Exercises: ORM and Entity Framework

You can check your solutions here: https://judge.softuni.bg/Contests/3199/Entity-Framework-Introduction.

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

1. Games Information

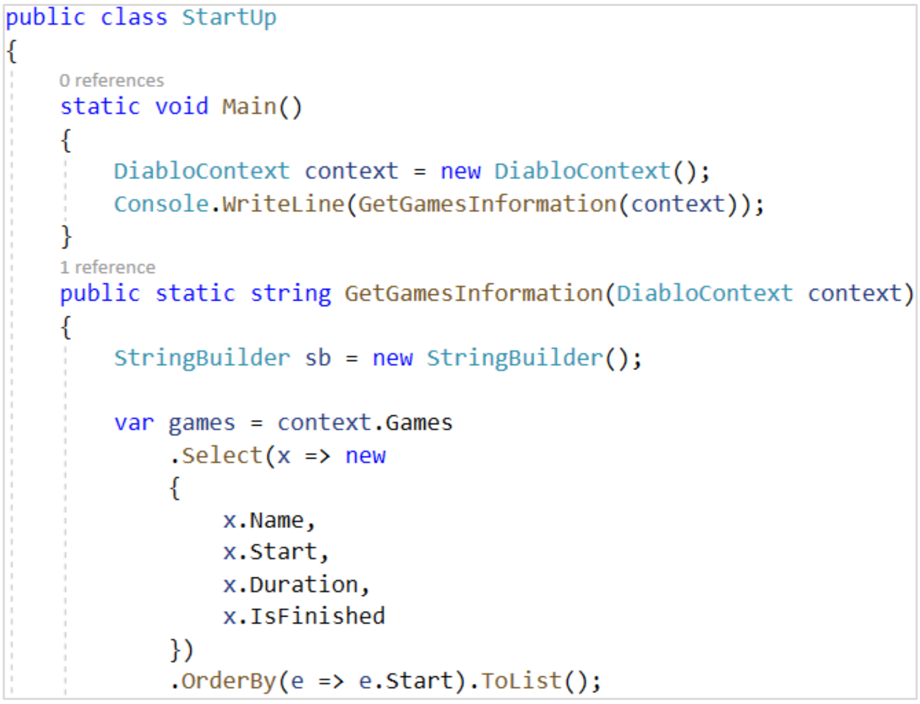
NOTE: You will need method `public static string GetGamesInformation(DiabloContext context)` and `public StartUp` class.

Now we can use the **DiabloContext** to extract data from **Diablo** database. Your first task is to extract **all games** and return 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

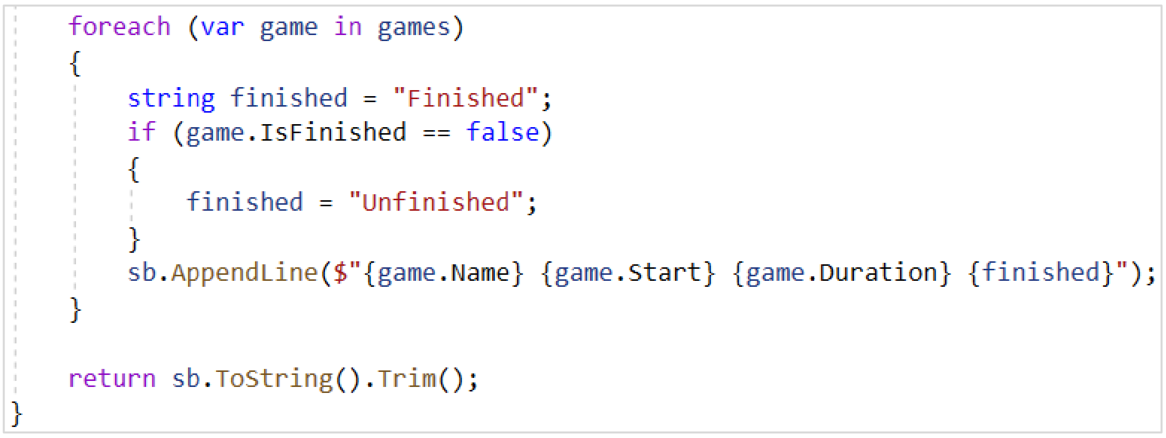
Output
`California pepperberry 06-Jan-10 8:29:00 PM  Finished`
`Papyrus lions head 07-Jan-10 5:14:00 PM 7 Unfinished`
…

Hints:



```
public class StartUp
{
    0 references
    static void Main()
    {
        DiabloContext context = new DiabloContext();
        Console.WriteLine(GetGamesInformation(context));
    }
    1 reference
    public static string GetGamesInformation(DiabloContext context)
    {
        StringBuilder sb = new StringBuilder();

        var games = context.Games
            .Select(x => new
            {
                x.Name,
                x.Start,
                x.Duration,
                x.IsFinished
            })
            .OrderBy(e => e.Start).ToList();
```

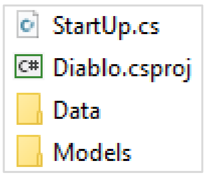


```
    foreach (var game in games)
    {
        string finished = "Finished";
        if (game.IsFinished == false)
        {
            finished = "Unfinished";
        }
        sb.AppendLine($"{game.Name} {game.Start} {game.Duration} {finished}");
    }

    return sb.ToString().Trim();
}
```

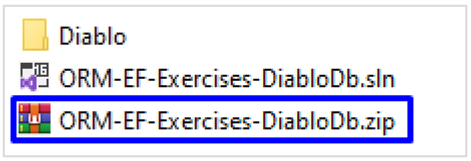
Submit in Judge

Delete "**bin**"/"**obj**" folders. These are the files in the Diablo folder:



StartUp.cs
Diablo.csproj
Data
Models

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



Diablo
ORM-EF-Exercises-DiabloDb.sln
ORM-EF-Exercises-DiabloDb.zip

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

2. 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`
…

3. 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`
…

4. 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`**.

5. 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`
…

6. 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`
…

7. 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**.
Then print the **character name** and the **count of the items** every **game of the user** on a new row.
Order the **games of the use** by **items count** (**ascending**), then by **character** (**ascending**).
Format of the output:

Print each user in the format:

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

```
"Characters:",
```

And for each **game of the user** print it in the format:

```
" - {Character}, Items:{User Game Items Count}".
```

Example

Output
`Username: baroquegainful - Count Games:6`
`Characters:`
` - Sorceress, Items:3`

`- Barbarian, Items:5`
`- Sorceress, Items:5`
`- Monk, Items:9`
`- Sorceress, Items:9`
`- Amazon, Items:11`
…

8. 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)`
…