# Exercises: ORM and Entity Framework Overview

Problems for exercises and homework for the "Databases Programming and ORM" course from the official "Applied Programmer" curriculum.

You can check your solutions here: <https://judge.softuni.org/Contests/3150>

(delete all "**bin**"/"**obj**" folders)

## Import the Diablo Database

Import the **Diablo database**. If you already have this database, delete it and import it again. You are given the SQL script for the Diablo database in the resources.

## Install EF SQL Server Tools from NuGet

Use the provided **skeleton** from resources! Do not change its methods, classes and namespaces!

***NOTE***: **If** the installed **versions of the NuGet packages** **are** **newer** than version 3, your **solution** may be marked as **incorrect** when tested in judge.

|  |
| --- |
| Install-Package Microsoft.EntityFrameworkCore.Tools –Version 3.1.3 |
| Install-Package Microsoft.EntityFrameworkCore.SqlServer –Version 3.1.3 |
| Install-Package Microsoft.EntityFrameworkCore.SqlServer.Design |

## Games Information

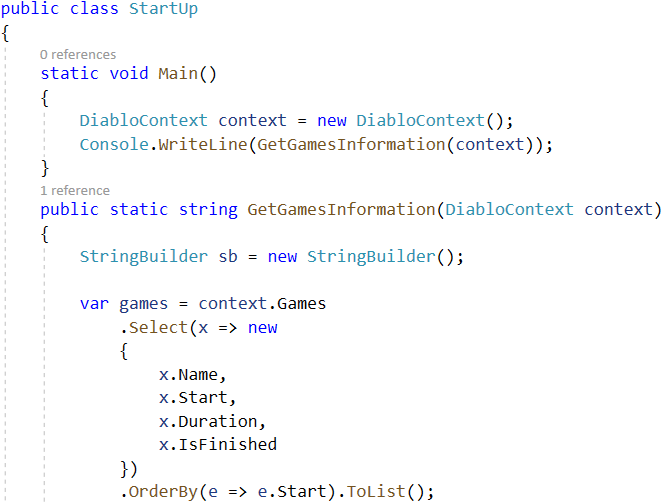
**NOTE**: You will need the method **public static string GetGamesInformation(DiabloContext context)** from the **StartUp** class.

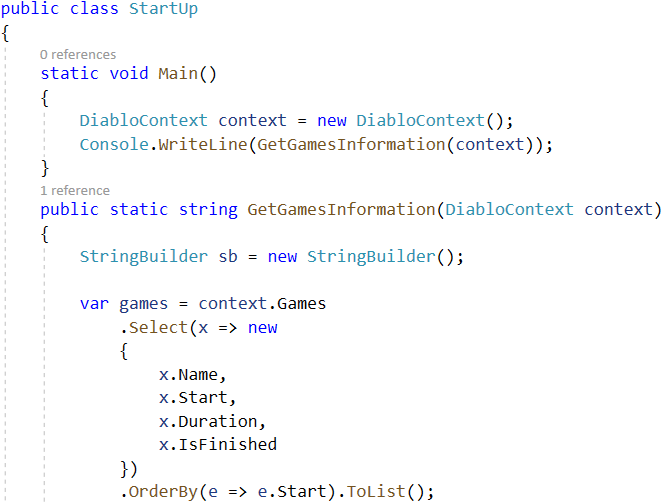
Now we can use the DiabloContextto extract data from Diablo database. Your first task is to extract **all games** andreturn their **Name**, **Start, Duration** and **IsFinished**, all of those separated with a space. Order them by **Start**. If the game is finished write Finished else Unfinished.

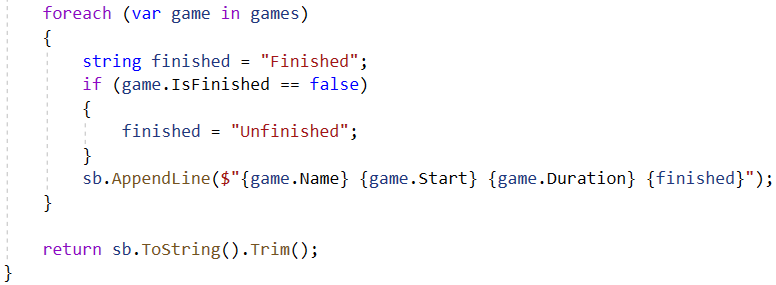
### Example

|  |
| --- |
| Output |
| California pepperberry 06-Jan-10 8:29:00 PM Finished |
| Papyrus lions head 07-Jan-10 5:14:00 PM 7 Unfinished |
| … |

### Hints





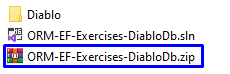


### Submit in Judge

Delete "**bin**"/"**obj**" folders. These are the files in the Tasks-Skeleton folder:



Add the **Diablo** folder and the **.sln** file to a new **.zip** archive.



Submit the **.zip file** to the **Judge** system.

## Items with Price Over 790

**NOTE**: You will need method **public static string GetItemsWithPriceOver790(DiabloContext context)** and **public StartUp** class.

Your task is to extract **all Items** with **Price** over **790**. Return their **names and price** in format **“{Name} - {Price}”.** **Price** must be rounded to **2** **symbols,** after the decimal separator. Sort them **alphabetically** by name.

### Example

|  |
| --- |
| Output |
| Amulets - 792.00 |
| Madstone - 795.00 |
| … |

## Items with Type Axe

**NOTE**: You will need method **public static string GetItemWithTypeAxe(DiabloContext context)** and **public StartUp** class.

Extract all Items with type **Axe** from ItemTypes. Order them by **price** (in ascending order), then by **name** (in descending order). Return only their **Name**, **ItemTypes** and **Price** rounded to **2** **symbols,** after the decimal separator in the format: "{ Name} with type {Item Type} - ${Price}".

### Example

|  |
| --- |
| Output |
| Wands with type Axe - $16.00 |
| Rimeheart with type Axe - $33.00 |
| … |

## Adding a New Game

**NOTE**: You will need method public static string **AddNewGame(DiabloContext context)** and **public StartUp** class.

Create a new game with:

* **Name** – Demo
* **Start** – 2016-02-13 00:00:00.000
* **Duration** – 7
* **GameType** – the game type that has **Id 5**
* **IsFinished** – false

Then order by **descending** all the games by their **Id**, take **10** rows and from them, take the **Name**. Return the results each on a new line:

### Example

|  |
| --- |
| Output |
| Demo |
| Victoria Peak |
| … |

After this **restore** your **database** for the tasks ahead!

### Hints

Use Convert.ToDateTime.

## Users And Games Information

**NOTE**: You will need method **public static string GetUsersAndGamesInformation(DiabloContext context)** and **public StartUp** class.

Find the first **10** users who **joined on** the period **2013 - 2014** (inclusive). Print each employee's **username**, **first name, last name** and **registration date.** Then return **all** of their **games** in the format

"-- Game: {Game Name}, Level: {Level} - {Joined On Date}, Duration: {Duration}",

each on a **new** **row**. If a game has no end date, print Not finished instead.

### Constraints

Use date format: "M/d/yyyy h:mm:ss tt".

### Example

|  |
| --- |
| Output |
| Username:VGeorgiev Names: Vladimir Georgiev - Registration Date: 16-Dec-13 12:00:00 AM |
| -- Game: Misty blue Limonium, Level: 67 - 11/24/2013 12:00:00 AM, Duration: 2 |
| -- Game: Amsterdam, Level: 20 - 5/25/2010 12:00:00 AM, Duration: 7 |
| -- Game: Pompeii, Level: 22 - 3/8/2010 12:00:00 AM, Duration: 2 |
| Username:VGeorgiev Names: Vladimir Georgiev - Registration Date: 16-Dec-13 12:00:00 AM |
| … |

## Users Games

**NOTE**: You will need method **public static string GetUsersGames(DiabloContext context)** and **public StartUp** class.

Find all users, **ordered** by the number of **games** **played** (**descending**), then by **username** (**ascending**), and finally by **first name** (**ascending**). Take only the **first 10 users**. For each user return it in the format:

"{Username}, {Email} - {Games Count} games"

### Example

|  |
| --- |
| Output |
| Pesho, pesho@abv.bg - 10 games |
| rotoriginally, gosyen2000@hotmail.com - 10 games |
| … |

## Users with Games More Than 5

**NOTE**: You will need method **public static string GetUsersWithMoreThan5Games(DiabloContext context)** and **public StartUp** class.

Find **all users** with more than **5 games**. Order them by **games count** (**ascending**), then by **username** (**alphabetically**).   
For each user, print the **username** and the **count of his games**.

Print each user in the format:

"Username: {Username} - Count Games:{Users Games Count}"

### Example

|  |
| --- |
| **Output** |
| Username: baroquegainful - Count Games:6 |
| … |

## Increase Price

**NOTE**: You will need method **public static string IncreasePrice(DiabloContext context)** and **public StartUp** class.

Write a program that increases the **price** by **12%** of all **items** whose statistical **luck** is equal to **18**. Then **return name, speed and price** (2 symbols after the decimal separator)for those items whose price was increased. Order them by **name** (**ascending**), then by **price** (**ascending**). Format of the output.

### Example

|  |
| --- |
| Output |
| Ancestors Grace 7 ($632.80) |
| Band of Untold Secrets 6 ($702.24) |
| Cosmic Strand 6 ($272.16) |
| … |