Osztály külön fájlba. Setterek

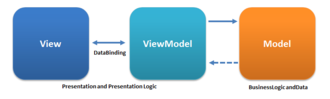
# A feladat szövege:

Vektorgrafikus rendszerekben lehetőség van négyzet rajzolására. Az ilyen négyzetek méretét lehetséges változtatni. Az ilyen CAD rendszereknél fontos lehet a négyzet területének és kerületének meghatározása. Készítsen olyan négyzet osztályt, amelynek adott az oldalhossza! Az oldalhossz lekérhető és módosítható tulajdonsága legyen a négyzetnek! A négyzet területe és kerülete legyen lekérdezhető tulajdonsága a négyzetnek!

Példányosítson egy ilyen négyzetet és a főprogramban mutassa be a tulajdonságok kezelését!

Az osztály a projekt model rétegében készítse el egy külön fájlba!

A weben elérthető [MVVM](https://hu.wikipedia.org/wiki/Modell-n%C3%A9zet-n%C3%A9zetmodell) tananyag.



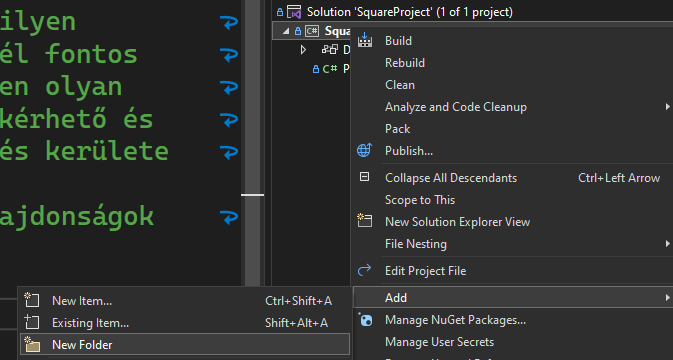
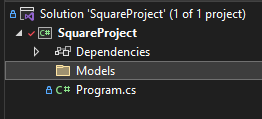
## Projekt létrehozása

Hozzon létre egy projektet „SquareProject néven.

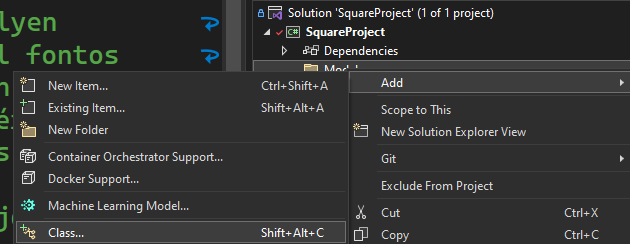
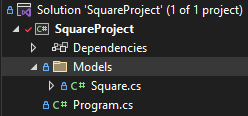
## Model réteg létrehozása

Készítsen a projektben egy Models mappát. Square osztály a model rétegben és a Square.cs állományban.

1. Models mappa



1. A Models mappába hozzon létre egy „Square” osztályt:



**public** class Square

{

}

* Ismert a négyzet oldalának hossza
* Négyzetet konstruktora amely megkapja a négyzet oldalát és eltárolja

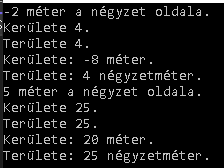
# Tulajdonságok

A weben elérhető [C# tulajdonságok](https://csharptutorial.hu/docs/hellovilag-hellocsharp/5-osztalyok-objektumok/tulajdonsagok/) tananyag.

* Side tulajdonság get metódussal
* Side tulajdonság set metódusa
* A négyzet adatait ilyen szövegként jelenítjük meg
* A megírt kódok tesztelése

# Kivételkezelés

A program negatív oldal érték esetén hibás válaszokat ad



Ha a négyzet oldala negatív vagy nulla kivételt dobunk.

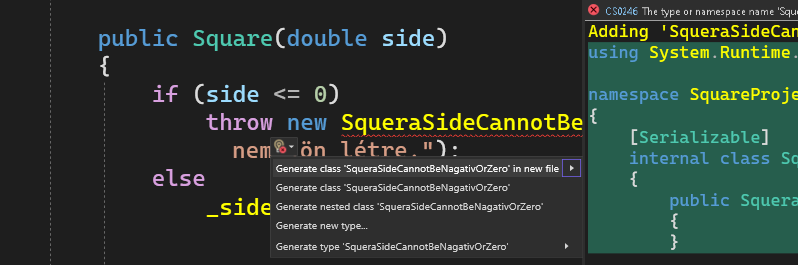
[Kivételkezelés – C# programozási útmutató - C# | Microsoft Learn](https://learn.microsoft.com/hu-hu/dotnet/csharp/fundamentals/exceptions/exception-handling)

Kivétel dobódásakor az adott utasítássorozat végrehajtása megszakad, és a vezérlés átadódik a kivételkezelő blokkokhoz.

SqueraSideCannotBeNagativOrZero saját kivétel.

* A saját kivétel nevét beszédesre választjuk.
* A paraméterbe a kivétel szövege kerül.

A saját kivételt az IDE segítségével egy új fájlba legyártjuk.



A saját kivételt a new utasítással példányosítjuk.

Elékészül a SqueraSideCannotBeNagativOrZero.cs állomány.

* Módosításkor is megadhatunk negatív oldalt
* Ha negatív oldalú négyzetet példaányosítunk akkor elkapjuk a kivételt

int wrongSide = -2;

try

{

Square mySquare = new Square(wrongSide);

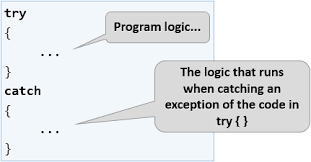
Console.WriteLine(mySquare);

Console.WriteLine("Kerülete: " + mySquare.Perimeter + " méter.");

Console.WriteLine("Területe: " + mySquare.Area + " négyzetméter.");

}

catch

{ }

A kivételkezelés egy programozási mechanizmus, melynek célja a program futását szándékosan vagy nem szándékolt módon megszakító esemény (hiba) vagy utasítás kezelése.

<https://hu.wikipedia.org/wiki/Kivételkezelés>

Kivételkezelő blokk: a try blokk tartalmazza a kivételkezeléssel védett utasításokat.

<https://csharptutorial.hu/docs/hellovilag-hellocsharp/4-vezerlesi-szerkezetek-es-blokkok/try-catch/>

* A kivétel szövegét megjeleníthetjük a képernyőn.



* catch ágból lehet több

A try blokk után állhat egy finally blokk is.

A finally blokk opcionális része a kivételkezelésnek. A benne elhelyezett kód akkor is lefut, ha a vezérlés átkerül a catch blokkba. Leginkább olyan esetekben van értelme a használatának, ha kivétel esetén is fel szeretnénk szabadítani az erőforrást, amit használunk.

A hibát loggolhatjuk a consolra. (output ablak). Lásd a képet a következő oldalon.

