1. Create models (almost POCO)
    a. ModelBase
        i. Id, CodeName, DisplayName
    b. Game
        i. Price, StockId
    c. Developer
        i. Multiple games
    d. Awards
        i. Multiple games
    e. GameWrapper
        i. Single (required) developer
        ii. Multiple (optional) awards
    f. Gamer
        i. Same BaseClass
        ii. DifferentContext
2. Create contexts
    a. GamersContext
        i. just Gamer
    b. GameStoreContext
        i. Everything but Gamer
    c. No tests needed as these are actually only EF objects, correct based on EF and working migrations
3. Create migrations project
    a. Can and should be separated from web application (–> console application)
    b. Everybody should use only one set of context, same as web application
    c. Multiple contexts means multiple folders
        i. Update configuration after creation, so individual migrations are grouped in a folder as well
    d. Add a connection string to app.config (named web.config in web.apps (similar to appsettings.json, just diferent format))
        i. Provider for LocalDb is “System.Data.SqlClient” (defined by EF nugget installation script)

```
Enable-Migrations -ContextTypeName GameStore.Entities.Contexts.GameContext -EnableAutomaticMigrations -
MigrationsDirectory GameContextMigrations -ProjectName GameStore.Migrations -StartUpProjectName GameStore.Migrations -
ContextProjectName GameStore.Entities -ConnectionStringName GameStoreConnection -Verbose
```

```
Enable-Migrations -ContextTypeName GameStore.Entities.Contexts.GameStoreContext -EnableAutomaticMigrations -
MigrationsDirectory GameStoreContextMigrations -ProjectName GameStore.Migrations -StartUpProjectName
GameStore.Migrations -ContextProjectName -ConnectionStringName GameStoreConnection GameStore.Entities -Verbose
```

    e. Use PMC to enable migration for individual contexts
        i. Rename Configuration files – <ContextName>Configuration
        ii. Add subfolder \Migrations to each configuration MigrationFolder property
        iii. Either allow AutomaticMigrationDataLossAllowed or disable EnableAutomaticMigrations
    f. Add initial migration for all contexts

```
Add-Migration -Name Initial -ProjectName GameStore.Migrations -StartUpProjectName GameStore.Migrations -
ConfigurationTypeName GameStoreContextConfiguration -ConnectionStringName GameStoreConnection -Verbose
```

```
Add-Migration -Name Initial -ProjectName GameStore.Migrations -StartUpProjectName GameStore.Migrations -
ConfigurationTypeName GamersContextConfiguration -ConnectionStringName GameStoreConnection -Verbose
```

    g. Update (Create) database

```
Update-Database -ConfigurationTypeName GameStoreContextConfiguration -StartUpProjectName GameStore.Migrations -
ProjectName GameStore.Migrations -ConnectionStringName GameStoreConnection -Verbose
```

```
Update-Database -ConfigurationTypeName GamersConfiguration -StartUpProjectName GameStore.Migrations -ProjectName
GameStore.Migrations -ConnectionStringName GameStoreConnection -Verbose
```

    h. Look to SQL Mgmt studio

i. Rename GameWrappers to Games using OnModelCreating override, using modelBuilder.Entity.ToTable
j. Create new migration – only for changed context

```
Add-Migration -Name FixWrapperName -ProjectName GameStore.Migrations -StartUpProjectName GameStore.Migrations -
ConfigurationTypeName GameStoreContextConfiguration -ConnectionStringName GameStoreConnection -Verbose
```

k. Use console application Main method to seed contexts using migration initializer
    i. Do not forget to force initializer to use same connection as context
    ii. Do not forget to provide both contexts with same connection string name
    iii. Do not forget to force initialization for each context (lazily omitted otherwise)
l. Run application (migration should happen)
m. Add some seeding data to both configurations
    i. Use AddOrUpdate, do not forget to specify what to compare on
n. Run application (seeding should happen)

```
Add-Migration -Name DeveloperRequired -ProjectName GameStore.Migrations -StartUpProjectName GameStore.Migrations -
ConfigurationTypeName GameStoreContextConfiguration -ConnectionStringName GameStoreConnection -Verbose
```

o. Make Developer required
p. Add constraint for Unique code names
    i. Not only unique, also max length has to be specified for strings
    ii. Both contexts need migration as all classes implement ModelBase

```
Add-Migration -Name UniqueCodeNameAdded -ProjectName GameStore.Migrations -StartUpProjectName GameStore.Migrations -
ConfigurationTypeName GameStoreContextConfiguration -ConnectionStringName GameStoreConnection -Verbose
```

```
Add-Migration -Name UniqueCodeNameAdded -ProjectName GameStore.Migrations -StartUpProjectName GameStore.Migrations -
ConfigurationTypeName GamersContextConfiguration -ConnectionStringName GameStoreConnection -Verbose
```

q. Run application

4. Create Repositories
    a. GameStoreRepository – supplies basically only games
    b. GameDeveloperRepository – supplies developers
    c. (GamersRepository, …)
    d. Create interfaces, so they can be injected
    e. Created internal constructors with context(s)
        i. pass new instance from default constructor
        ii. default constructor should accept a configuration object with a connection string (provided from upper level)
        iii. default constructor should also accept a Logger and call log fuction of contexts
    f. Add AutoMapper to map between Game and GameWrapper
        i. Test that all methods return correct type
        ii. Test that automapper is correctly configured (no property information is lost)
    g. Faking EF is not completely simple, yet reasonably simple
5. Update GameStockService
    a. Let it require repositories
    b. Adjust method and interface a bit
    c. Add AutoMapper to map between DataTransferObjects (services API) and DataAccessObjects (entities API)
        i. Test that automapper is correctly configured (no property information is lost)
    d. Fix existing tests
6. Wire-up in GameStore.Api
    a. Add transient services for repositories
    b. Add configuration for obtaining connection string (options.ConnectionString = Configuration.GetConnectionString("GameStoreConnection"))
    c. Add same connection string to appsettings.json (provider is not required)

d. Fix existing tests