3. LABOR

OSZTÁLY ÉS OBJEKTUM: SZINTAKTIKA ÉS KONVENCIÓK

Hallgatónak: általános információk

1 iMSc pont jár az 5. és 6. feladatok együttes teljesítéséért.

Kötelező feladatok

1. Emlékeztető

Közösen tekintsétek át az eddig tanult OOP alapvetéseket:

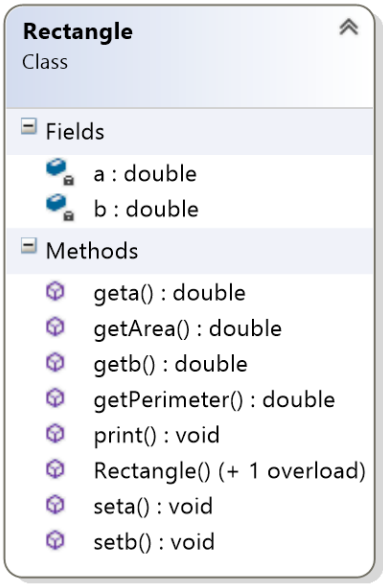
- Egységbezárás (encapsulation)
  - accessor/getter
  - mutator/setter
- Absztrakció (abstraction)
  - adatmodell

2. Rectangle osztály

Tipp: osztály létrehozás gyorsan

- Ahelyett, hogy külön létrehoznál egy myClass.h és egy myClass.cpp-t
- Jobb klikk a projekt nevére -> Add -> Class…

Hozz létre egy új *„Empty project" projektet* (*ShapePractice*), majd írj egy téglalapot modellező osztályt (*Rectangle*), ami legyen képes a következőkre:



1. ábra UML diagramm a Rectangle osztályhoz

- Ki tudja számolni a *téglalap területét, kerületét*.
- Az osztály *deklarációját rectangle.h állományba* tároljuk
  - Gondoskodjunk róla, hogy egy esetleges *többszöri beépítés ne jelentsen problémát*.
  - A terület-kerület számító függvényeket az *osztályon kívül definiáljuk: rectangle.cpp*
- Téglalap inicializálás után *mindig korrekt értékeket* tartalmazzon. Inicializálni lehessen:
  - Alapértelmezett értékek megadása nélkül (ekkor mindegyik oldal nulla lesz), illetve
  - Egy érték megadásával (ekkor a paraméterben megadott oldalú négyzet lesz), illetve
  - Mindkét oldal megadásával.
- Az oldalakat ne lehessen kívülről rosszindulatúan megváltoztatni (negatív értékre beállítani), viszont a téglalap oldalait le lehessen kérdezni.
- Diagnosztikai céllal írjunk egy publikus kiíró függvényt (*print()*), amely kiírja az oldalakat!

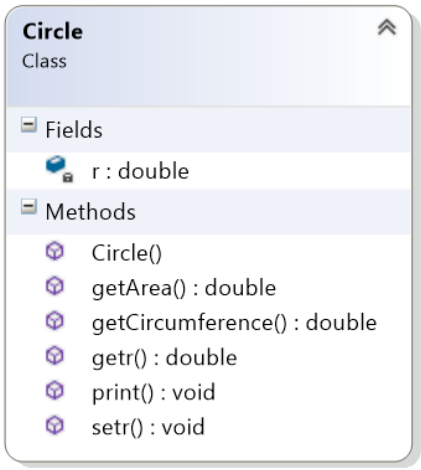
Használd fel a következő kódrészletet tesztelésre:

```
printf("Testing Rectangle...\n");
Rectangle r1;
Rectangle r2(4);
Rectangle r3(2, 5);

r1.print(); // a=0.00, b=0.00
r2.print(); // a=4.00, b=4.00
r3.print(); // a=2.00, b=5.00

printf("Area of r1=%.2lf\n", r1.getArea()); // Area of r1=0.00
printf("Area of r2=%.2lf\n", r2.getArea()); // Area of r2=16.00
printf("Area of r3=%.2lf\n", r3.getArea()); // Area of r3=10.00

printf("Perimeter of r1=%.2lf\n", r1.getPerimeter()); // Perimeter of r1=0.00
printf("Perimeter of r2=%.2lf\n", r2.getPerimeter()); // Perimeter of r2=16.00
printf("Perimeter of r3=%.2lf\n", r3.getPerimeter()); // Perimeter of r3=14.00

r3.seta(r2.geta());
r3.setb(r2.getb());
r3.print(); // a=4.00, b=4.00
printf("Area of r3=%.2lf\n", r3.getArea()); // Area of r3=16.00

printf("Perimeter of r3=%.2lf\n", r3.getPerimeter()); // Perimeter of r3=16.00
```

3. Circle osztály

Az előző projekt felhasználásával, írj egy kört reprezentáló osztályt (*Circle*).

- Az osztály tárolja kívülről hozzáférhetetlenül a kör sugarát (*r*), amit az osztály példányosításánál kezdeti értékként be lehessen állítani, ha nem adunk meg kezdeti értéket, akkor zérust feltételezünk. A sugár értékeket inicializálás után lehessen változtatni, de a sugár értékét csak pozitív értékekre.
- Az osztály képes legyen megmondani a kör kerületét és területét!
- A program tagolása (*circle.h*: osztálydeklaráció, *circle.cpp*: hosszabb implementációjú függvények törzse) a C++ programozók körében elterjedt konvenciók alapján történjen, és védjük le a többszörös beépítésből eredő újradefinicióról szóló hibaüzeneteket.
- Írjunk egy diagnosztikai függvényt, amely kiírja a tagváltozó értékét (*print()*).

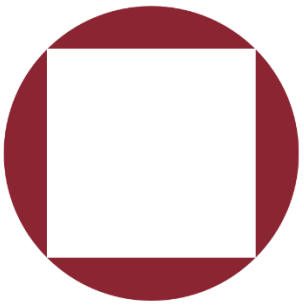


2. ábra UML diagramm a Circle osztályhoz

Használd fel a következő kódrészletet tesztelésre:

```
printf("\nTesting Circle...\n");
Circle c1;
Circle c2(10);

c1.print(); // r=0.00
c2.print(); // r=10.00

printf("Area of c1=%.2lf\n", c1.getArea()); // Area of c1 = 0.00
printf("Area of c2=%.2lf\n", c2.getArea()); // Area of c2 = 314.16

printf("Circumference of c1=%.2lf\n", c1.getCircumference()); // Circumference of c1 = 0.00
printf("Circumference of c2=%.2lf\n", c2.getCircumference()); // Circumference of c2 = 62.83

c1.setr(c2.getr());
c1.print();
printf("Radius of c1=%.2lf\n", c1.getr()); // Radius of c1=10.00
printf("Area of c1=%.2lf\n", c1.getArea()); // Area of c1=314.16
printf("Circumference of c1=%.2lf\n", c1.getCircumference()); // Circumference of c1=62.83
```

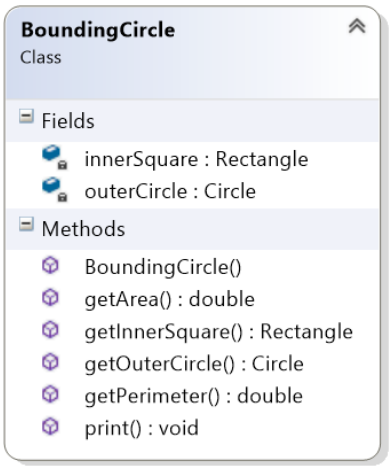
4. Kompozíció (composition) kapcsolat



3. ábra A síkidom részei a bordó részek

Készítsd el az 1. ábrán látható síkidomot reprezentáló *BoundingCircle* nevű osztályt. (Egy ilyen síkidom úgy kapható meg, hogy egy négyzetet kivágsz az azt minimálisan befoglaló körlapból.)

- Privát tagváltozói:
  - innerSquare: Rectangle
  - outerCircle: Circle
- Ezeket lehessen konstruktorral beállítani
  - Alapértelmezett konstruktor:
    - innerSquare legyen egységnyi oldalhosszúságú
    - outerCircle legyen $\sqrt{2}/2$ sugarú
  - Legyen egy másik konstruktor, ami a négyzet oldalhosszúságát várja. Ebből kiszámítandó a kör sugara (a*$\sqrt{2}/2$). Az oldalhosszúság alapértelmezetten 1.
- Legyen továbbá egy print() diagnosztikai függvénye, ami meghívja a tagváltozók print()-jeit
- Írj gettert mind a két tagváltozó számára, hogy kívülről külön-külön is elérhetők legyenek
- Ügyelj a konstans referenciák használatára, ott ahol szükséges
- Legyen kerület és terület kiszámoló függvénye is



4. ábra UML diagramm a BoundingCircle osztályhoz

Használd fel a következő kódrészletet tesztelésre:

```
printf("\nTesting BoundingCircle...\n");
BoundingCircle b1;
BoundingCircle b2(10);

b1.print(); // inner square: a=1.00, b=1.00\nouter circle: r=0.71
b2.print(); // inner square: a=10.00, b=10.00\nouter circle: r=7.07

printf("Area of c1=%.2lf\n", b1.getArea()); // Area of c1=0.57
printf("Area of c2=%.2lf\n", b2.getArea()); // Area of c2=57.08
printf("Perimeter of c1=%.2lf\n", b1.getPerimeter()); // Perimeter of c1=8.44
printf("Perimeter of c2=%.2lf\n", b2.getPerimeter()); // Perimeter of c2=84.43

b1.getInnerSquare().print(); // a=1.00, b=1.00
b1.getOuterCircle().print(); // r=0.71
```

***Megjegyzés**: 1) az olyan osztályokat, amelye létrehozásuk után nem módosíthatók, **immutable** osztályoknak hívjuk. 2) a most megírt osztály lehetne **mutable** (létrehozás után módosítható) is.*

5. Modellezés UML segítségével

Visual Studio-ban tekintsétek meg az elkészült feladatok UML diagrammját.

VS Class Diagram megtekintés

- Jobb klikk projekt nevére -> View -> View Class Diagram
- Jobb klikk az egyik osztályra a megjelenő felületen -> Expand
  - ha nem lenne kijelölve az összes: CTRL+A
- Jobb klikk az üres felületre
  - Change Members Format -> Display Full Signature
  - Adjust Shapes Width

6. Bevezetés a polimorfizmusba

- Tegyük fel, hogy készítesz külön-külön 100 Rectangle-t, 100 Circle-t és 100 BoundingCircle-t tároló tömböt (*rectangles, circles, boundingCircles*).
- Amiket feltöltesz véletlenszerű tulajdonságokkal rendelkező, megfelelő típusú objektumokkal.
- Ezek után meghívnád mindegyiken a print függvényt.

Vitassátok meg, hogy hogyan lehetne megoldani ezt a rendkívül kellemetlen kódismétlést és erőforráspazarlást.

(A konkrét megvalósítás egy későbbi labor során fog előkerülni, de hasznos, ha már most felmerül bennetek, hogy ez bizony egy jelentős probléma, de valahogy meg is lehet oldani…).