext och värden för figures läng, bredd, höjd, mantelarea, begränsningsarea och volym.

	Längd	Bredd	Höjd	Mantelarea	Begränsningsarea	Volym
Cylinder	96,9	5,4	9,5	2049,1	2070,1	3904,2
Sphere	29,6	29,6	29,6	2752,5	2752,5	13579,2
Cuboid	12,0	41,4	50,6	5404,1	5397,7	25136,1
Cylinder	2	63,2	9,5	904,1	897,9	5235,1

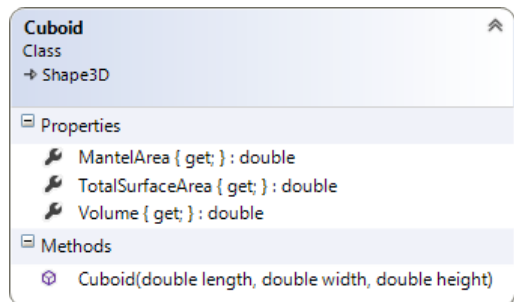
Figur C.11.

Alla övriga värden på formatsträngen ska leda till att ett undantag av typen `FormatException` kastas.

Metoden `ToString()` ska returnera en sträng formaterad enligt kraven för formatsträngen "G".

Klassen Cuboid

Klassen `Cuboid` ärver från den abstrakta basklassen `Shape3D`. I och med att det ska gå att instansiera objekt av klassen, d.v.s. den ska vara konkret, måste den implementera de abstrakta egenskaperna `MantelArea`, `TotalSurfaceArea` och `Volume` i basklassen.



Figur C.12.

Egenskapen `MantelArea`

Publik egenskapen av typen `double` som ska ge ett rätblocks mantelarea.

Egenskapen `TotalSurfaceArea`

Publik egenskapen av typen `double` som ska ge ett rätblocks begränsningsarea.

Egenskapen `Volume`

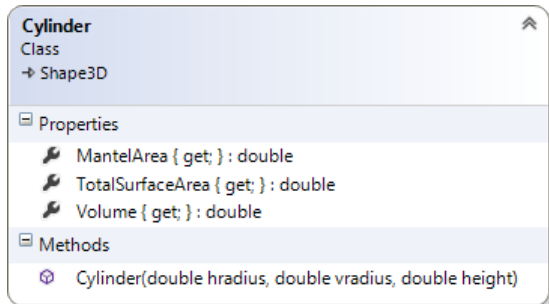
Publik egenskapen av typen `double` som ska ge ett rätblocks volym.

Konstruktorn

Publik konstruktör som genom anrop av basklassens konstruktör ser till att det nya objektets längd, bredd och höjd sätts.

Klassen `Cylinder`

Klassen `Cylinder` ärver från den abstrakta basklassen `Shape3D`. I och med att det ska gå att instansiera objekt av klassen, d.v.s. den ska vara konkret, måste den implementera de abstrakta egenskaperna `MantelArea`, `TotalSurfaceArea` och `Volume` i basklassen.



Figur C.13.

Egenskapen MantelArea

Publik egenskapen av typen `double` som ska ge en cylinders mantelarea.

Egenskapen TotalSurfaceArea

Publik egenskapen av typen `double` som ska ge en cylinders begränsningsarea.

Egenskapen Volume

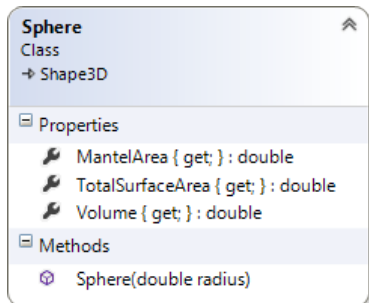
Publik egenskapen av typen `double` som ska ge en cylinders volym.

Konstruktorn

Publik konstruktör som genom anrop av basklassens konstruktör ser till att det nya objektets vertikala radie, vertikala bredd och höjd sätts.

Klassen Sphere

Klassen **Sphere** ärver från den abstrakta basklassen **Shape3D**. I och med att det ska gå att instansiera objekt av klassen, d.v.s. den ska vara konkret, måste den implementera de abstrakta egenskaperna **MantelArea**, **TotalSurfaceArea** och **Volume** i basklassen.



Figur C.14.

Egenskapen MantelArea

Publik egenskapen av typen `double` som ska ge en sfärs mantelarea.

Egenskapen TotalSurfaceArea

Publik egenskapen av typen `double` som ska ge en sfärs begränsningsarea.

Egenskapen Volume

Publik egenskapen av typen `double` som ska ge en sfärs volym.

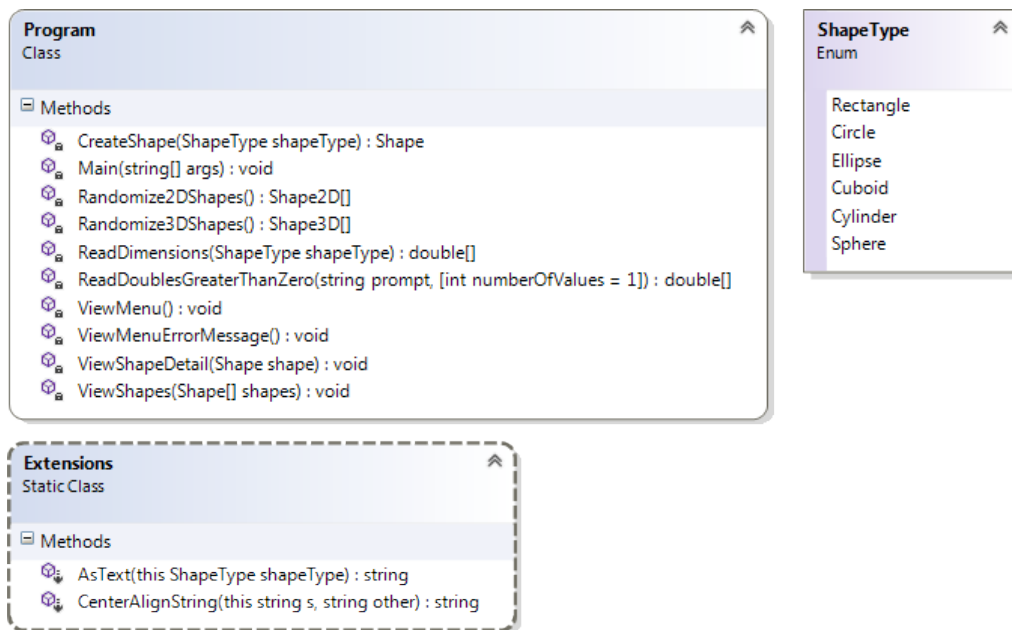
Konstruktorn

Publik konstruktör som genom anrop av basklassens konstruktör ser till att det nya objektets radie sätts.

Klassdiagram över Program, ShapeType och Extensions

Klassen **Program** och den uppräkningsbara typen **ShapeType** ska användas för att skriva den menystyrda delen av applikationen där användaren väljer vilken figur för vilken dimensioner ska matas in och olika värden beräknas samt figurens detaljer presenteras.

Den statiska klassen **Extensions** innehåller utökningsmetoder som bryter ut viss funktionalitet från klassen **Program**.



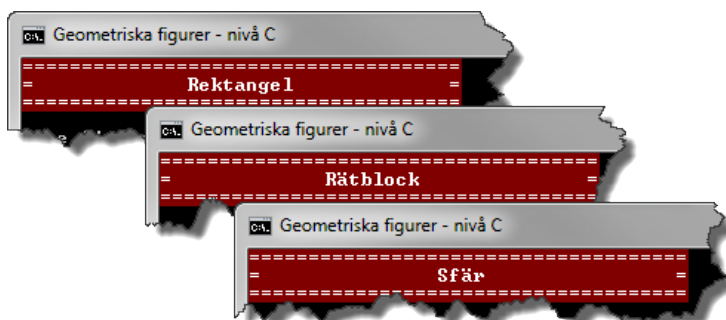
Figur C.15.

Klassen Program

Metoden CreateShape

Den privata statiska metoden CreateShape ska läsa in en figurs dimensioner, skapa objektet och returnera en referens till det. Metoden ska ha en parameter av typen ShapeType vars värde bestämmer vilken typ av figur som ska skapas.

Metoden ansvarar för att en rubrik skrivs ut där det framgår vilken typ av figur dimensionerna ska anges för.



Figur C.16. Rubriken ändras efter typen av figur dimensionerna ska läsas in för.

Att "översätta" ett enum-värde till text kan inte göras genom att överskugga ToString() för den uppräkningsbara typen ShapeType. Det är dock fullt möjligt att skapa en utökningsmetod, Extensions.AsText(), som gör detta.

Strängen med figurens typ ska centreras rubriken. Det finns ingen metod i klassen String där en sträng kan centreras i förhållande till en annan sträng. Elegantast löses även detta med hjälp av en utökningsmetod, Extensions.CenterAlignString().

CreateShape läser inte några värden utan det ska ske genom anrop av metoden ReadDimensions() som returnerar en array med värden av typen double där antalet värden i array beror på typen av figur.

Metoden Main

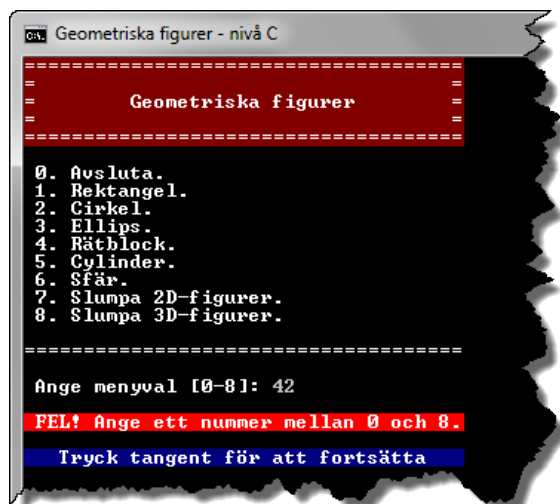
Metoden Main ska anropa metoden ViewMenu() för att visa en meny. Så länge som användaren inte väljer att avsluta applikationen ska menyn visas på nytt efter att en figurs, eller flera figurers, detaljer presenterats.

Väljer användaren att skapa en figur ska metoden CreateShape() anropas som skapar och returnerar en referens till ett objekt som symboliserar den figur som valts. Referensen till objektet används sedan

vid anrop av `ViewShapeDetail()` som presenterar figurens detaljer.

Användaren ska även kunna välja att slumpa 2D-figurer och 3D-figurer. Då ska metoden `Randomize2DShapes()` respektive `Randomize3DShapes()` anropas. Arrayen med referenser som metoderna returnerar ska sorteras och skickas med som argument vid anrop av metoden `ViewShapes()` som presenterar figurernas detaljer i form av en enkel tabell.

Menyalternativen är numrerade från 0 till 8 och väljer inte användaren ett värde i det slutna intervallet mellan 0 och 8 ska ett felmeddelande visas och användaren uppmanas att trycka på en tangent för att få en ny chans att ange ett korrekt menyalternativ.



Figur C.17. Felmeddelande då användaren inte angett ett val i det slutna intervallet mellan 0 och 8.

Metoden `Randomize2DShapes`

Den privata statistiska metoden `Randomize2DShapes()` ska slumpa mellan 5 och 20 figurer vars längder och bredder slumpas till värden av typen `double` i det halvöppna intervallet mellan 5,0 och 100,0 ([5, 100[).

Typ av figur ska också slumpas och då måste en "switch"-sats användas tillsammans med uppräkningsbara typen `ShapeType`, som ska användas för att typomvandla heltalet 0 till `ShapeType.Ellips`, heltalet 1 till `ShapeType.Circle` och heltalet 2 till `ShapeType.Rectangle`.

Referenserna till figurerna ska sparas i en array som metoden returnerar en referens till.

Metoden `Randomize3DShapes`

Den privata statistiska metoden `Randomize3DShapes()` ska slumpa mellan 5 och 20 figurer vars dimensioner slumpas till värden av typen `double` i det halvöppna intervallet mellan 5,0 och 100,0 ([5, 100[).

Typ av figur ska också slumpas och då måste en "switch"-sats användas tillsammans med uppräkningsbara typen `ShapeType`, som ska användas för att typomvandla heltalet 3 till `ShapeType.Cuboid`, heltalet 4 till `ShapeType.Cylinder` och heltalet 5 till `ShapeType.Sphere`.

Referenserna till figurerna ska sparas i en array som metoden returnerar en referens till.

Metoden `ReadDimensions`

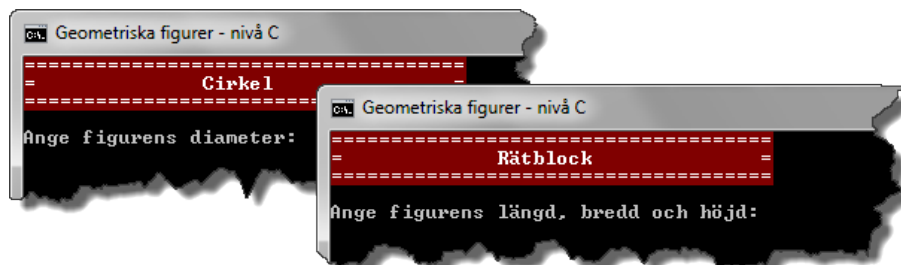
Den privata statistiska metoden `ReadDimensions()` ska returnera en array där elementens värden är av typen `double` och större än 0.

Till metoden ska det vara möjligt att skicka med ett argument av typen `ShapeType` som bestämmer argumenten, ledtext och antal värden, som används vid anrop av metoden `ReadDoublesGreaterThanZero()` som sköter själva inläsningen av figurens dimensioner.

Metoden `ReadDoublesGreaterThanZero`

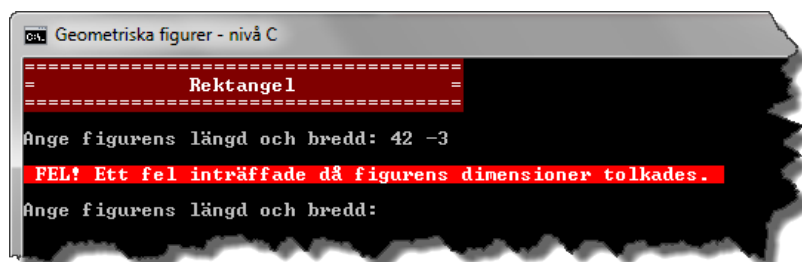
Den privata statistiska metoden `ReadDoublesGreaterThanZero()` ska returnera en array där elementens värden är av typen `double` och större än 0.

Till metoden ska det vara möjligt att skicka med två argument. Det första argument ska vara en sträng med information som ska visas i anslutning till där inmatningen av värdet sker. Det andra argumentet ska ange hur många värden som ska anges. I Figur C.18 har olika strängar, och antalet förväntade värden, skickats med som argument vid anropet av metoden.



Figur C.18. Beroende på typ av figur är antalet värden användaren ska ange olika.

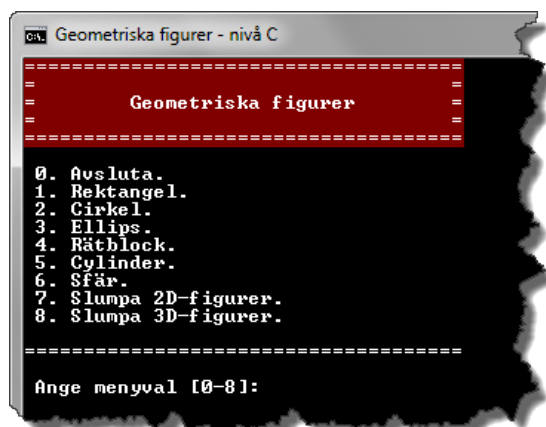
Samtliga värden ska användaren kunna ange på en och samma rad. Den inlästa strängen ska sedan delas upp i delsträngar där varje delsträng ska tolkas till ett värde av typen `double`. Om det inmatade inte kan tolkas som ett korrekt värde, eller om antalet angivna värden inte överensstämmer med det andra argumentets värde, ska användaren få en chans att göra en ny inmatning efter att ett tydligt felmeddelande presenterats (se Figur C.19).



Figur C.19. Felmeddelande efter inmatning av sträng som inte kan tolkas som figurens dimensioner.

Metoden `ViewMenu`

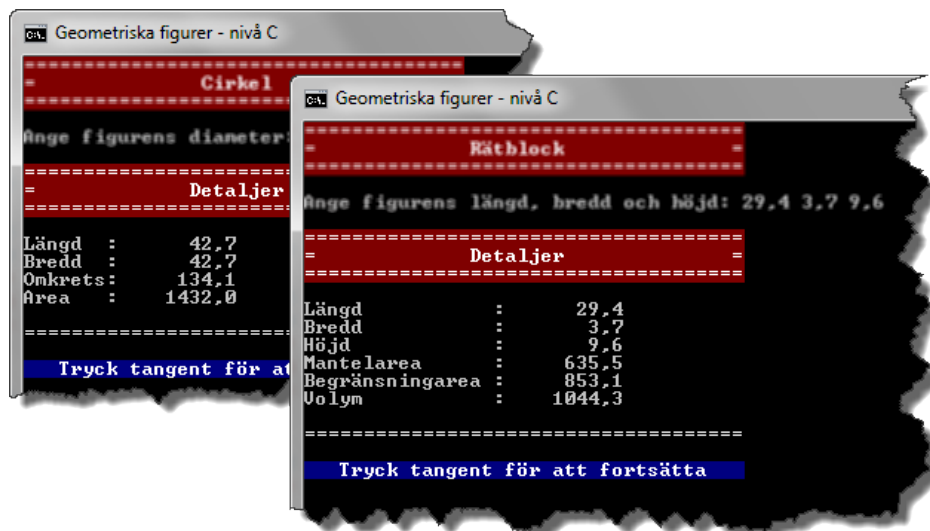
Den privata statiska metoden `ViewMenu()` ska bara presentera en meny. Någon inläsning ska inte göras av metoden.



Figur C.20. Applikationen efter att menyn presenterats.

Metoden `ViewShapeDetail`

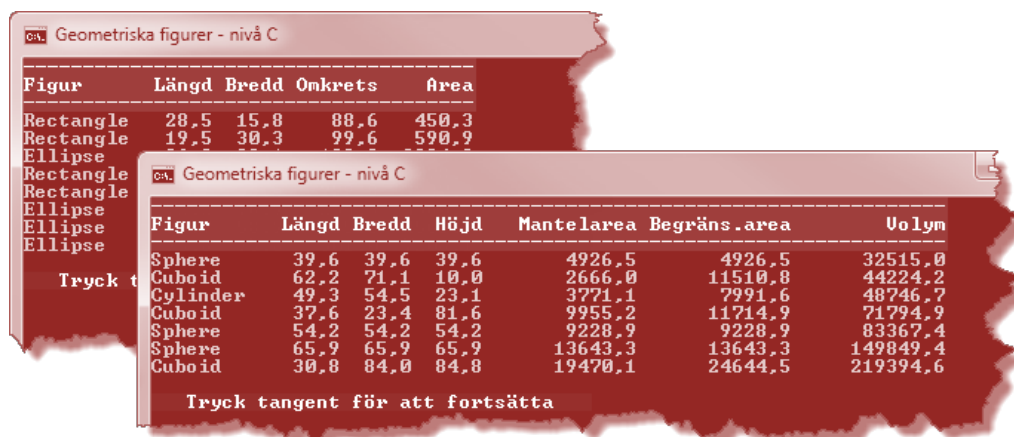
Den privata statiska metoden `ViewShapeDetail()` ska presentera en figurs detaljer. Vid anrop av metoden skickas ett argument med som refererar till figurens vars detaljer ska presenteras. Parametern `shape` av typen `Shape` refererar till figuren. Genom att utnyttja att basklasserna `Shape2D` och `Shape3D` överskuggar metoden `ToString()` förenklas koden väsentligt då en figurs detaljer ska presenteras.



Figur C.21. Presentation av en 2D- och en 3D-figurs detaljer.

Metoden ViewShapes

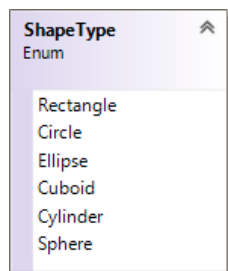
Den privata statiska metoden `ViewShapes()` ska presentera figurerna som skickas till metoden i en enkel tabell. Metoden måste se till en figurs detaljer presenteras genom att använda en formaterad textbeskrivning för varje objekt, d.v.s. `ToString("R")` måste användas då ett objekts detaljer ska presenteras.



Figur C.22. Tabeller med 2D- och 3D-figurers detaljer.

Den uppräkningsbara typen ShapeType

Den uppräkningsbara typen `ShapeType` används för att definiera vilka typer av figurer applikationen kan hantera. Typen används då metoden `Main()` anropar metoden `CreateShape()` för att informera vilken typ av figur som ska skapas. Typen definieras lämpligen i filen `Shape.cs`, men då utanför klassdefinitionen så att den inte blir en del av typen `Shape`.



Figur C.23.

C-krav

1. Applikationen ska kunna beräkna och presentera omkrets och area för figurer av typen cirkel, ellips och rektangel. För figurer av typen rätblock, cylinder och sfär ska mantelarea, begränsningsarea och volym beräknas och presenteras.
2. Klasserna Shape, Shape2d, Ellips, Rectangle Shape3D, Cuboid, Cylinder, Sphere, Extensions och Program ska implementeras enligt klassdiagrammen i nivå C. Det gäller även den uppräkningsbara typen ShapeType.
3. Metoden ToString() i klassen Shape2D ska returnera en sträng där figurens namn är vänster justerat och längd, bredd, omkrets och area är högerjusterade.
4. Metoden ToString() i klassen Shape3D ska returnera en sträng där figurens namn är vänster justerat och längd, bredd, höjd, mantelarea, begränsningsarea och volym är högerjusterade.
5. Klasserna Ellipse och Rectangle ska implementera de abstrakta "read-only"-egenskaperna Area och Perimeter i basklassen Shape2D. Area och omkrets ska för respektive typ av figur beräknas enligt uttrycken under rubriken "Formelsamling" på sidan 19.
6. Klasserna Cuboid, Cylinder och Sphere ska implementera de abstrakta "read-only"-egenskaperna MantelArea, TotalSurfaceArea och Volume i basklassen Shape3D. Area och omkrets ska för respektive typ av figur beräknas enligt uttrycken under rubriken "Formelsamling" på sidan 19.
7. Egenskaper som har en set-metod ska validera datat innan det tilldelas fältet egenskapen kapslar in.
8. Metoden Main() i klassen Program ska se till att en meny visas där användaren kan välja för vilken typ av figur beräkningar och presentation ska göras. Det ska även vara möjligt att slumpa 2D- eller 3D-figurer, som sorteras med avseende på area respektive volym, och presentera en tabell över de framslumpade figurerna. Användaren ska via ett menyalternativ kunna välja att avsluta applikationen.
9. Metoden Main() i klassen Program ska se till att figurer av typerna Ellipse och Rectangle slumpas, sorteras och presenteras.
10. Metoden Randomize2DShapes() ska slumpa mellan 5 och 20 objekt av typerna Ellipse och Rectangle. Figurernas längder och bredder ska slumpas till värden av typen double mellan inklusive 5,0 och exklusive 100,0.
11. Metoden ViewShapeDetail() i klassen Program ska använda metoden ToString() i klassen Shape för att presentera en figurs detaljer på flera rader.
12. Metoden ViewShapes() i klassen Program ska använda metoden ToString("R") i klassen Shape för att presentera en figurs detaljer på en rad.
13. Fel som inträffar i applikationen ska tas om hand och relevanta felmeddelanden ska visas.

Läsvärt

- Klasser
 - Essential C# 5.0, 209-220.
 - <http://msdn.microsoft.com/en-us/library/0b0thckt.aspx>.
- Åtkomstmodifierare ("Access Modifiers")
 - Essential C# 5.0, 227-229.
 - <http://msdn.microsoft.com/en-us/library/ms173121.aspx>.

- Egenskaper
 - Essential C# 5.0, 229-235.
 - <http://msdn.microsoft.com/en-us/library/x9fsa0sw.aspx>.
- Konstruktörer
 - Essential C# 5.0, 244-248, 250-253.
 - <http://msdn.microsoft.com/en-us/library/k6sa6h87.aspx>.
- Arv
 - Essential C# 5.0, 277-311.
 - <http://msdn.microsoft.com/en-us/library/ms173149.aspx>.
- IComparable (sortering av objekt)
 - Essential C# 5.0, 320-321.
 - <http://msdn.microsoft.com/en-us/library/ey2t2ys5.aspx>
- Uppräkningsbara typer (enum)
 - Essential C# 5.0, 358-366.
 - <http://msdn.microsoft.com/en-us/library/sbbt4032.aspx>.
- Klassen Array
 - <http://msdn.microsoft.com/en-us/library/czz5hkty.aspx>.
- Klassen Random
 - <http://msdn.microsoft.com/en-us/library/ts6se2ek.aspx>