Proof of implementing the code guidelines

Inhoud

[Primitive types, Operators 3](#_Toc168321583)

[LINQ: 4](#_Toc168321584)

[Acces modifiers 4](#_Toc168321585)

[Properties 5](#_Toc168321586)

[Tuples 5](#_Toc168321587)

[Arrays 5](#_Toc168321588)

[Selection 6](#_Toc168321589)

[Loops 7](#_Toc168321590)

[List 8](#_Toc168321591)

[Class 9](#_Toc168321592)

[Constructor 10](#_Toc168321593)

[Fields 11](#_Toc168321594)

[Methods 12](#_Toc168321595)

[Classes: objects as fields, in methods 13](#_Toc168321596)

[Static classes: methods, fields, constructors 14](#_Toc168321597)

[JSON (reading/writing) 15](#_Toc168321598)

[JSON (exception handling) 16](#_Toc168321599)

[Polymorphism overloading 16](#_Toc168321600)

[Inheritance 17](#_Toc168321601)

[Abstraction: interfaces 17](#_Toc168321602)

[Abstraction: Abstract classes 17](#_Toc168321603)

[Generics 17](#_Toc168321604)

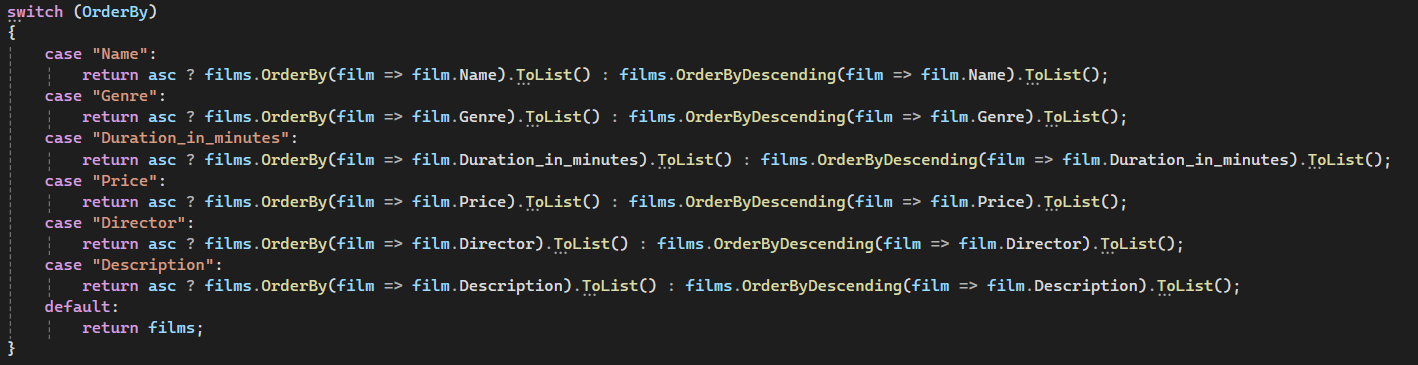
# Primitive types, Operators

Afbeelding met tekst, software, Lettertype, Multimediasoftware

Automatisch gegenereerde beschrijving

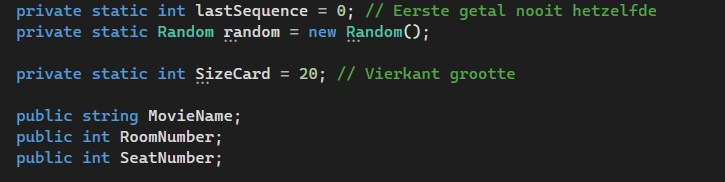
Er wordt hier gebruikt gemaakt van 2 fields om de film korting te geven en hoeveel uur van te voren de korting moet starten.

# LINQ:



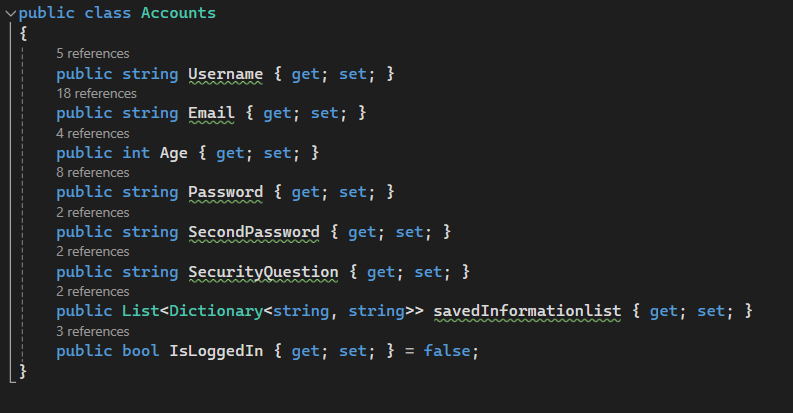
Gebruikt om de list te sorteren i.p.v. een for loop, want het is makkelijker.

# Access modifiers



Sommige variabele zijn hier private, zodat ze niet in andere files gebruikt kunnen worden en alleen in de file zelf. Dit brengt visuele duidelijkheid voor ons.

# Properties



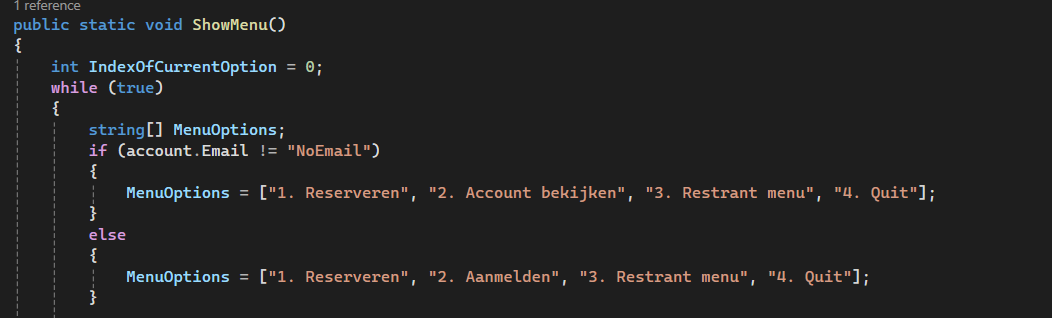
Hier kan je zien dat de Account class verschillende properties heeft.

# Tuples



We hadden een functie nodig die 2 dingen returnen dus gebruikte we in AddAccount de tuple.

# Arrays



We hebben arrays gebruikt om menu opties op te slaan

# Selection

Afbeelding met tekst, schermopname, software

Automatisch gegenereerde beschrijving

Hier wordt een if statement gebruikt om te kijken of de user password klopt nadat de user zijn wachtwoord wilt wijzigen er wordt 2 gevraagd voor een nieuwe wachtwoord voor bevestiging dus er wordt gecheckt of ze allebei hetzelfde zijn.

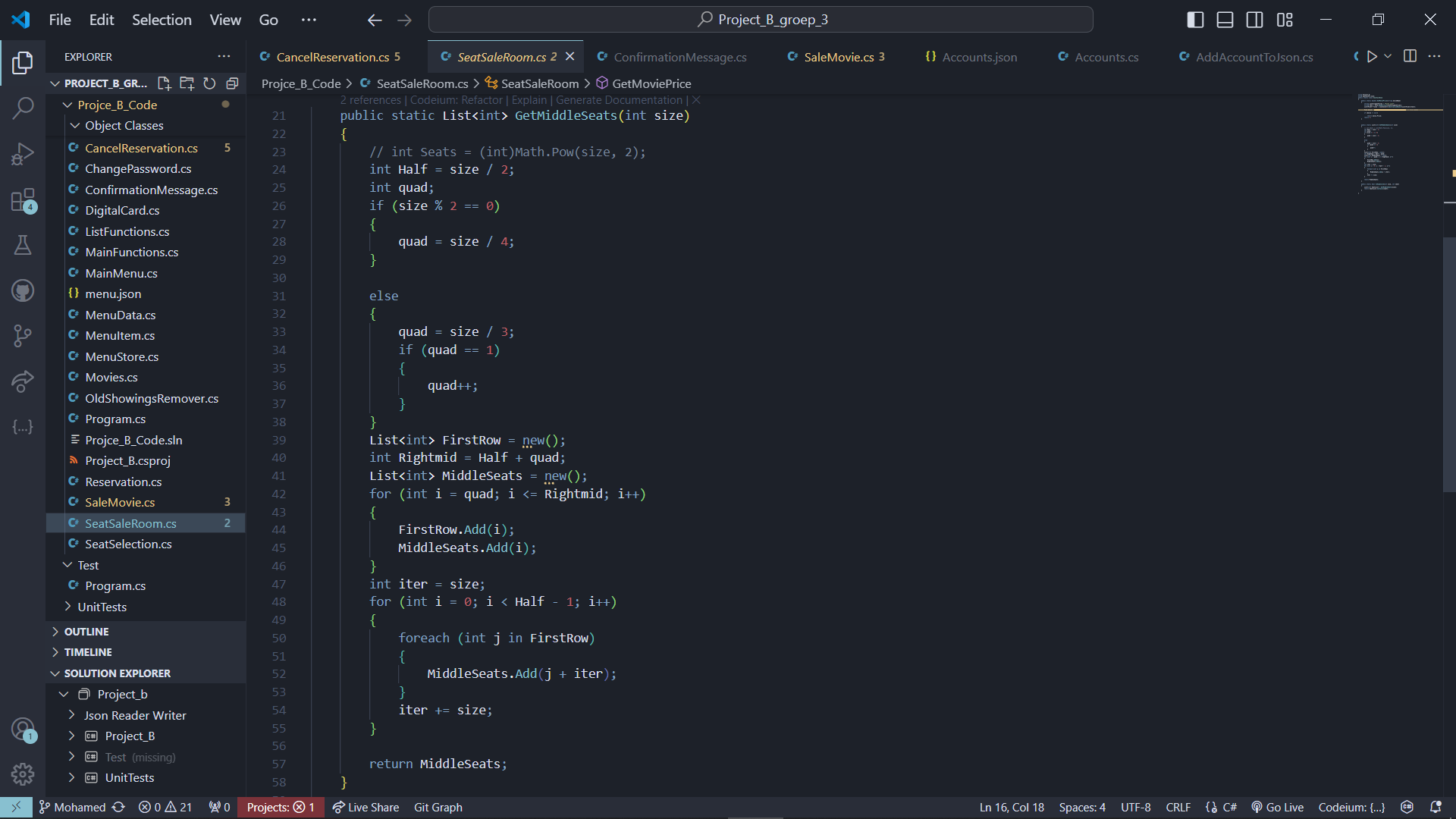
# Loops

Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving

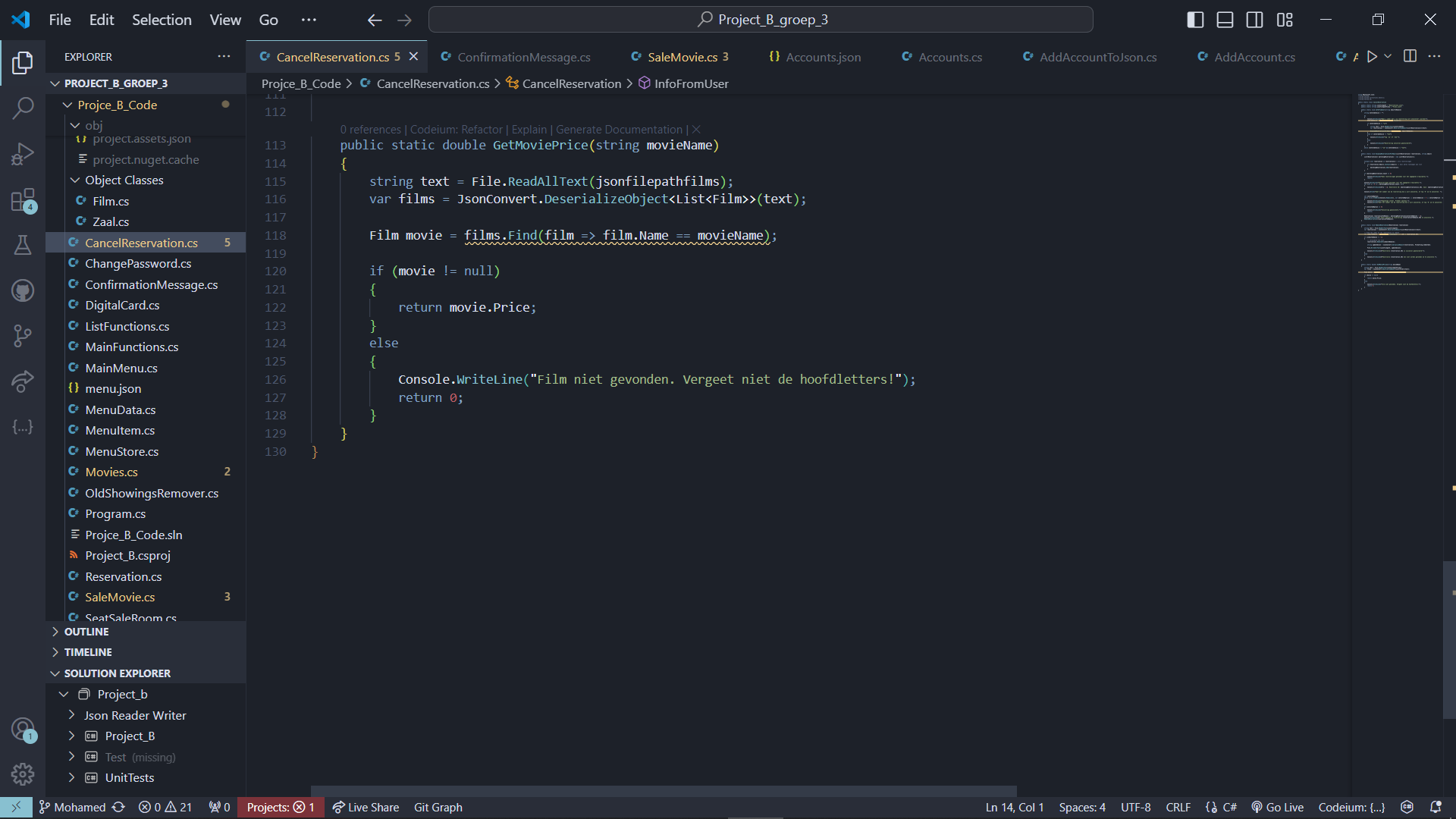
Hier wordt een for loop gebruikt om de stoelen te pakken die in het midden van de zaal zijn. Het werkt zo: stel je voor de zaal bestaat uit 400 stoelen, dan zijn er 20 rijen en 20 kolommen. De 20 rijen wordt door 2 gedeeld (dus 10) en de antwoord daar op wordt door 2 gedeeld (dus 5) dan doe 10-5 en 10+5 dan heb je de 1e middelste rij, en dat doe je 10x omdat 10 helft is aan de aantal rijen.

# List



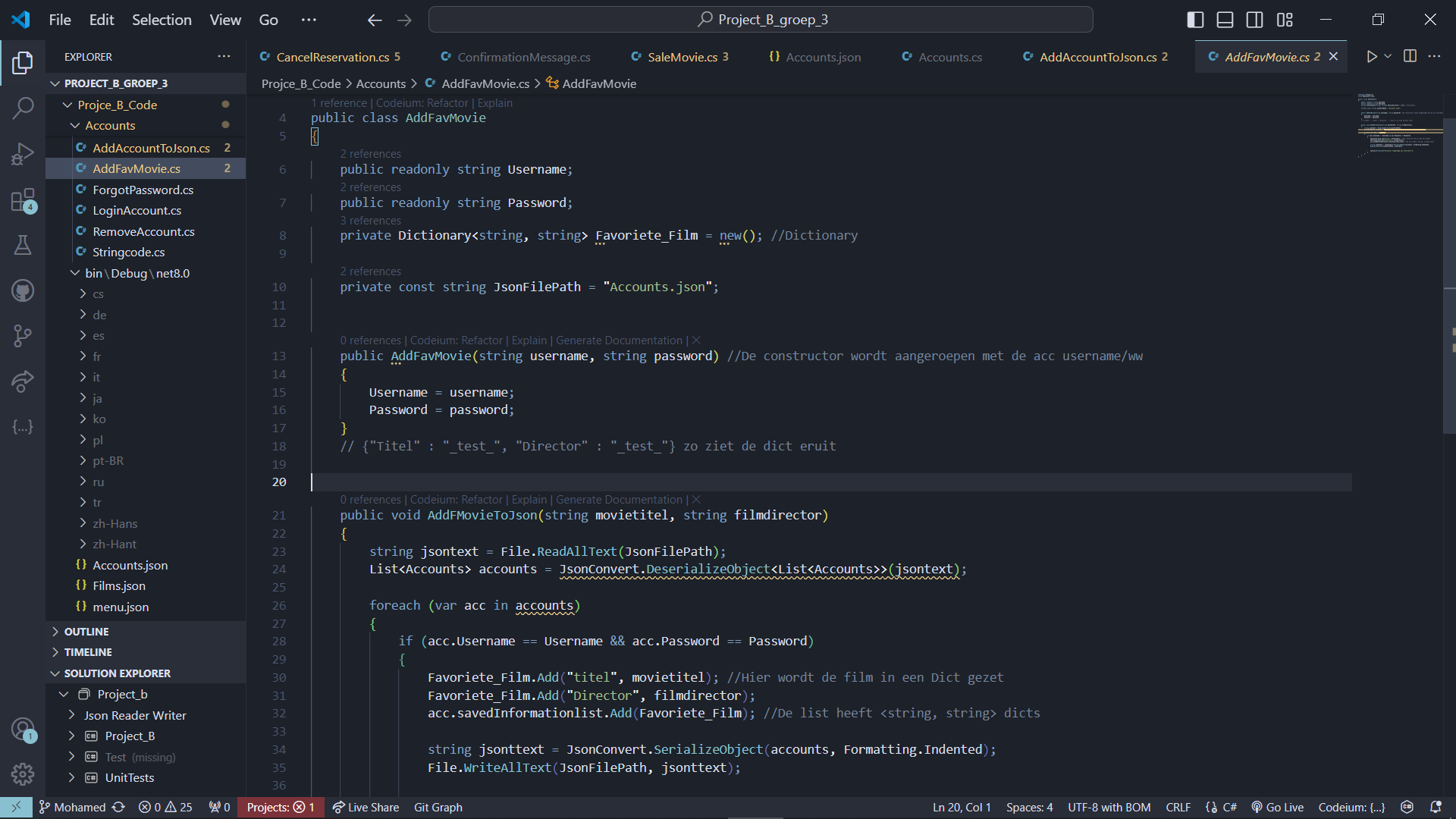
Hier worden 2 list gebruikt namelijk de firstrow en middleseats. De firstrow wordt gebruikt zodat de for loop goed kan functioneren.

# Class



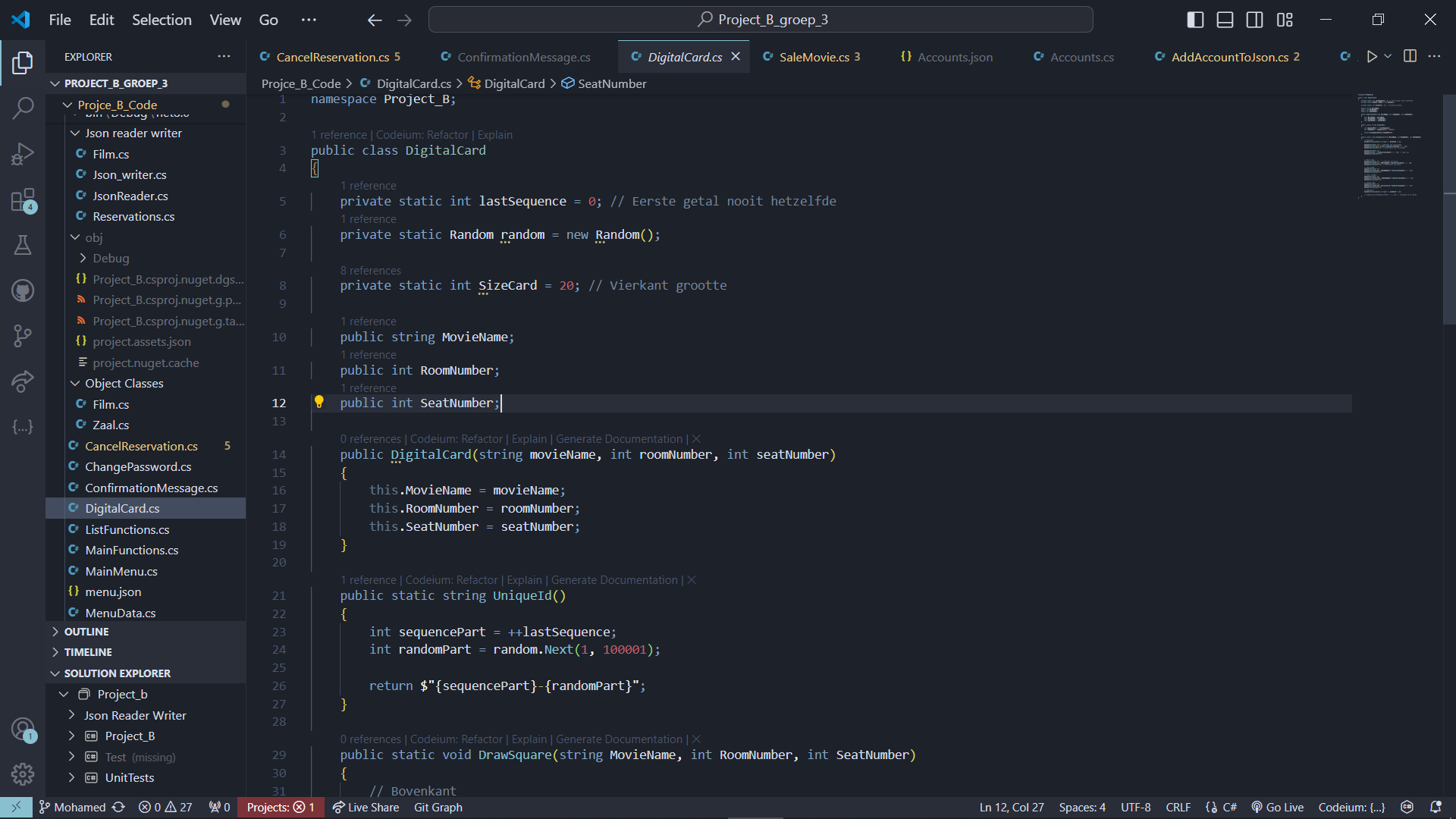
Hier word teen method gebruikt om de film prijs te krijgen.

# Constructor



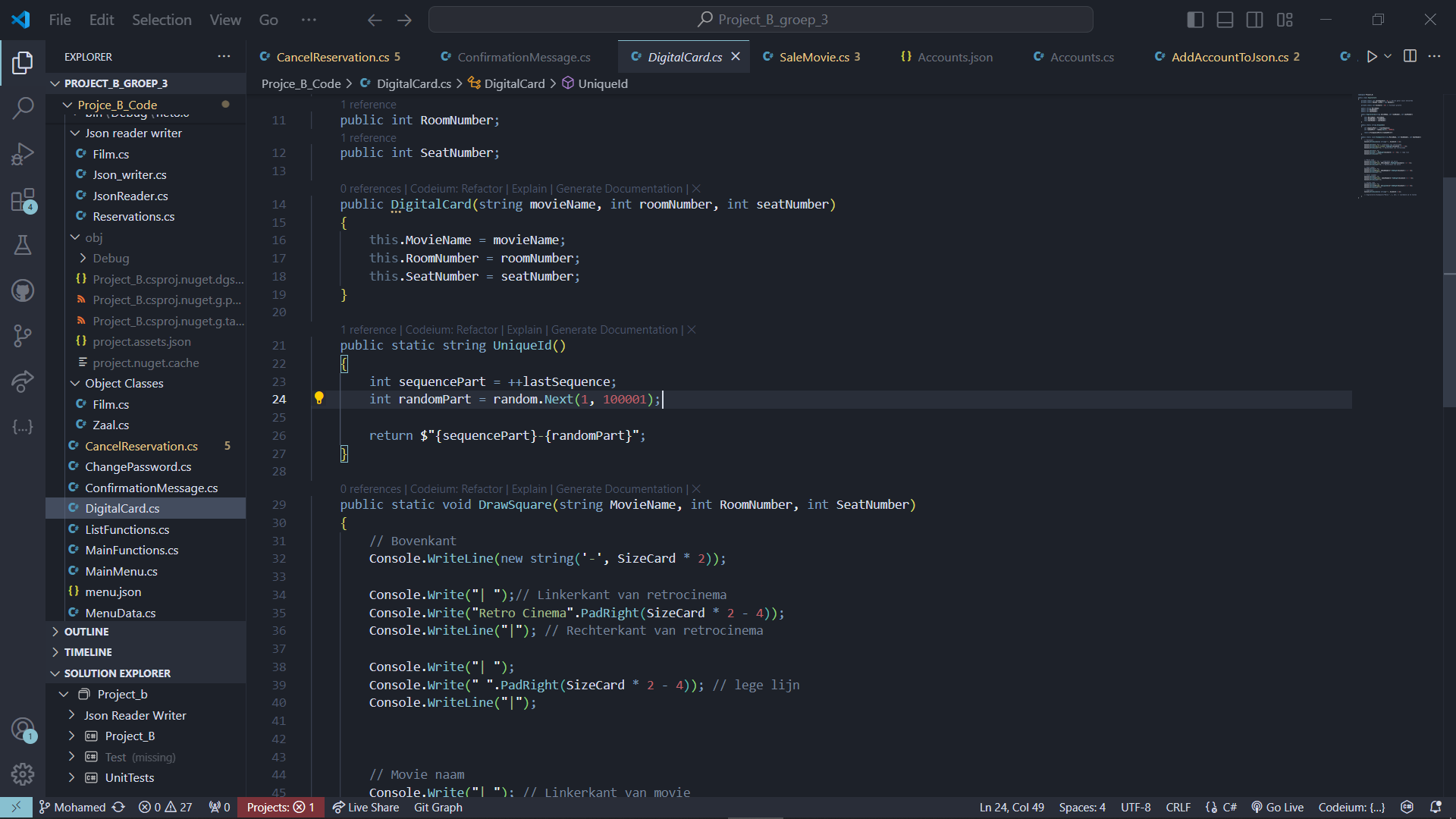
Hier word teen constructor gebruikt om de wachtwoord en username te krijgen om te daarna de films toe te voegen aan de user favoriete films lijst.

# Fields



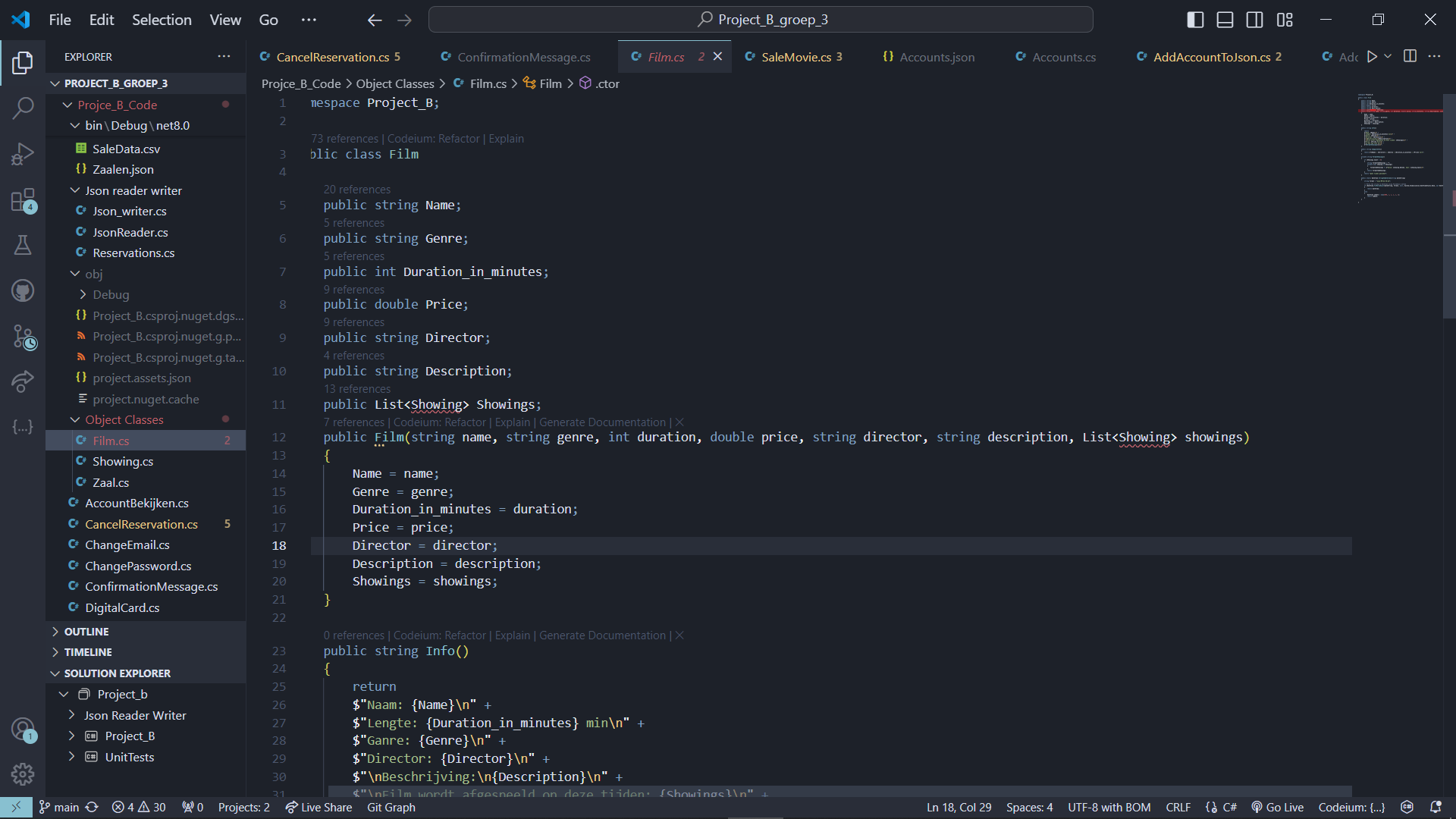
Hier worden fields gebruikt om informatie over de zaal en film te kunnen onderscheiden en zo een een digitale kaartje te printen met de informatie over zijn reservatie.

# Methods



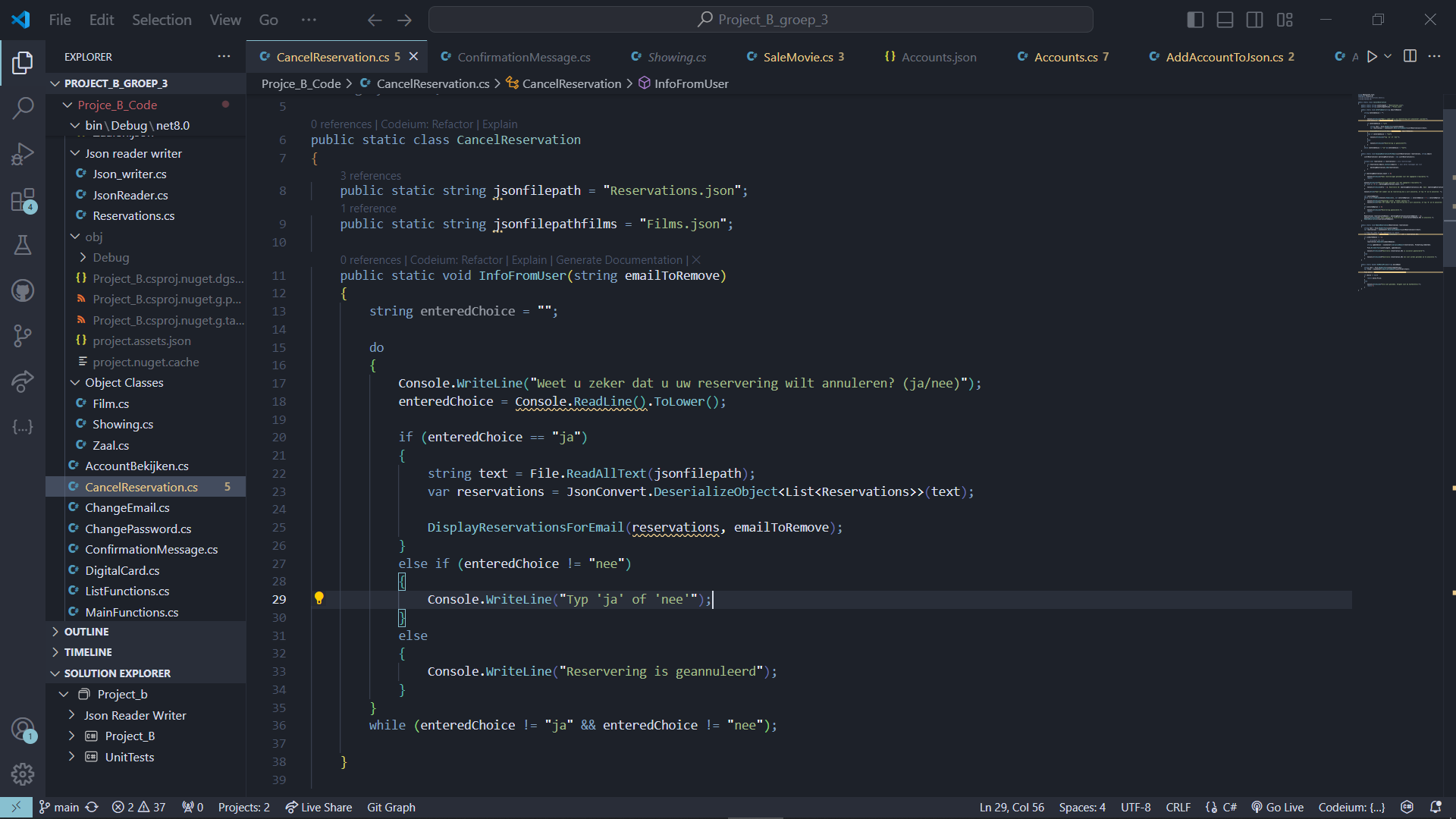
Hier wordt een method gebruikt om random getallen te creëren maar de eerste getal is nooit random maar die wordt steeds met 1 opgeteld.

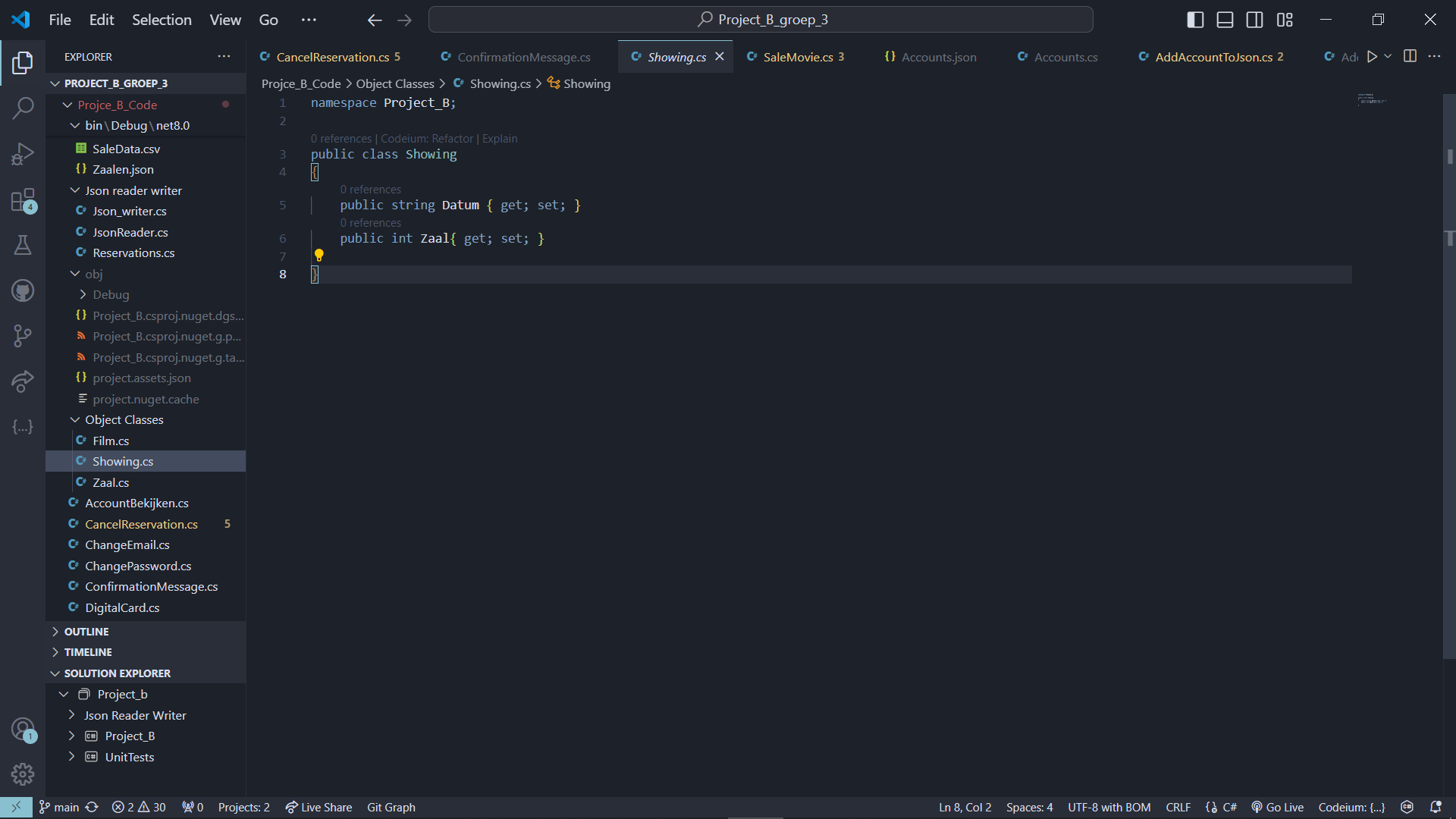
# Classes: objects as fields, in methods



Hier is een object gemaakt van de class Showings om zo de films beter te kunnen detailleren.

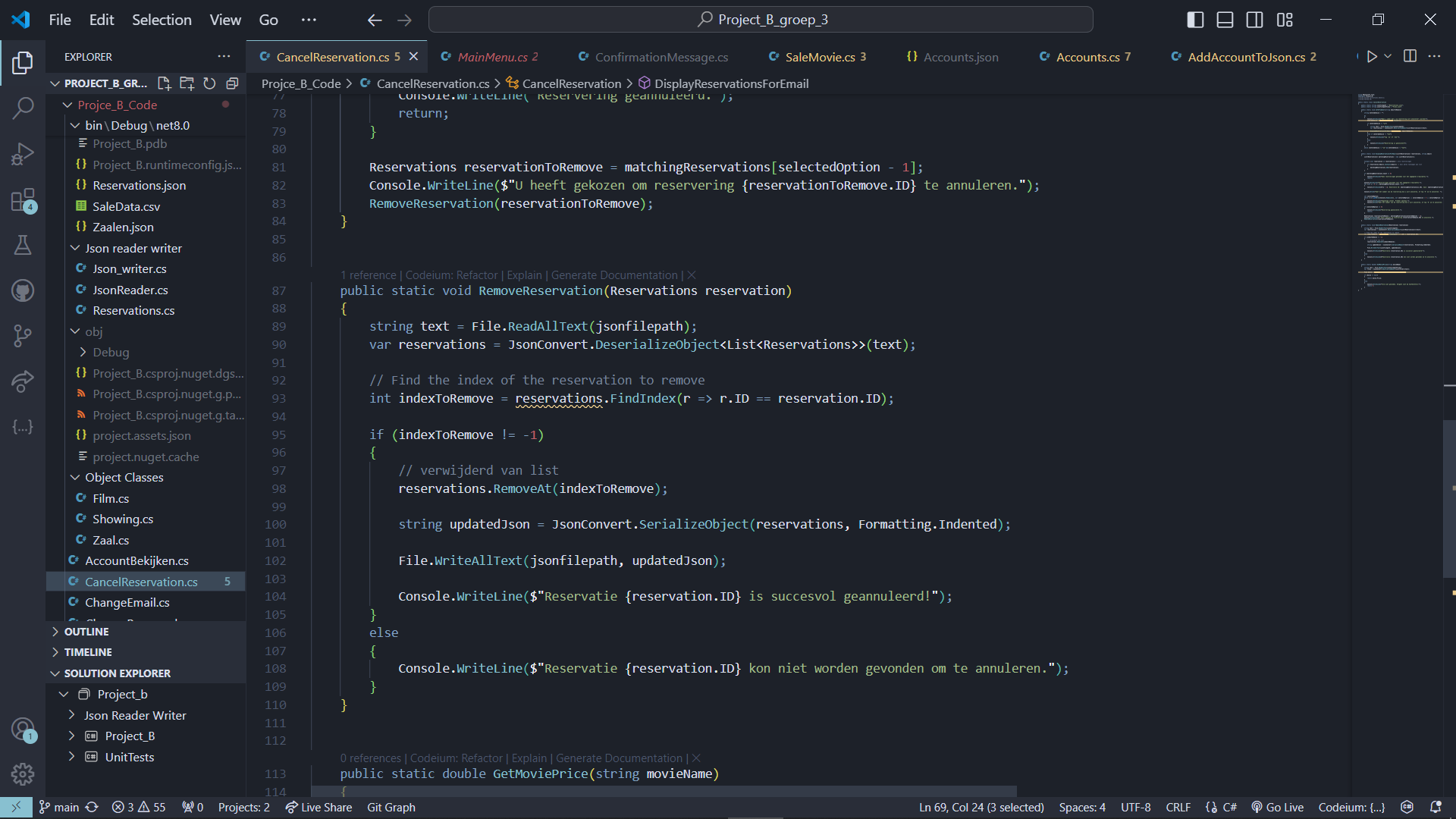
# Static classes: methods, fields, constructors





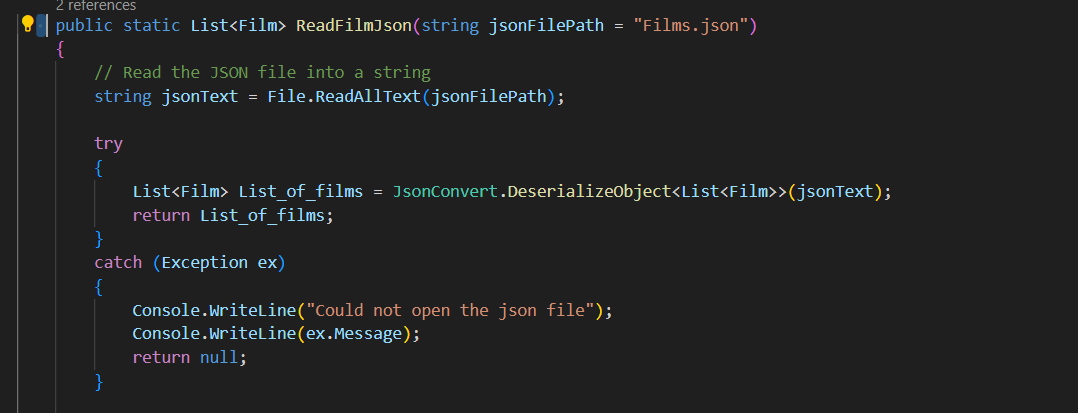
Hier word gebruik gemaakt van een static methods, classes, fields. Er is gekozen voor de static omdat je dan de method kan roepen zonder een object omdat er maar een mail wordt doorgegeven.

# JSON (reading/writing)



Hier wordt de json file geopend en er wordt reservaties getoond van de user, de user geeft dan aan welke reservatie hij wilt verwijder gebaseerd op de reservatie id.

# JSON (exception handling)



Hier zie je dat gebruik gemaakt hebben van een try catch block om te checken of de films in de juiste format staan in de json of of de json file niet bestaat/corrupt is

# Polymorphism overloading

We hebben geen gebruik gemaakt van ploymorphism omdat we geen classes hebben die van andere classes inheriten. We hebben ook geen gebruik gemaakt van overloading omdat we geen methods nodig hadden die met 2 verschillende parameters werken.

# Inheritance

We hebben geen gebruik gamaakt van inheritance omdat we geen classes hadden waarbij het handiger was om van andere classes te inheriten. We hadden bijvoorbeeld geen verschillende soorten users (Admin/superuser).

# Abstraction: interfaces

We hebben geen gebruik gemaakt van interfaces omdat we geen classes hadden die een gezamenlijke functionaliteit hebben. We hadden ook bijvoorbeeld geen IComparable<Film> nodig om film objecten met elkaar te vergelijken omdat we steeds dynamisch wilden aanpassen op wat we wilde sorteren. Daar is LINQ veel handiger voor.

# Abstraction: Abstract classes

We hadden ook geen gebruik gemaakt van abstract classes omdat we geen hiërarchie hadden tussen de classes (objecten). Dit heeft dezelfde reden als waarom we geen inheritance hebben gebruikt.

# Generics

We hebben generics niet gebruikt omdat elke functie zo hebben gemaakt dat hij voor een specifiek type is en we hebben geen list nodig met gemengde types. Ook omdat we geen interfaces/ inheritance/ abstraction hadden gebruikt konden we generics niet meer gebruiken (ook de where keyword niet).