**НП “ИТ кариера”**

**Модул 07**

**Проект**

**Sports Club**

**Глория Арсова и Виктория Георгиева**

**ПМГ „Акад. Боян Петканчин“ - гр. Хасково**

Хасково 2020г.

Картина, която съдържа екранна снимка, компютър, лаптоп

Описанието е генерирано автоматичноСтруктура

Папка Business

* Class GameBusiness

namespace SportsClub.Business

{

public class GameBusiness

{

private SportsClubContext sportsClubContext;

private string Connection;

private const string defaultConnection = "Server = .\\SQLEXPRESS; Database= SportsClub; Integrated Security=True;";

public GameBusiness(string connection = defaultConnection)

{

Connection = connection;

}

/// <summary>

/// Method "GetAll" returns a list of all games and the information about them

/// </summary>

public List<Game> GetAll()

{

using (sportsClubContext = new SportsClubContext(Connection))

{

return sportsClubContext.Games.ToList();

}

}

/// <summary>

/// Method "Get" returns the game to which the given id matches

/// </summary>

/// <param name="id"></param>

public Game Get(int? id)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

return sportsClubContext.Games.Find(id);

}

}

/// <summary>

/// Method "Add" adds a new game to the database

/// </summary>

/// <param name="game"></param>

public void Add(Game game)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

CheckIfTeamExists(game.FirstTeamId);

CheckIfTeamExists(game.SecondTeamId);

sportsClubContext.Games.Add(game);

sportsClubContext.SaveChanges();

}

}

/// <summary>

/// Method "Update" finds an existing game and changes the information about it

/// </summary>

/// <param name="game"></param>

public void Update(Game game)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

var item = sportsClubContext.Games.Find(game.Id);

if (item != null)

{

CheckIfTeamExists(game.FirstTeamId);

CheckIfTeamExists(game.SecondTeamId);

sportsClubContext.Entry(item).CurrentValues.SetValues(game);

sportsClubContext.SaveChanges();

}

}

}

/// <summary>

/// Method "Delete" finds an existing game to which the given id matches and deletes it

/// </summary>

/// <param name="id"></param>

public void Delete(int id)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

var item = sportsClubContext.Games.Find(id);

if (item != null)

{

try

{

sportsClubContext.Games.Remove(item);

}

catch

{

sportsClubContext.Games.Remove(item);

}

sportsClubContext.SaveChanges();

}

}

}

/// <summary>

/// Method "CheckIfTeamExists" checks if the given id matches any of the existing teams

/// </summary>

/// <param name="teamId"></param>

private void CheckIfTeamExists(int? teamId)

{

bool teamIdExists = false;

foreach (var team in sportsClubContext.Teams)

{

if (team.Id == teamId)

{

teamIdExists = true;

}

}

if (!teamIdExists)

{

throw new ArgumentException("No team with such id");

}

}

}

* Class PlayerBusiness

namespace SportsClub.Business

{

public class PlayerBusiness

{

private SportsClubContext sportsClubContext;

private string Connection;

private const string defaultConnection = "Server = .\\SQLEXPRESS; Database= SportsClub; Integrated Security=True;";

public PlayerBusiness(string connection = defaultConnection)

{

Connection = connection;

}

/// <summary>

/// Method "GetAll" returns a list of all players and the information about them

/// </summary>

public List<Player> GetAll()

{

using (sportsClubContext = new SportsClubContext(Connection))

{

return sportsClubContext.Players.ToList();

}

}

/// <summary>

/// Method "Get" returns the player to which the given id matches

/// </summary>

/// <param name="id"></param>

public Player Get(int? id)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

return sportsClubContext.Players.Find(id);

}

}

/// <summary>

/// Method "Get" returns a list of all players to which the given name matches and the information about them

/// </summary>

/// <param name="name"></param>

/// <returns></returns>

public List<Player> Get(string name)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

List<Player> players = sportsClubContext.Players.ToList();

List<Player> playersWithGivenName = new List<Player>();

foreach (var item in players)

{

if (item.Name == name)

{

playersWithGivenName.Add(item);

}

}

return playersWithGivenName;

}

}

/// <summary>

/// Method "Add" adds a new player to the database

/// </summary>

/// <param name="player"></param>

public void Add(Player player)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

CheckIfTeamExists(player.TeamId);

CheckIfPlayerNameIsCorrect(player);

sportsClubContext.Players.Add(player);

sportsClubContext.SaveChanges();

}

}

/// <summary>

/// Method "Update" finds an existing player and changes the information about him

/// </summary>

/// <param name="player"></param>

public void Update(Player player)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

var item = sportsClubContext.Players.Find(player.Id);

if (item != null)

{

CheckIfTeamExists(player.TeamId);

CheckIfPlayerNameIsCorrect(player);

sportsClubContext.Entry(item).CurrentValues.SetValues(player);

sportsClubContext.SaveChanges();

}

}

}

/// <summary>

/// Method "Delete" finds an existing player to which the given id matches and deletes him

/// </summary>

/// <param name="id"></param>

public void Delete(int id)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

var item = sportsClubContext.Players.Find(id);

if (item != null)

{

try

{

sportsClubContext.Players.Remove(item);

}

catch

{

sportsClubContext.Players.Remove(item);

}

sportsClubContext.SaveChanges();

}

}

}

/// <summary>

/// Method "CheckIfTeamExists" checks if the given id matches any of the existing teams

/// </summary>

/// <param name="teamId"></param>

private void CheckIfTeamExists(int? teamId)

{

bool teamIdExists = false;

foreach (var team in sportsClubContext.Teams)

{

if (team.Id == teamId)

{

teamIdExists = true;

}

}

if (!teamIdExists)

{

throw new ArgumentException("No team with such id");

}

}

/// <summary>

/// Method "CheckIfPlayerNameIsCorrect" checks if the given player name is not "" or null

/// </summary>

/// <param name="player"></param>

public void CheckIfPlayerNameIsCorrect(Player player)

{

if (player.Name == "" || player.Name == null)

{

throw new ArgumentException("Player name can't be empty");

}

}

}

* Class SportBusiness

namespace SportsClub.Business

{

public class SportBusiness

{

private SportsClubContext sportsClubContext;

private string Connection;

private const string defaultConnection = "Server = .\\SQLEXPRESS; Database= SportsClub; Integrated Security=True;";

public SportBusiness(string connection = defaultConnection)

{

Connection = connection;

}

/// <summary>

/// Method "GetAll" returns a list of all sports and the information about them

/// </summary>

public List<Sport> GetAll()

{

using(sportsClubContext = new SportsClubContext(Connection))

{

return sportsClubContext.Sports.ToList();

}

}

/// <summary>

/// Method "Get" returns the sport to which the given id matches

/// </summary>

/// <param name="id"></param>

public Sport Get(int? id)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

return sportsClubContext.Sports.Find(id);

}

}

/// <summary>

/// Method "Get" returns the sport to which the given name matches and the information about them

/// </summary>

/// <param name="name"></param>

/// <returns></returns>

public Sport Get(string name)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

List<Sport> sports = sportsClubContext.Sports.ToList();

foreach (var item in sports)

{

if (item.Name == name)

{

return item;

}

}

return null;

}

}

/// <summary>

/// Method "Add" adds a new sport to the database

/// </summary>

/// <param name="sport"></param>

public void Add(Sport sport)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

CheckIfSportExists(sport);

CheckIfSportNameIsCorrect(sport);

sportsClubContext.Sports.Add(sport);

sportsClubContext.SaveChanges();

}

}

/// <summary>

/// Method "Update" finds an existing sport and changes the information about it

/// </summary>

/// <param name="sport"></param>

public void Update(Sport sport)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

var item = sportsClubContext.Sports.Find(sport.Id);

if (item != null)

{

CheckIfSportExists(sport);

CheckIfSportNameIsCorrect(sport);

sportsClubContext.Entry(item).CurrentValues.SetValues(sport);

sportsClubContext.SaveChanges();

}

}

}

/// <summary>

/// Method "Delete" finds an existing sport to which the given id matches and deletes it

/// </summary>

/// <param name="id"></param>

public void Delete(int id)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

var item = sportsClubContext.Sports.Find(id);

if (item != null)

{

foreach (var team in sportsClubContext.Teams)

{

if (team.SportId == item.Id)

{

TeamBusiness teamBusiness = new TeamBusiness();

teamBusiness.Delete(team.Id);

}

}

try

{

sportsClubContext.Sports.Remove(item);

}

catch

{

sportsClubContext.Sports.Remove(item);

}

sportsClubContext.SaveChanges();

}

}

}

/// <summary>

/// Method "CheckIfSportExists" checks if the given name matches any of the existing sports

/// </summary>

/// <param name="sport"></param>

private void CheckIfSportExists(Sport sport)

{

foreach (var existentSport in sportsClubContext.Sports)

{

if (sport.Name == existentSport.Name)

{

throw new ArgumentException("Sport already exists");

}

}

}

/// <summary>

/// Method "CheckIfSportNameIsCorrect" checks if the given sport name is not "" or null

/// </summary>

/// <param name="sport"></param>

public void CheckIfSportNameIsCorrect(Sport sport)

{

if (sport.Name == "" || sport.Name == null)

{

throw new ArgumentException("Sport name can't be empty");

}

}

}

* Class TeamBusiness

namespace SportsClub.Business

{

public class TeamBusiness

{

private SportsClubContext sportsClubContext;

private string Connection;

private const string defaultConnection = "Server = .\\SQLEXPRESS; Database= SportsClub; Integrated Security=True;";

public TeamBusiness(string connection = defaultConnection)

{

Connection = connection;

}

/// <summary>

/// Method "GetAll" returns a list of all teams and the information about them

/// </summary>

public List<Team> GetAll()

{

using (sportsClubContext = new SportsClubContext(Connection))

{

return sportsClubContext.Teams.ToList();

}

}

/// <summary>

/// Method "Get" returns the team to which the given id matches

/// </summary>

/// <param name="id"></param>

public Team Get(int? id)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

return sportsClubContext.Teams.Find(id);

}

}

/// <summary>

/// Method "Get" returns a list of all teams to which the given name matches and the information about them

/// </summary>

/// <param name="name"></param>

/// <returns></returns>

public List<Team> Get(string name)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

List<Team> teams = sportsClubContext.Teams.ToList();

List<Team> teamsWithGivenName = new List<Team>();

foreach (var item in teams)

{

if (item.Name == name)

{

teamsWithGivenName.Add(item);

}

}

return teamsWithGivenName;

}

}

/// <summary>

/// Method "Add" adds a new team to the database

/// </summary>

/// <param name="team"></param>

public void Add(Team team)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

CheckIfSportExists(team.SportId);

CheckIfTrainerExists(team.TrainerId);

CheckIfTeamNameIsCorrect(team);

CheckIfTeamExists(team);

sportsClubContext.Teams.Add(team);

sportsClubContext.SaveChanges();

}

}

/// <summary>

/// Method "Update" finds an existing team and changes the information about it

/// </summary>

/// <param name="team"></param>

public void Update(Team team)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

var item = sportsClubContext.Teams.Find(team.Id);

if (item != null)

{

CheckIfSportExists(team.SportId);

CheckIfTrainerExists(team.TrainerId);

CheckIfTeamNameIsCorrect(team);

CheckIfTeamExists(team);

sportsClubContext.Entry(item).CurrentValues.SetValues(team);

sportsClubContext.SaveChanges();

}

}

}

/// <summary>

/// Method "Delete" finds an existing team to which the given id matches and deletes it

/// </summary>

/// <param name="id"></param>

public void Delete(int id)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

var item = sportsClubContext.Teams.Find(id);

if (item != null)

{

foreach (var player in sportsClubContext.Players)

{

if (player.TeamId == item.Id)

{

sportsClubContext.Players.Remove(player);

}

}

foreach (var game in sportsClubContext.Games)

{

if (game.FirstTeamId == item.Id || game.SecondTeamId == item.Id)

{

sportsClubContext.Games.Remove(game);

}

}

try

{

sportsClubContext.Teams.Remove(item);

}

catch

{

sportsClubContext.Teams.Remove(item);

}

sportsClubContext.SaveChanges();

}

}

}

/// <summary>

/// Method "CheckIfTeamExists" checks if the given team matches any of the existing teams

/// </summary>

/// <param name="team"></param>

private void CheckIfTeamExists(Team team)

{

bool teamExists = false;

foreach (var existingTeam in sportsClubContext.Teams)

{

if (existingTeam.Name == team.Name &&

existingTeam.SportId == team.SportId &&

existingTeam.TrainerId == team.TrainerId)

{

teamExists = true;

}

}

if (teamExists)

{

throw new ArgumentException("Team already exists");

}

}

/// <summary>

/// Method "CheckIfSportExists" checks if the given id matches any of the existing sports

/// </summary>

/// <param name="sportId"></param>

private void CheckIfSportExists(int? sportId)

{

bool sportIdExists = false;

foreach (var sport in sportsClubContext.Sports)

{

if (sport.Id == sportId)

{

sportIdExists = true;

}

}

if (!sportIdExists)

{

throw new ArgumentException("No sport with such id");

}

}

/// <summary>

/// Method "CheckIfTrainerExists" checks if the given id matches any of the existing trainers

/// </summary>

/// <param name="trainerId"></param>

private void CheckIfTrainerExists(int? trainerId)

{

bool trainerIdExists = false;

foreach (var trainer in sportsClubContext.Trainers)

{

if (trainer.Id == trainerId)

{

trainerIdExists = true;

}

}

if (!trainerIdExists)

{

throw new ArgumentException("No trainer with such id");

}

}

/// <summary>

/// Method "CheckIfTeamNameIsCorrect" checks if the given team name is not "" or null

/// </summary>

/// <param name="team"></param>

public void CheckIfTeamNameIsCorrect(Team team)

{

if (team.Name == "" || team.Name == null)

{

throw new ArgumentException("Team name can't be empty");

}

}

}

* Class TrainerBusiness

namespace SportsClub.Business

{

public class TrainerBusiness

{

private SportsClubContext sportsClubContext;

private string Connection;

private const string defaultConnection = "Server = .\\SQLEXPRESS; Database= SportsClub; Integrated Security=True;";

public TrainerBusiness(string connection = defaultConnection)

{

Connection = connection;

}

/// <summary>

/// Method "GetAll" returns a list of all trainers and the information about them

/// </summary>

public List<Trainer> GetAll()

{

using (sportsClubContext = new SportsClubContext(Connection))

{

return sportsClubContext.Trainers.ToList();

}

}

/// <summary>

/// Method "Get" returns the trainer to which the given id matches

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public Trainer Get(int? id)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

return sportsClubContext.Trainers.Find(id);

}

}

/// <summary>

/// Method "Get" returns a list of all trainers to which the given name matches and the information about them

/// </summary>

/// <param name="name"></param>

/// <returns></returns>

public List<Trainer> Get(string name)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

List<Trainer> trainers = sportsClubContext.Trainers.ToList();

List<Trainer> trainersWithGivenName = new List<Trainer>();

foreach (var item in trainers)

{

if (item.Name == name)

{

trainersWithGivenName.Add(item);

}

}

return trainersWithGivenName;

}

}

/// <summary>

/// Method "Add" adds a new trainer to the database

/// </summary>

/// <param name="trainer"></param>

public void Add(Trainer trainer)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

CheckIfTrainerNameIsCorrect(trainer);

sportsClubContext.Trainers.Add(trainer);

sportsClubContext.SaveChanges();

}

}

/// <summary>

/// Method "Update" finds an existing trainer and changes the information about him

/// </summary>

/// <param name="trainer"></param>

public void Update(Trainer trainer)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

var item = sportsClubContext.Trainers.Find(trainer.Id);

if (item != null)

{

CheckIfTrainerNameIsCorrect(trainer);

sportsClubContext.Entry(item).CurrentValues.SetValues(trainer);

sportsClubContext.SaveChanges();

}

}

}

/// <summary>

/// Method "Delete" finds an existing trainer to which the given id matches and deletes him

/// </summary>

/// <param name="id"></param>

public void Delete(int id)

{

using (sportsClubContext = new SportsClubContext(Connection))

{

var item = sportsClubContext.Trainers.Find(id);

if (item != null)

{

foreach (var team in sportsClubContext.Teams)

{

if(team.TrainerId==item.Id)

{

TeamBusiness teamBusiness = new TeamBusiness();

teamBusiness.Delete(team.Id);

}

}

try

{

sportsClubContext.Trainers.Remove(item);

}

catch

{

sportsClubContext.Trainers.Remove(item);

}

sportsClubContext.SaveChanges();

}

}

}

/// <summary>

/// Method "CheckIfTrainerNameIsCorrect" checks if the given trainer name is not "" or null

/// </summary>

/// <param name="trainer"></param>

public void CheckIfTrainerNameIsCorrect(Trainer trainer)

{

if (trainer.Name == "" || trainer.Name == null)

{

throw new ArgumentException("Trainer name can't be empty");

}

}

}

Папка Data

* + Class SportsClubContext

namespace SportsClub.Data

{

public partial class SportsClubContext : DbContext

{

private string Connection;

private const string defaultConnection = "Server = .\\SQLEXPRESS; Database= SportsClub; Integrated Security=True;";

public SportsClubContext(string connection = defaultConnection)

{

Connection = connection;

}

public SportsClubContext(DbContextOptions<SportsClubContext> options)

: base(options)

{

}

public virtual DbSet<Game> Games { get; set; }

public virtual DbSet<Player> Players { get; set; }

public virtual DbSet<Sport> Sports { get; set; }

public virtual DbSet<Team> Teams { get; set; }

public virtual DbSet<Trainer> Trainers { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

optionsBuilder.UseSqlServer(Connection);

}

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Game>(entity =>

{

entity.HasOne(d => d.FirstTeam)

.WithMany(p => p.GameFirstTeam)

.HasForeignKey(d => d.FirstTeamId)

.HasConstraintName("FK\_\_Game\_\_FirstTeamI\_\_6FE99F9F");

entity.HasOne(d => d.SecondTeam)

.WithMany(p => p.GameSecondTeam)

.HasForeignKey(d => d.SecondTeamId)

.HasConstraintName("FK\_\_Game\_\_SecondTeam\_\_70DDC3D8");

});

modelBuilder.Entity<Player>(entity =>

{

entity.Property(e => e.Name)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

entity.HasOne(d => d.Team)

.WithMany(p => p.Player)

.HasForeignKey(d => d.TeamId)

.HasConstraintName("FK\_\_Player\_\_TeamId\_\_47DBAE45");

});

modelBuilder.Entity<Sport>(entity =>

{

entity.Property(e => e.Name)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

});

modelBuilder.Entity<Team>(entity =>

{

entity.Property(e => e.Name)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

entity.HasOne(d => d.Sport)

.WithMany(p => p.Team)

.HasForeignKey(d => d.SportId)

.HasConstraintName("FK\_\_Team\_\_SportId\_\_3E52440B");

entity.HasOne(d => d.Trainer)

.WithMany(p => p.Team)

.HasForeignKey(d => d.TrainerId)

.HasConstraintName("FK\_\_Team\_\_TrainerId\_\_3F466844");

});

modelBuilder.Entity<Trainer>(entity =>

{

entity.Property(e => e.Name)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

});

OnModelCreatingPartial(modelBuilder);

}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);

}

* Папка Models
  + Class Game

namespace SportsClub.Data.Models

{

/// <summary>

/// In class "Game" are implemented the properties of the game

/// </summary>

public partial class Game

{

public Game(int? firstTeamId = null, int? secondTeamId = null)

{

FirstTeamId = firstTeamId;

SecondTeamId = secondTeamId;

}

public int Id { get; set; }

public int? FirstTeamId { get; set; }

public int? SecondTeamId { get; set; }

public virtual Team FirstTeam { get; set; }

public virtual Team SecondTeam { get; set; }

}

}

* + Class Player

namespace SportsClub.Data.Models

{

/// <summary>

/// In class "Player" are implemented the properties of the player

/// </summary>

public partial class Player

{

public Player(string name = null, int? age = null, int? teamId = null)

{

Name = name;

Age = age;

TeamId = teamId;

}

public int Id { get; set; }

public string Name { get; set; }

public int? Age { get; set; }

public int? TeamId { get; set; }

public virtual Team Team { get; set; }

}

}

* + Class Sport

namespace SportsClub.Data.Models

{

/// <summary>

/// In class "Sport" are implemented the properties and the consctructor of the sport

/// </summary>

public partial class Sport

{

public Sport(string name = null)

{

Team = new HashSet<Team>();

Name = name;

}

public int Id { get; set; }

public string Name { get; set; }

public virtual ICollection<Team> Team { get; set; }

}

}

* + Class Team

namespace SportsClub.Data.Models

{

/// <summary>

/// In class "Team" are implemented the properties and the constructor of the team

/// </summary>

public partial class Team

{

public Team(string name = null, int? sportId = null, int? trainerId = null)

{

GameFirstTeam = new HashSet<Game>();

GameSecondTeam = new HashSet<Game>();

Player = new HashSet<Player>();

Name = name;

SportId = sportId;

TrainerId = trainerId;

}

public int Id { get; set; }

public string Name { get; set; }

public int? SportId { get; set; }

public int? TrainerId { get; set; }

public virtual Sport Sport { get; set; }

public virtual Trainer Trainer { get; set; }

public virtual ICollection<Game> GameFirstTeam { get; set; }

public virtual ICollection<Game> GameSecondTeam { get; set; }

public virtual ICollection<Player> Player { get; set; }

}

}

* + Class Trainer

namespace SportsClub.Data.Models

{

/// <summary>

/// In class "Trainer" are implemented the properties and the constructor of the trainer

/// </summary>

public partial class Trainer

{

public Trainer(string name = null)

{

Team = new HashSet<Team>();

Name = name;

}

public int Id { get; set; }

public string Name { get; set; }

public virtual ICollection<Team> Team { get; set; }

}

}

Папка Presentation

* + Class Display

namespace SportsClub.Presentation

{

/// <summary>

/// In class "Display" are implemented the Menu and the methods, by which the entered information is taken

/// </summary>

public static class Display

{

public static int actionExitOperation = 7;

private static int tablesExitOperation = 6;

private static PlayerDisplay playerDisplay;

private static TeamDisplay teamDisplay;

private static TrainerDisplay trainerDisplay;

private static SportDisplay sportDisplay;

private static GameDisplay gameDisplay;

/// <summary>

/// Method "GetInput" allows us to use the functions of the table we choose

/// </summary>

public static void GetInput()

{

var operation = -1;

do

{

ShowTableNames();

try

{

operation = GetIntNumber(operation); //gets the number which we entered

switch (operation)

{

case 1:

{

playerDisplay = new PlayerDisplay();

break;

}

case 2:

{

teamDisplay = new TeamDisplay();

break;

}

case 3:

{

trainerDisplay = new TrainerDisplay();

break;

}

case 4:

{

sportDisplay = new SportDisplay();

break;

}

case 5:

{

gameDisplay = new GameDisplay();

break;

}

default:

{

if (operation != tablesExitOperation) //checks if the number which we eneterd in not in the menu

{

Console.WriteLine("Not an operation number");

GetBackToMenu();

}

break;

}

}

}

catch (ArgumentException exception)

{

Console.WriteLine(exception.Message);

GetBackToMenu();

}

} while (operation != tablesExitOperation);

}

/// <summary>

/// Method "ShowActions" shows a list of action to choose from

/// </summary>

public static void ShowActions()

{

Console.WriteLine("Choose action:");

Console.WriteLine("1. List all entries");

Console.WriteLine("2. Add new entry");

Console.WriteLine("3. Update entry");

Console.WriteLine("4. Fetch entry by ID");

Console.WriteLine("5. Delete entry by ID");

Console.WriteLine("6. Fetch entry by name");

Console.WriteLine("7. Exit");

Console.Write("Enter operation number: ");

}

/// <summary>

/// Method "ShowTableNames" shows a list of the tables names to choose from

/// </summary>

private static void ShowTableNames()

{

Console.Clear();

Console.WriteLine("-- Sports Club --");

Console.WriteLine("Choose table:");

Console.WriteLine("1. Players");

Console.WriteLine("2. Teams");

Console.WriteLine("3. Trainers");

Console.WriteLine("4. Sports");

Console.WriteLine("5. Games");

Console.WriteLine("6. Exit");

Console.Write("Enter operation number: ");

}

/// <summary>

/// Method "GetIntNumber" returns the number which we entered

/// </summary>

/// <param name="number"></param>

/// <returns></returns>

public static int GetIntNumber(int number)

{

try

{

number = int.Parse(Console.ReadLine());

return number;

}

catch

{

throw new ArgumentException("This should be written with numbers only");

}

}

/// <summary>

/// Method "GetIntNumber" returns the number which we entered

/// </summary>

/// <param name="number"></param>

/// <returns></returns>

public static int? GetIntNumber(int? number)

{

try

{

number = int.Parse(Console.ReadLine());

return number;

}

catch

{

throw new ArgumentException("This should be written with numbers only");

}

}

/// <summary>

/// Method "GetBackToMenu" bring us back to the Menu

/// </summary>

public static void GetBackToMenu()

{

Console.WriteLine();

Console.WriteLine("Press enter to go back...");

Console.ReadLine();

}

}

}

* + Class GameDisplay

namespace SportsClub.Presentation

{

/// <summary>

/// In class "GameDisplay" are implemented the methods, which describe how the game information is displayed

/// </summary>

public class GameDisplay

{

private TeamBusiness teamBusiness = new TeamBusiness();

private GameBusiness gameBusiness = new GameBusiness();

/// <summary>

/// Method "GameDisplay" allows us to use the functions we choose

/// </summary>

public GameDisplay()

{

var operation = -1;

do

{

Console.Clear();

Console.WriteLine("-- Games --");

Display.ShowActions();

try

{

operation = Display.GetIntNumber(operation); //gets the number which we entered

Console.Clear();

switch (operation)

{

case 1:

{

ListAll();

Display.GetBackToMenu();

break;

}

case 2:

{

Add();

Display.GetBackToMenu();

break;

}

case 3:

{

Update();

Display.GetBackToMenu();

break;

}

case 4:

{

FetchById();

Display.GetBackToMenu();

break;

}

case 5:

{

Delete();

Display.GetBackToMenu();

break;

}

case 6:

{

FetchByName();

Display.GetBackToMenu();

break;

}

default:

{

if (operation != Display.actionExitOperation)//checks if the number which we eneterd in not in the menu

{

Console.WriteLine("Not an operation number");

Display.GetBackToMenu();

}

break;

}

}

}

catch (ArgumentException exception)

{

Console.WriteLine(exception.Message);

Display.GetBackToMenu();

}

} while (operation != Display.actionExitOperation);

}

/// <summary>

/// Method "ListAll" shows a list of all games and the information about them

/// </summary>

private void ListAll()

{

Console.WriteLine("Games:");

List<Game> games = gameBusiness.GetAll();

if (games.Count != 0)

{

Console.WriteLine();

var table = new ConsoleTable("Id", "Match");

foreach (var item in games)

{

var team1 = teamBusiness.Get(item.FirstTeamId);

var team2 = teamBusiness.Get(item.SecondTeamId);

table.AddRow(item.Id, $"{team1.Name} vs {team2.Name}");

}

table.Write();

}

else

{

Console.WriteLine("No games");

}

}

/// <summary>

/// Method "Add" adds a new game to the database

/// </summary>

private void Add()

{

Game game = new Game();

Console.WriteLine("Enter First Team Id:");

game.FirstTeamId = Display.GetIntNumber(game.FirstTeamId);

Console.WriteLine("Enter Second Team Id:");

game.SecondTeamId = Display.GetIntNumber(game.SecondTeamId);

gameBusiness.Add(game);

}

/// <summary>

/// Method "Update" finds an existing game and changes the information about it

/// </summary>

private void Update()

{

Console.WriteLine("Enter Id:");

int id = 0;

id = Display.GetIntNumber(id);

Game game = gameBusiness.Get(id);

if (game != null)

{

Console.WriteLine("Enter new first team id:");

game.FirstTeamId = Display.GetIntNumber(game.FirstTeamId);

Console.WriteLine("Enter new second team id:");

game.SecondTeamId = Display.GetIntNumber(game.SecondTeamId);

gameBusiness.Update(game);

}

else

{

Console.WriteLine("Game not found");

}

}

/// <summary>

/// Method "FetchById" finds an existing game to which the given id matches and shows the information about it

/// </summary>

private void FetchById()

{

Console.WriteLine("Enter id:");

int id = 0;

id = Display.GetIntNumber(id);

Game game = gameBusiness.Get(id);

if (game != null)

{

Console.WriteLine();

Console.WriteLine($"Id: {game.Id}");

var team1 = teamBusiness.Get(game.FirstTeamId);

var team2 = teamBusiness.Get(game.SecondTeamId);

Console.WriteLine($"Match: {team1.Name} vs {team2.Name}");

}

else

{

Console.WriteLine("Game not found");

}

}

/// <summary>

/// Method "FetchByName" finds an existing game to which the given name matches and shows the information about it

/// </summary>

private void FetchByName()

{

Console.WriteLine("Games don't have names");

}

/// <summary>

/// Method "Delete" finds an existing game and deletes it

/// </summary>

private void Delete()

{

Console.WriteLine("Enter id:");

int id = 0;

id = Display.GetIntNumber(id);

gameBusiness.Delete(id);

}

}

}

* + Class PlayerDisplay

namespace SportsClub.Presentation

{

/// <summary>

/// In class "PlayerDisplay" are implemented the methods, which describe how the player information is displayed

/// </summary>

public class PlayerDisplay

{

private PlayerBusiness playerBusiness = new PlayerBusiness();

private TeamBusiness teamBusiness = new TeamBusiness();

/// <summary>

/// Method "PlayerDisplay" allows us to use the functions we choose

/// </summary>

public PlayerDisplay()

{

var operation = -1;

do

{

Console.Clear();

Console.WriteLine("-- Players --");

Display.ShowActions();

try

{

operation = Display.GetIntNumber(operation);//gets the number which we entered

Console.Clear();

switch (operation)

{

case 1:

{

ListAll();

Display.GetBackToMenu();

break;

}

case 2:

{

Add();

Display.GetBackToMenu();

break;

}

case 3:

{

Update();

Display.GetBackToMenu();

break;

}

case 4:

{

FetchById();

Display.GetBackToMenu();

break;

}

case 5:

{

Delete();

Display.GetBackToMenu();

break;

}

case 6:

{

FetchByName();

Display.GetBackToMenu();

break;

}

default:

{

if (operation != Display.actionExitOperation)//checks if the number which we eneterd in not in the menu

{

Console.WriteLine("Not an operation number");

Display.GetBackToMenu();

}

break;

}

}

}

catch (ArgumentException exception)

{

Console.WriteLine(exception.Message);

Display.GetBackToMenu();

}

} while (operation != Display.actionExitOperation);

}

/// <summary>

/// Method "ListAll" shows a list of all players and the information about them

/// </summary>

private void ListAll()

{

Console.WriteLine("Players:");

List<Player> players = playerBusiness.GetAll();

if (players.Count != 0)

{

Console.WriteLine();

var table = new ConsoleTable("Id", "Name", "Age", "Team");

foreach (var item in players)

{

var team = teamBusiness.Get(item.TeamId);

table.AddRow(item.Id, item.Name, item.Age, team.Name);

}

table.Write();

}

else

{

Console.WriteLine("No players");

}

}

/// <summary>

/// Method "Add" adds a new player to the database

/// </summary>

private void Add()

{

Player player = new Player();

Console.WriteLine("Enter Name:");

player.Name = Console.ReadLine();

Console.WriteLine("Enter Age:");

player.Age = Display.GetIntNumber(player.Age);

Console.WriteLine("Enter TeamId:");

player.TeamId = Display.GetIntNumber(player.TeamId);

playerBusiness.Add(player);

}

/// <summary>

/// Method "Update" finds an existing player and changes the information about him

/// </summary>

private void Update()

{

Console.WriteLine("Enter Id:");

int id = 0;

id = Display.GetIntNumber(id);

Player player = playerBusiness.Get(id);

if (player != null)

{

Console.WriteLine("Enter new name:");

player.Name = Console.ReadLine();

Console.WriteLine("Enter new age:");

player.Age = Display.GetIntNumber(player.Age);

Console.WriteLine("Enter new team id:");

player.TeamId = Display.GetIntNumber(player.TeamId);

playerBusiness.Update(player);

}

else

{

Console.WriteLine("Player not found");

}

}

/// <summary>

/// Method "FetchById" finds an existing player to which the given id matches and shows the information about him

/// </summary>

private void FetchById()

{

Console.WriteLine("Enter id:");

int id = 0;

id = Display.GetIntNumber(id);

Player player = playerBusiness.Get(id);

if (player != null)

{

Console.WriteLine($"Id: {player.Id}");

Console.WriteLine($"Name: {player.Name}");

Console.WriteLine($"Age: {player.Age}");

var team = teamBusiness.Get(player.TeamId);

Console.WriteLine($"Team: {team.Name}");

}

else

{

Console.WriteLine("Player not found");

}

}

/// <summary>

/// Method "FetchByName" finds an existing player to which the given name matches and shows the information about him

/// </summary>

private void FetchByName()

{

Console.WriteLine("Enter name:");

string name = Console.ReadLine();

List<Player> players = playerBusiness.Get(name);

if (players.Count != 0)

{

Console.WriteLine();

var table = new ConsoleTable("Id", "Name", "Age", "Team");

foreach (var item in players)

{

var team = teamBusiness.Get(item.TeamId);

table.AddRow(item.Id, item.Name, item.Age, team.Name);

}

table.Write();

}

else

{

Console.WriteLine("No players with such name");

}

}

/// <summary>

/// Method "Delete" finds an existing player and deletes him

/// </summary>

private void Delete()

{

Console.WriteLine("Enter id:");

int id = 0;

id = Display.GetIntNumber(id);

playerBusiness.Delete(id);

}

}

}

* + Class SportDisplay

using System.Text;

namespace SportsClub.Presentation

{

/// <summary>

/// In class "SportDisplay" are implemented the methods, which describe how the sport information is displayed

/// </summary>

public class SportDisplay

{

private SportBusiness sportBusiness = new SportBusiness();

/// <summary>

/// Method "SportDisplay" allows us to use the functions we choose

/// </summary>

public SportDisplay()

{

var operation = -1;

do

{

Console.Clear();

Console.WriteLine("-- Sports --");

Display.ShowActions();

try

{

operation = Display.GetIntNumber(operation);//gets the number which we entered

Console.Clear();

switch (operation)

{

case 1:

{

ListAll();

Display.GetBackToMenu();

break;

}

case 2:

{

Add();

Display.GetBackToMenu();

break;

}

case 3:

{

Update();

Display.GetBackToMenu();

break;

}

case 4:

{

FetchById();

Display.GetBackToMenu();

break;

}

case 5:

{

Delete();

Display.GetBackToMenu();

break;

}

case 6:

{

FetchByName();

Display.GetBackToMenu();

break;

}

default:

{

if (operation != Display.actionExitOperation)//checks if the number which we eneterd in not in the menu

{

Console.WriteLine("Not an operation number");

Display.GetBackToMenu();

}

break;

}

}

}

catch (ArgumentException exception)

{

Console.WriteLine(exception.Message);

Display.GetBackToMenu();

}

} while (operation != Display.actionExitOperation);

}

/// <summary>

/// Method "ListAll" shows a list of all sports and the information about them

/// </summary>

private void ListAll()

{

Console.WriteLine("Sports:");

List<Sport> sports = sportBusiness.GetAll();

if (sports.Count != 0)

{

Console.WriteLine();

var table = new ConsoleTable("Id", "Name");

foreach (var item in sports)

{

table.AddRow(item.Id, item.Name);

}

table.Write();

}

else

{

Console.WriteLine("No sports");

}

}

/// <summary>

/// Method "Add" adds a new sport to the database

/// </summary>

private void Add()

{

Sport sport = new Sport();

Console.WriteLine("Enter Name:");

sport.Name = Console.ReadLine();

sportBusiness.Add(sport);

}

/// <summary>

/// Method "Update" finds an existing sport and changes the information about it

/// </summary>

private void Update()

{

Console.WriteLine("Enter Id:");

int id = 0;

id = Display.GetIntNumber(id);

Sport sport = sportBusiness.Get(id);

if (sport != null)

{

Console.WriteLine("Enter new name:");

sport.Name = Console.ReadLine();

sportBusiness.Update(sport);

}

else

{

Console.WriteLine("Sport not found");

}

}

/// <summary>

/// Method "FetchById" finds an existing sport to which the given id matches and shows the information about it

/// </summary>

private void FetchById()

{

Console.WriteLine("Enter id:");

int id = 0;

id = Display.GetIntNumber(id);

Sport sport = sportBusiness.Get(id);

if (sport != null)

{

Console.WriteLine($"Id: {sport.Id}");

Console.WriteLine($"Name: {sport.Name}");

}

else

{

Console.WriteLine("Sport not found");

}

}

/// <summary>

/// Method "FetchByName" finds an existing sport to which the given name matches and shows the information about it

/// </summary>

private void FetchByName()

{

Console.WriteLine("Enter name:");

string name = Console.ReadLine();

Sport sport = sportBusiness.Get(name);

if (sport != null)

{

Console.WriteLine();

var table = new ConsoleTable("Id", "Name");

table.AddRow(sport.Id, sport.Name);

table.Write();

}

else

{

Console.WriteLine("No such sport");

}

}

/// <summary>

/// Method "Delete" finds an existing sport and deletes it

/// </summary>

private void Delete()

{

Console.WriteLine("Enter id:");

int id = 0;

id = Display.GetIntNumber(id);

sportBusiness.Delete(id);

}

}

}

* + Class TeamDisplay

namespace SportsClub.Presentation

{

/// <summary>

/// In class "TeamDisplay" are implemented the methods, which describe how the team information is displayed

/// </summary>

public class TeamDisplay

{

private TeamBusiness teamBusiness = new TeamBusiness();

private TrainerBusiness trainerBusiness = new TrainerBusiness();

private SportBusiness sportBusiness = new SportBusiness();

/// <summary>

/// Method "TeamDisplay" allows us to use the functions we choose

/// </summary>

public TeamDisplay()

{

var operation = -1;

do

{

Console.Clear();

Console.WriteLine("-- Teams --");

Display.ShowActions();

try

{

operation = Display.GetIntNumber(operation);//gets the number which we entered

Console.Clear();

switch (operation)

{

case 1:

{

ListAll();

Display.GetBackToMenu();

break;

}

case 2:

{

Add();

Display.GetBackToMenu();

break;

}

case 3:

{

Update();

Display.GetBackToMenu();

break;

}

case 4:

{

FetchById();

Display.GetBackToMenu();

break;

}

case 5:

{

Delete();

Display.GetBackToMenu();

break;

}

case 6:

{

FetchByName();

Display.GetBackToMenu();

break;

}

default:

{

if (operation != Display.actionExitOperation)//checks if the number which we eneterd in not in the menu

{

Console.WriteLine("Not an operation number");

Display.GetBackToMenu();

}

break;

}

}

}

catch (ArgumentException exception)

{

Console.WriteLine(exception.Message);

Display.GetBackToMenu();

}

} while (operation != Display.actionExitOperation);

}

/// <summary>

/// Method "ListAll" shows a list of all teams and the information about them

/// </summary>

private void ListAll()

{

Console.WriteLine("Teams:");

List<Team> teams = teamBusiness.GetAll();

if (teams.Count != 0)

{

Console.WriteLine();

var table = new ConsoleTable("Id", "Name", "Sport", "Trainer");

foreach (var item in teams)

{

var sport = sportBusiness.Get(item.SportId);

var trainer = trainerBusiness.Get(item.TrainerId);

table.AddRow(item.Id, item.Name, sport.Name, trainer.Name);

}

table.Write();

}

else

{

Console.WriteLine("No teams");

}

}

/// <summary>

/// Method "Add" adds a new team to the database

/// </summary>

private void Add()

{

Team team = new Team();

Console.WriteLine("Enter Name:");

team.Name = Console.ReadLine();

Console.WriteLine("Enter SportId:");

team.SportId = Display.GetIntNumber(team.SportId);

Console.WriteLine("Enter TrainerId:");

team.TrainerId = Display.GetIntNumber(team.TrainerId);

teamBusiness.Add(team);

}

/// <summary>

/// Method "Update" finds an existing team and changes the information about it

/// </summary>

private void Update()

{

Console.WriteLine("Enter Id:");

int id = 0;

id = Display.GetIntNumber(id);

Team team = teamBusiness.Get(id);

if (team != null)

{

Console.WriteLine("Enter new name:");

team.Name = Console.ReadLine();

Console.WriteLine("Enter new sport id:");

team.SportId = Display.GetIntNumber(team.SportId);

Console.WriteLine("Enter new trainer id:");

team.TrainerId = Display.GetIntNumber(team.TrainerId);

teamBusiness.Update(team);

}

else

{

Console.WriteLine("Team not found");

}

}

/// <summary>

/// Method "FetchById" finds an existing team to which the given id matches and shows the information about it

/// </summary>

private void FetchById()

{

Console.WriteLine("Enter id:");

int id = 0;

id = Display.GetIntNumber(id);

Team team = teamBusiness.Get(id);

if (team != null)

{

Console.WriteLine($"Id: {team.Id}");

Console.WriteLine($"Name: {team.Name}");

var sport = sportBusiness.Get(team.SportId);

Console.WriteLine($"Sport: {sport.Name}");

var trainer = trainerBusiness.Get(team.TrainerId);

Console.WriteLine($"Trainer: {trainer.Name}");

}

else

{

Console.WriteLine("Team not found");

}

}

/// <summary>

/// Method "FetchByName" finds an existing team to which the given name matches and shows the information about it

/// </summary>

private void FetchByName()

{

Console.WriteLine("Enter name:");

string name = Console.ReadLine();

List<Team> teams = teamBusiness.Get(name);

if (teams.Count != 0)

{

Console.WriteLine();

var table = new ConsoleTable("Id", "Name", "Sport", "Trainer");

foreach (var item in teams)

{

var sport = sportBusiness.Get(item.SportId);

var trainer = trainerBusiness.Get(item.TrainerId);

table.AddRow(item.Id, item.Name, sport.Name, trainer.Name);

}

table.Write();

}

else

{

Console.WriteLine("No teams with such name");

}

}

/// <summary>

/// Method "Delete" finds an existing team and deletes it

/// </summary>

private void Delete()

{

Console.WriteLine("Enter id:");

int id = 0;

id = Display.GetIntNumber(id);

teamBusiness.Delete(id);

}

}

}

* + Class TrainerDisplay

namespace SportsClub.Presentation

{

/// <summary>

/// In class "TrainerDisplay" are implemented the methods, which describe how the trainer information is displayed

/// </summary>

public class TrainerDisplay

{

private TrainerBusiness trainerBusiness = new TrainerBusiness();

/// <summary>

/// Method "TrainerDisplay" allows us to use the functions we choose

/// </summary>

public TrainerDisplay()

{

var operation = -1;

do

{

Console.Clear();

Console.WriteLine("-- Trainers --");

Display.ShowActions();

try

{

operation = Display.GetIntNumber(operation);//gets the number which we entered

Console.Clear();

switch (operation)

{

case 1:

{

ListAll();

Display.GetBackToMenu();

break;

}

case 2:

{

Add();

Display.GetBackToMenu();

break;

}

case 3:

{

Update();

Display.GetBackToMenu();

break;

}

case 4:

{

FetchById();

Display.GetBackToMenu();

break;

}

case 5:

{

Delete();

Display.GetBackToMenu();

break;

}

case 6:

{

FetchByName();

Display.GetBackToMenu();

break;

}

default:

{

if (operation != Display.actionExitOperation)//checks if the number which we eneterd in not in the menu

{

Console.WriteLine("Not an operation number");

Display.GetBackToMenu();

}

break;

}

}

}

catch (ArgumentException exception)

{

Console.WriteLine(exception.Message);

Display.GetBackToMenu();

}

} while (operation != Display.actionExitOperation);

}

/// <summary>

/// Method "ListAll" shows a list of all trainers and the information about them

/// </summary>

private void ListAll()

{

Console.WriteLine("Trainers:");

List<Trainer> trainers = trainerBusiness.GetAll();

if (trainers.Count != 0)

{

Console.WriteLine();

var table = new ConsoleTable("Id", "Name");

foreach (var item in trainers)

{

table.AddRow(item.Id, item.Name);

}

table.Write();

}

else

{

Console.WriteLine("No trainers");

}

}

/// <summary>

/// Method "Add" adds a new trainer to the database

/// </summary>

private void Add()

{

Trainer trainer = new Trainer();

Console.WriteLine("Enter Name:");

trainer.Name = Console.ReadLine();

trainerBusiness.Add(trainer);

}

/// <summary>

/// Method "Update" finds an existing trainer and changes the information about him

/// </summary>

private void Update()

{

Console.WriteLine("Enter Id:");

int id = 0;

id = Display.GetIntNumber(id);

Trainer trainer = trainerBusiness.Get(id);

if (trainer != null)

{

Console.WriteLine("Enter new name:");

trainer.Name = Console.ReadLine();

trainerBusiness.Update(trainer);

}

else

{

Console.WriteLine("Trainer not found");

}

}

/// <summary>

/// Method "FetchById" finds an existing trainer to which the given id matches and shows the information about him

/// </summary>

private void FetchById()

{

Console.WriteLine("Enter id:");

int id = 0;

id = Display.GetIntNumber(id);

Trainer trainer = trainerBusiness.Get(id);

if (trainer != null)

{

Console.WriteLine($"Id: {trainer.Id}");

Console.WriteLine($"Name: {trainer.Name}");

}

else

{

Console.WriteLine("Trainer not found");

}

}

/// <summary>

/// Method "FetchByName" finds an existing trainer to which the given name matches and shows the information about him

/// </summary>

private void FetchByName()

{

Console.WriteLine("Enter name:");

string name = Console.ReadLine();

List<Trainer> trainers = trainerBusiness.Get(name);

if (trainers.Count != 0)

{

Console.WriteLine();

var table = new ConsoleTable("Id", "Name");

foreach (var item in trainers)

{

table.AddRow(item.Id, item.Name);

}

table.Write();

}

else

{

Console.WriteLine("No trainers with such name");

}

}

/// <summary>

/// Method "Delete" finds an existing trainer and deletes him

/// </summary>

private void Delete()

{

Console.WriteLine("Enter id:");

int id = 0;

id = Display.GetIntNumber(id);

trainerBusiness.Delete(id);

}

}

}

Class Program

namespace SportsClub

{

class Program

{

static void Main(string[] args)

{

Console.ForegroundColor = ConsoleColor.White;

Console.BackgroundColor = ConsoleColor.Black;

Display.GetInput();

}

}

}