De weekopdracht bestaat uit meerdere, kleine opdrachten. Je begint met een stukje bestaande code:  
  
<https://github.com/itvitaelearning/PillarsOfOOP_WeekOpdracht.git>

Je mag hier een eigen branche op maken en daarmee verder werken. Als je dit doet, geef er dan een duidelijke naam aan zodat we weten welke branche van wie is.

In de solution staat een MVC REST API. Als je deze start krijg je een website met Swagger. Hiermee kun je de actions testen en kijken of je code goed werkt. Je hoeft niet een console applicatie o.i.d. aan te maken. Dit is ook sterk af te raden, omdat dependency injection ingewikkelder werkt met een console app.

Lees per opdracht goed door wat er moet gebeuren. Denk niet te ver na en hou je aan de opdracht.

# Voorbereiding:

1. Zorg ervoor dat de startcode aanwezig is op je eigen systeem. Het project staat in Teams met de naam “WeekOpdrachtDependencyInjection”
2. Open de solution en build het; weet zeker dat alles werkt.
3. Start de API en test de 2 methodes met Swagger

# Opdracht 1: Fix het met dependency injection

1. Maak een interface voor CalculatePiService en koppel deze interface aan de betreffende service (CalculatePiService dus)
2. Pas de PiController aan zodat de nieuwe interface geinjecteerd wordt en gebruikt.  
   Een paar hints
   1. Interface koppelen aan de implementatie doe je in de StartUp
   2. In de controller PiController wordt een private readonly verwacht van de interface.
3. Zorg ervoor dat alles werkt naar behoren. De uitkomst van de nieuwe implementatie mag niet afwijken van de oude implementatie

**Bonus**

1. Maak unit tests met de nieuwe implementatie.

# Opdracht 2: Bird sounds

1. Bekijk de controller ‘BirdController’. Zoals je ziet returned elke action het geluid van een specifieke vogel. Dit is redelijk onhandig en de geluiden horen bij een vogel.
2. Maak een abstracte class ‘Bird’ in de business-laag.  
   Deze abstracte class moet 1 method hebben: “Sound()” en returned een string.
3. Maak nu per vogel (duck, goose en chicken) een class aan die de abstracte class ‘Bird’ inheriten en return het juiste geluid.
4. Pas de controller BirdController zo aan dat de juiste geluiden uit de juiste classes komen.

Opmerking: Dependency injection is hier niet nodig. Het gaat hier puur om de abstractie.

# Opdracht 3: Overloading

In de MovieService zit een methode om een movie per ID op te halen. Er moet een tweede methode bij die een movie ophaalt aan de hand van de titel (let op: 1 movie!). Dit moet gedaan worden met overloading

1. Hernoem de method in MovieService naar ‘Get(int id)’
2. Maak een tweede method met dezelfde naam, maar dan met een paramter string ‘Get(string name)’
3. Implementeer de method Get(string name).
4. Pas de MovieController aan zodat de bestaande functie werkt
5. Voeg een nieuwe action aan waarmee men een movie kan ophalen aan de hand van de naam.

# Bonus opdracht:

## Heb je de bovenstaande opdrachten af? Kijk dan eens naar de volgende opdracht. Iets ingewikkelder en minder informatie. Gebruik voornamelijk Google om de antwoorden te vinden.

## Opdracht 1:

1. Maak een repository aan met entity framework
2. Zorg ervoor dat Movie-class in de database komt met movies
3. Injecteer de repository in de MovieService en maak de methodes werkend
4. Maak unit tests op de MovieService met de repository