2. Egyszerű osztályok II.

1. Adott síkbeli pontok közül hány esik rá egy adott kör lemezére?

Specifikáció:

A = (x:Pont*, k:Kör, db:ℕ)

Ef = (x=x' ∧ k=k')

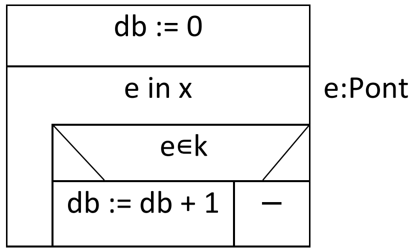
$Uf = (Ef \wedge db = \sum_{e \text{ in } x',\ e \in k} 1)$

Számlálás

enor(Pont) ~ Pont* felsorolása

felt(i) ~ e∈k

Algoritmus:



Megj: Ha az x egy [1..n] indexelésű tömb lenne, akkor a tömb elemeinek felsorolását egy i=1..n számlálós ciklussal is végezhetnénk; az e változó helyett pedig x[i]-t kellene használni.

Kör és a Pont típusa. Ábrázoljuk a köröket a középpontjukkal és a sugarukkal, a pontokat a koordinátájukkal.

Típusdefiníciók:

Kör

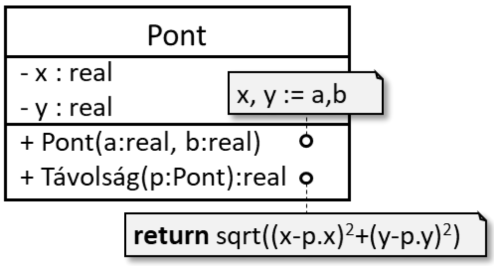
körök	l := p∈k (k:Kör, p:Pont, l:𝕃)
c : Pont r : ℝ Inv: r ≥ 0	l := $\lvert\overline{c, p}\rvert \le r$

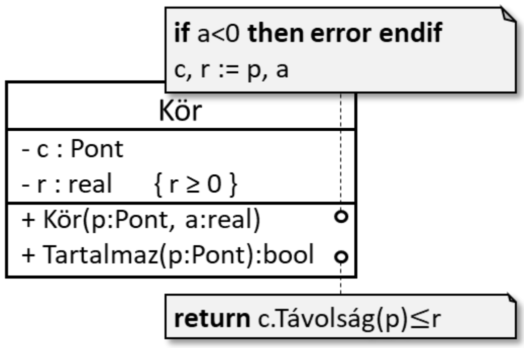
Pont

pontok	d := $\lvert\overline{q, p}\rvert$ (p, q :Pont, d:ℝ)
x, y :ℝ	d := $\sqrt{(p.x - q.x)^2 + (p.y - q.y)^2}$

Megj: A tervezés során inkább a „felülről-lefelé” irányt követjük, de az objektum-orientált kódolás az „alulról-felfelé” építkezést szereti.

Osztályok:





Megj: Két pont távolsága, illetve egy pontnak egy másiktól való távolsága nem eltérő fogalmak, ám metódusként való leírásuk különbözhet. Itt a második értelmezés jelenik meg: egy adott pontra (c) kell meghívni a Távolság() metódust egy másik ponttal (p), hogy a két pont távolságát kiszámoljuk: c.Távolság(p). A további példákban viszont két egyenlő súlyú objektum műveleteivel találkozunk, amelyeket ezért osztályszintű metódusként vezetünk be.

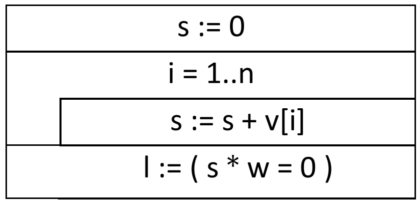
2. Adott síkvektorok összege merőleges-e egy adott síkvektorra (skaláris szorzatuk nulla-e).

Specifikáció:

A = (v:Vector ^{n}, w:Vector, l: $\mathbb{L}$)
Ef = (v=v' $\wedge$ w=w')

Uf = (Ef $\wedge$ l = ($(\Sigma_{i=1..n}\, v[i])*w = 0.0$))

Algoritmus:



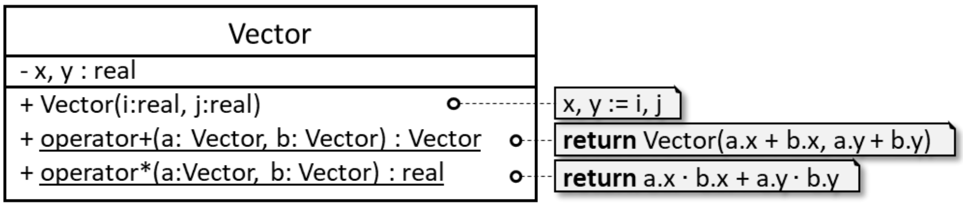
s:Vector
i:$\mathbb{Z}$

Típusdefiníció: **Vector**

síkvektorok	c := a+b	(a, b, c : Vector)
	s := a·b	(a, b: Vector, s:$\mathbb{R}$)
x, y:$\mathbb{R}$	c.x, c.y := a.x+b.x, a.y+b.y	
	s := a.x·b.x + a.y·b.y	

A programok leírásában meg kell különböztetnünk, hogy mikor beszélünk az 'a', a 'b', vagy a 'c' vektor x koordinátájáról: a.x, b.x, illetve c.x.

Osztály:



Az összeadás és a skaláris szorzás bemenete nem egy vektorról szól: a bemenetük két vektor, az összeadásnak a kimenete egy harmadik. Nem lenne elegáns (bár megtehetnénk), ha ezeket a műveleteket egyetlen Vector típusú objektum műveleteiként vezetnénk be. Ehelyett ezek a Vector osztály (osztályszintű) metódusai lesznek, és ezeket nem egy kitüntetett vektor objektumra kell meghívni úgy, hogy paraméterként adjuk a másik vektort, hanem olyan metódusként, amelynek két vektor partamétere van, az összeadás esetében pedig a Vector típusú visszatérési értéke.

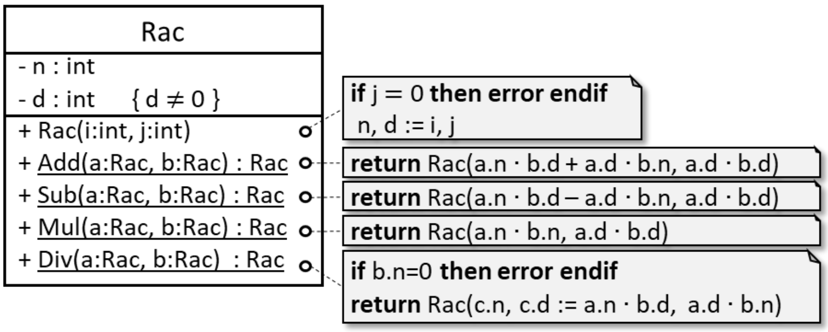
3. Racionális számok. (Ábrázoljuk a racionális számokat egész számpárokkal.)

Típusdefiníció: **Rac**

$\mathbb{Q}$	c := a ± b (a, b, c: Rac)
	c := a · b (a, b, c: Rac)
	c := a / b (b≠0) (a, b, c: Rac)
n, d: $\mathbb{Z}$ Inv: d≠0	c.n, c.d := a.n · b.d ± a.d · b.n, a.d · b.d
	c.n, c.d := a.n · b.n, a.d · b.d
	if b.n=0 **then error endif** c.n, c.d := a.n · b.d, a.d · b.n (b.n≠0)

A típusinvariáns lehetne a d>0 is, vagy „n és d relatív prím” is.

Osztálydiagram:



4. Komplex számok. (Ábrázoljuk a komplex számokat az algebrai alakjukkal (x+y·i).)

Típusdefiníció: **Komplex**

$\mathbb{C}$	c := a±b (a, b, c: Komplex)
	c := a*b (a, b, c: Komplex)
	c := a/b (b≠0) (a, b, c: Komplex)
x, y: $\mathbb{R}$ // x+i·y	c.x, c.y := a.x ± b.x, a.y ± b.y
	c.x, c.y := a.x · b.x – a.y · b.y , a.x · b.y + a.y · b.x
	if b.x=0 **or** b.y=0 **then error endif** c.x, c.y := (a.x · b.x + a.y · b.y) / (b.x ^{2}+ b.y ^{2}), (a.y · b.x – a.x · b.y) / (b.x ^{2}+ b.y ^{2})

Osztálydiagram: