# Shape and profession aspects

### Description

## Shape aspects

Geometric shapes lend themselves to be represented by objects of classes, but also by aspects.

The following are aspects that are of interest for both 2D shapes and 3D shapes, and which are independent of how shapes are represented as objects. For shapes in 2D, two properties are of interest, the perimeter and the area. For shapes in 3D, two properties are of interest, the surface and the volume.

These aspects are more like interface in other languages, as they do not have any data item, but only methods with signature.

```
shapeApects: module
   class Shape2D:
      area -> a: var Real:<
         inner
      perimeter -> p: var Real:<
         inner

   class Shape3D:
      surface -> s: var Real:<
         inner
      volume -> v: var Real:<
         inner
```

In the first round we define shapes without using points á la (x, y) or (x, y, z), and we define how the aspects are defined for the various shapes. We shall later see that the same aspects may be applied to shapes represented by points.

A circle is defined by its radius, and the Shape2D aspect of the class Circle is described by a singular object with Shape2D as superclass and the area and perimeter defined in terms of the radius:

```
class Circle:
   radius: var Real
   2D: obj Shape2D
      area::
         return PI * radius * radius
      perimeter::
         return 2.0 * PI * radius
```

Rectangles and triangles are described classes with similar singular objects:

```
class Rectangle:
   height, width: var Real
   2D: obj Shape2D
      perimeter::
         return 2 * height + 2 * width
      area::
         return height * width
```

```
class Triangle:
   a,b,c: var Real  -- the three sides
   2D: obj Shape2D
      area::
         s: var Real
         s := (a / 2) + (b / 2) + (c / 2)
         return Math.Sqrt(s * ((s - a * (s - b) * (s - c)))))
```

3D shapes are not specialized 2D shapes; they are rather characterized by an Shape3D aspect. It would of course be possible to represent e.g. a class Cylinder as a subclass of Circle, just adding the hight, but a correct representation/model is rather to define Cylinder as having both a Circle, a height, and the aspect of being a 3D shape:

```
class Cylinder: Shape3D
   base: var Circle
   height: var Real
   3D: obj Shape3D
      surface:
        return 2 * base.area + height * base.perimeter
      volume::
         return base.area * height
```

+++ **Heading**?

Men vil folk ikke mene at Circle, Rectangle, etc er subklasser af Shape2D og tilsvarende for Cylinder?
Og Shape kan vel være super til Shape2D og Shape3D

+++ Her kunne man så lave nogle tilsvarende klasser, men baseret på Point. De samme aspects kan også bruges til disse, og så kunne man tilføje nogle som havde at gøre med grafik (f.eks. draw) og move, men dette har jeg ikke gennemført før vi bestemmer os for om vi skal have det med.

```
class Point(x, y: var Integer):
   ...

class Circle:
   center: var/ref Point
   radius: var Real
   2D: obj Shape2D
      area::
         return PI * radius * radius
      perimeter::
         return 2.0 * PI * radius

class Triangle:
   a,b,c: var Point
   2D: obj Shape2D
      area::
         return abs(
            (a.x * (b.2 – c.y) + b.x * (c.y – a.y) +
             c.x * (a.y – b.y)) / 2.0
            )

class Profession:
   se nedenfor
   full time/part time
   education
   training
   association

class Nationality:
   placeOfBirth: ref Nation

class Nation:
   ...

class Denmark: Nation
```

```
   ...

class Teacher: Profession
   where: ref School
   teaching: obj Set(Discipline)

class Danish: Nationality
   ???
   placeOfBirth := Denmark

class Person(name: var String):
   prof: obj Teacher
   prof: ref Profession       -- may change profession

   nation: obj Danish
   nation: ref Nationality    -- may change nationality
```

Profession, fra https://en.wikipedia.org/wiki/Profession

There is considerable agreement about defining the characteristic features of a profession. They have a “professional association, cognitive base, institutionalized training, licensing, work autonomy, colleague control… (and) code of ethics”,[30] to which Larson then also adds, “high standards of professional and intellectual excellence,” (Larson, p. 221) that “professions are occupations with special power and prestige”, (Larson, p.x) and that they comprise “an exclusive elite group,” (Larson, p. 20) in all societies. Members of a profession have also been defined as “workers whose qualities of detachment, autonomy, and group allegiance are more extensive than those found among other groups…their attributes include a high degree of systematic knowledge; strong community orientation and loyalty; self-regulation; and a system of rewards defined and administered by the community of workers.”[31]

A profession has been further defined as: “a special type of occupation…(possessing) corporate solidarity…prolonged specialized training in a body of abstract knowledge, and a collectivity or service orientation…a vocational sub-culture which comprises implicit codes of behavior, generates an esprit de corps among members of the same profession, and ensures them certain occupational advantages…(also) bureaucratic structures and monopolistic privileges to perform certain types of work…professional literature, legislation, etc.”[32]

A critical characteristic of a profession is the need to cultivate and exercise professional *discretion* – that is, the ability to make case by case *judgements* that cannot be determined by an absolute rule or instruction.[33]

Det følgende er diskussioner som har været pr email:

```
class Teacher(where: ref School): Profession
   ...

class Danish(kindOfDanish: var String): Nationality
   ...

class Person(name: var String, where):
   asTeacher: obj Teacher(where) ...
   asDanish: obj Danish(kindOfDanish) ...

class Person+(name: var String, where):
   asTeacher: obj Teacher(where) ...
   asNorwegian: obj Norwegian(kindOfNorwegian) ...

-----------------------------------------------------------
Person(”mads madsen”).(.asTeacher(Tønder)(.asDanish("Jyde"))

Person(”mads madsen”).(asTeacher(Tønder), asDanish("Jyde"))

Person(”mads madsen”).asTeacher(Tønder).asDanish("Jyde")

-----------------------------------------------------------
class Person(name: var String):
   as Teacher ...
   as Danish ...
```