Exercise 2.2.1

In exercise 2.1.2 you implemented a Reverse Polish calculator. One obvious modification is to use inheritance and delegates to make the calculator easy to extend with new operations. We need the inheritance to generalize the types of operations, e.g. unary, binary, ternary, and the delegates to generalize each type of operations. For the inheritance part you should use this interface

```
public interface IOperation
{
   double Execute(double arg1, params double[] argn);
}
```

We do not know the possible number of arguments in general, and need the open-ended definition we can achieve by use of `params`.

In the calculator you can use the `is` keyword to determine the type of operation (and thus the number of needed arguments)

```
IOperation operation = dictoionary[keyword];

if (operation is UnaryOperation)
```

and then fetch the number of arguments you need on the stack.

You must implement a general version of the Reverse Polish calculator that can handle any unary and binary operations. Here are the requirements:

- The calculator must compute with doubles
- It must support a mapping between keywords and operations, such that new operations can be added to the calculator, by adding a mapping between the operations keyword and a "function" defining how to compute the result of invocation of the operation [Hint: use the Dictionary collection], e.g.

  ```
  "+": new BinaryOperation( (x, y) => x + y)

  "sqrt": new UnaryOperation( (x) => Math.Sqrt(x))
  ```

- Unary and binary operations must be supported
- Implementation of at least the addition, subtraction, multiplication, division, power, sqrt and abs operations
- Unit testing

Exercise 2.2.2

In this exercise, we will work with LINQ and lambda. We will work with a small IMDB fragment:

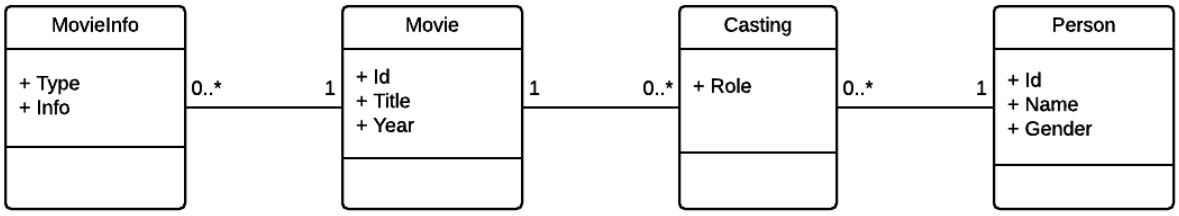
- movie.csv: 1000 movies defined by movieid,title,year
- person.csv: 14709 persons defined by personid,name,gender

- cast.csv: 18102 "castings" defined by personid,movieid,role
- movieinfo.csv: 12494 lines of movie information defined by movieid,type,data

The task is to develop an application to use and manage this database. Today's exercise is the first step; further exercises should be able to build on top of this and extend the application. So take into your considerations a design architecture ready for changes!

The file imdb_data.zip contains the database in CSV.

First part of the exercise is to load data from the CSV files into an object-oriented model for movie, casting, person, and movieinfo, providing a model like the following:



Second part is to create a data service for the model just created. We want an interface to the data service and an implementation using the model above. The interface must provide the following methods:

- Movies – returns all movies
- Persons – returns all persons
- Casting – returns all castings
- AddPerson – adding a new person to the database, using max(personid) + 1 as the new id.

Implement all queries by use of LINQ.

Make a small program that invoke everything:

- Load data from data files
- Create an instance of the "IMDB system".
- Add a new person
- Write a list with the name of the first 5 persons [use LINQ]
- Write the counting of movie by year. Order the output by the descending count [use LINQ]

As usual, unit testing is required.

Optional: Since we have two layers here, you need to test both. To do this a common solution is to make use of mocks/stubs in order to remove dependencies, e.g. the dependency between the Imdb class and the service class. A commonly used tool in C# is the Moq framework, google it to see example usage. To include Moq into your test VS project, right click on the “References” folder and select “Manage NuGet Packages” write “Moq” in the “Search Online” field and install the Moq framework. In the source file, you must add `using Moq;` to get access to the framework.

Exercise 2.2.3

Use the data created in the first part of 2.2.2 to answer the following questions with LINQ:

a) Count number of movies for each year
b) Which movie has the longest title
c) Get the title of all movie with genre “Drama”
d) Find the person casted with most movies
e) How many persons are named “Peter” as first name
f) Find the editor(s) of "The Pink Room" from 2011
g) Show the distribution on editors, actors, actresses, etc. for all movies
h) Find the title of all documentaries
i) Get the titles of all movies with a budget, ordered with the highest budget first