**Olympic Games**

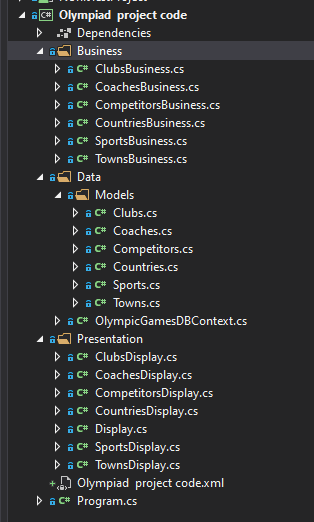
**Rio 2016**

*Изготвили: Надя Колева и Йоана Михайлова*

*От: ПМГ „Академик Боян Петканчин“, гр. Хасково*

*Проект по 7ми модул от НП „Обучение за ИТ кариера“*

**Olympiad Project code**



* Data Folder
  + Models Folder
    - **Clubs.cs**

public partial class Clubs

{

public Clubs()

{

Competitors = new HashSet<Competitors>();

}

public int Id { get; set; }

public string Name { get; set; }

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

}

* + - **Coaches.cs**

public partial class Coaches

{

public Coaches()

{

Competitors = new HashSet<Competitors>();

}

public int Id { get; set; }

public string Name { get; set; }

public int SportId { get; set; }

public virtual Sports Sport { get; set; }

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

}

* + - **Sports.cs**

public partial class Sports

{

public Sports()

{

Coaches = new HashSet<Coaches>();

Competitors = new HashSet<Competitors>();

}

public int Id { get; set; }

public string Name { get; set; }

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

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

}

* + - **Competitors.cs**

public partial class Competitors

{

public int Id { get; set; }

public string FullName { get; set; }

public string BirthDate { get; set; }

public int Age { get; set; }

public string Gender { get; set; }

public string Weight { get; set; }

public int TownId { get; set; }

public int? ClubId { get; set; }

public int? CoachId { get; set; }

public int SportId { get; set; }

public virtual Clubs Club { get; set; }

public virtual Coaches Coach { get; set; }

public virtual Sports Sport { get; set; }

public virtual Towns Town { get; set; }

}

* + - **Countries.cs**

public partial class Countries

{

public Countries()

{

Towns = new HashSet<Towns>();

}

public int Id { get; set; }

public string Name { get; set; }

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

}

* + - **Towns.cs**

public partial class Towns

{

public Towns()

{

Competitors = new HashSet<Competitors>();

}

public int Id { get; set; }

public string Name { get; set; }

public int CountryId { get; set; }

public virtual Countries Country { get; set; }

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

}

* Business Folder
  + **ClubsBusiness.cs**

/// <summary>

/// Тhe <c>ClubsBusiness</c> class is in Business layer.

/// It works as a bridge between the ClubsDisplay class and the database.

/// </summary>

/// <remarks>

/// The results are received in row data in Data Table format.

/// ClubsBusiness converts it into Value Objects.

/// </remarks>

public class ClubsBusiness

{

private OlympicGamesDBContext olympicGamesDBContext;

/// <summary>

/// Constructor for ClubsBusiness class.

/// </summary>

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

public ClubsBusiness(OlympicGamesDBContext context)

{

this.olympicGamesDBContext = context;

}

/// <summary>

/// Gets all clubs in Clubs table.

/// </summary>

/// <returns>A list of all club's names and ids.</returns>

public List<Clubs> GetAllClubs()

{

return olympicGamesDBContext.Clubs.ToList();

}

/// <summary>

/// Finds the club with the id the user has entered.

/// </summary>

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

/// <returns>The id and the name of the club.</returns>

public Clubs GetClubById(int? id)

{

return olympicGamesDBContext.Clubs.Find(id);

}

/// <summary>

/// Finds the club with the name the user has entered.

/// </summary>

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

/// <returns>The id and the name of the club.</returns>

public Clubs GetClubByName(string name)

{

var club = olympicGamesDBContext.Clubs

.Where(c => c.Name == name)

.FirstOrDefault();

return club;

}

/// <summary>

/// Adds a club in the database.

/// </summary>

/// <param name="club">The club that is being added.</param>

public void AddClub(Clubs club)

{

olympicGamesDBContext.Clubs.Add(club);

olympicGamesDBContext.SaveChanges();

}

/// <summary>

/// Updates a club in the database.

/// </summary>

/// <param name="club">The club that is being updated.</param>

public void UpdateClub(Clubs club)

{

var item = olympicGamesDBContext.Clubs.Find(club.Id);

if (item != null)

{

olympicGamesDBContext.Entry(item).CurrentValues.SetValues(club);

olympicGamesDBContext.SaveChanges();

}

}

/// <summary>

/// Deletes a club from the database by given id.

/// </summary>

/// <param name="id">The id of the club wanted to be deleted.</param>

public void DeleteClubById(int id)

{

var item = olympicGamesDBContext.Clubs.Find(id);

if (item != null)

{

olympicGamesDBContext.Remove(item);

olympicGamesDBContext.SaveChanges();

}

}

}

* + **CoachesBusiness.cs**

/// <summary>

/// Тhe <c>CoachesBusiness</c> class is in Business layer.

/// It works as a bridge between the CoachesDisplay class and the database.

/// </summary>

/// <remarks>

/// The results are received in row data in Data Table format.

/// CoachesBusiness converts it into Value Objects.

/// </remarks>

public class CoachesBusiness

{

private OlympicGamesDBContext olympicGamesDBContext;

/// <summary>

/// Constructor for CoachesBusiness class.

/// </summary>

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

public CoachesBusiness(OlympicGamesDBContext context)

{

this.olympicGamesDBContext = context;

}

/// <summary>

/// Gets all coaches in Coaches table.

/// </summary>

/// <returns>A list of all coach's names, ids and sport.</returns>

public List<Coaches> GetAllCoaches()

{

return olympicGamesDBContext.Coaches.ToList();

}

/// <summary>

/// Finds the coach with the id the user has entered.

/// </summary>

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

/// <returns>The id, the name of the coach and the sport they are coaching.</returns>

public Coaches GetCoachById(int? id)

{

return olympicGamesDBContext.Coaches.Find(id);

}

/// <summary>

/// Finds the coach with the name the user has entered.

/// </summary>

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

/// <returns>The id, the name of the coach and the sport they are coaching.</returns>

public Coaches GetCoachByName(string name)

{

var coach = olympicGamesDBContext.Coaches

.Where(c => c.Name == name)

.FirstOrDefault();

return coach;

}

/// <summary>

/// Adds a coach in the database.

/// </summary>

/// <param name="coach">The coach that is being added.</param>

public void AddCoach(Coaches coach)

{

olympicGamesDBContext.Coaches.Add(coach);

olympicGamesDBContext.SaveChanges();

}

/// <summary>

/// Updates a coach in the database.

/// </summary>

/// <param name="coach">The coach that is being updated.</param>

public void UpdateCoach(Coaches coach)

{

var item = olympicGamesDBContext.Coaches.Find(coach.Id);

if (item != null)

{

olympicGamesDBContext.Entry(item).CurrentValues.SetValues(coach);

olympicGamesDBContext.SaveChanges();

}

}

/// <summary>

/// Deletes a coach from the database by given id.

/// </summary>

/// <param name="id">The id of the coach wanted to be deleted.</param>

public void DeleteCoachById(int id)

{

var item = olympicGamesDBContext.Coaches.Find(id);

if (item != null)

{

olympicGamesDBContext.Remove(item);

olympicGamesDBContext.SaveChanges();

}

}

}

* + **CompetitorsBusiness.cs**

/// <summary>

/// Тhe <c>CompetitorsBusiness</c> class is in Business layer.

/// It works as a bridge between the CompetitorsDisplay class and the database.

/// </summary>

/// <remarks>

/// The results are received in row data in Data Table format.

/// CompetitorsBusiness converts it into Value Objects.

/// </remarks>

public class CompetitorsBusiness

{

private OlympicGamesDBContext olympicGamesDBContext;

/// <summary>

/// Constructor for CompetitorsBusiness class.

/// </summary>

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

public CompetitorsBusiness(OlympicGamesDBContext context)

{

this.olympicGamesDBContext = context;

}

/// <summary>

/// Gets all competitors in Competitors table.

/// </summary>

/// <returns>A list of all competitors and everyting about them.</returns>

public List<Competitors> GetAllCompetitors()

{

return olympicGamesDBContext.Competitors.ToList();

}

/// <summary>

/// Finds the competitor with the id the user has entered.

/// </summary>

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

/// <returns>The name of the competitor and everyting about them.</returns>

public Competitors GetCompetitorById(int id)

{

return olympicGamesDBContext.Competitors.Find(id);

}

/// <summary>

/// Finds the competitor with the name the user has entered.

/// </summary>

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

/// <returns>The name of the competitor and everyting about them..</returns>

public Competitors GetCompetitorByName(string name)

{

var competitor = olympicGamesDBContext.Competitors

.Where(c => c.FullName == name)

.FirstOrDefault();

return competitor;

}

/// <summary>

/// Adds a competitor in the database.

/// </summary>

/// <param name="competitors">The competitor that is being added.</param>

public void AddCompetitors(Competitors competitors)

{

olympicGamesDBContext.Competitors.Add(competitors);

olympicGamesDBContext.SaveChanges();

}

/// <summary>

/// Updates a competitor in the database.

/// </summary>

/// <param name="competitor">The competitor that is being updated.</param>

public void UpdateCompetitor(Competitors competitor)

{

var item = olympicGamesDBContext.Competitors.Find(competitor.Id);

if (item != null)

{

olympicGamesDBContext.Entry(item).CurrentValues.SetValues(competitor);

olympicGamesDBContext.SaveChanges();

}

}

/// <summary>

/// Deletes a competitor from the database by given id.

/// </summary>

/// <param name="id">The id of the competitor wanted to be deleted.</param>

public void DeleteCompetitorById(int id)

{

var item = olympicGamesDBContext.Competitors.Find(id);

if (item != null)

{

olympicGamesDBContext.Remove(item);

olympicGamesDBContext.SaveChanges();

}

}

}

* + **CountriesBusiness.cs**

/// <summary>

/// Тhe <c>CountriesBusiness</c> class is in Business layer.

/// It works as a bridge between the CountrieDisplay class and the database.

/// </summary>

/// <remarks>

/// The results are received in row data in Data Table format.

/// CountriesBusiness converts it into Value Objects.

/// </remarks>

public class CountriesBusiness

{

private OlympicGamesDBContext olympicGamesDBContext;

/// <summary>

/// Constructor for CountriesBusiness class.

/// </summary>

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

public CountriesBusiness(OlympicGamesDBContext context)

{

this.olympicGamesDBContext = context;

}

/// <summary>

/// Gets all countries in Countries table.

/// </summary>

/// <returns>A list of all countries's names and their ids.</returns>

public List<Countries> GetAllCountries()

{

return olympicGamesDBContext.Countries.ToList();

}

/// <summary>

/// Finds the country with the id the user has entered.

/// </summary>

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

/// <returns>The id and the name of the country.</returns>

public Countries GetCountryById(int id)

{

return olympicGamesDBContext.Countries.Find(id);

}

/// <summary>

/// Finds the country with the name the user has entered.

/// </summary>

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

/// <returns>The id and the name of the country.</returns>

public Countries GetCountryByName(string name)

{

var country = olympicGamesDBContext.Countries

.Where(c => c.Name == name)

.FirstOrDefault();

return country;

}

}

* + **SportsBusiness.cs**

/// <summary>

/// Тhe <c>SportsBusiness</c> class is in Business layer.

/// It works as a bridge between the SportsDisplay class and the database.

/// </summary>

/// <remarks>

/// The results are received in row data in Data Table format.

/// SportsBusiness converts it into Value Objects.

/// </remarks>

public class SportsBusiness

{

private OlympicGamesDBContext olympicGamesDBContext;

/// <summary>

/// Constructor for SportsBusiness class.

/// </summary>

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

public SportsBusiness(OlympicGamesDBContext context)

{

this.olympicGamesDBContext = context;

}

/// <summary>

/// Gets all sports in Sports table.

/// </summary>

/// <returns>A list of all sport's names and their ids.</returns>

public List<Sports> GetAllSports()

{

return olympicGamesDBContext.Sports.ToList();

}

/// <summary>

/// Finds the sport with the id the user has entered.

/// </summary>

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

/// <returns>The id and the name of the sport.</returns>

public Sports GetSportById(int id)

{

return olympicGamesDBContext.Sports.Find(id);

}

/// <summary>

/// Finds the sport with the name the user has entered.

/// </summary>

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

/// <returns>The id and the name of the sport.</returns>

public Sports GetSportByName(string name)

{

var sport = olympicGamesDBContext.Sports

.Where(c => c.Name == name)

.FirstOrDefault();

return sport;

}

}

* + **TownsBusiness.cs**

/// <summary>

/// Тhe <c>TownsBusiness</c> class is in Business layer.

/// It works as a bridge between the TownsDisplay class and the database.

/// </summary>

/// <remarks>

/// The results are received in row data in Data Table format.

/// TownsBusiness converts it into Value Objects.

/// </remarks>

public class TownsBusiness

{

private OlympicGamesDBContext olympicGamesDBContext;

/// <summary>

/// Constructor for TownsBusiness class.

/// </summary>

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

public TownsBusiness(OlympicGamesDBContext context)

{

this.olympicGamesDBContext = context;

}

/// <summary>

/// Gets all towns in Towns table.

/// </summary>

/// <returns>A list of all town's names, ids and countries.</returns>

public List<Towns> GetAllTowns()

{

return olympicGamesDBContext.Towns.ToList();

}

/// <summary>

/// Finds the town with the id the user has entered.

/// </summary>

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

/// <returns>The id, the name of the town and the country it is in.</returns>

public Towns GetTownById(int id)

{

return olympicGamesDBContext.Towns.FirstOrDefault(p => p.Id == id);

}

/// <summary>

/// Finds the town with the name the user has entered.

/// </summary>

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

/// <returns>The id, the name of the town and the country it is in.</returns>

public Towns GetTownByName(string name)

{

var town = olympicGamesDBContext.Towns

.Where(c => c.Name == name)

.FirstOrDefault();

return town;

}

/// <summary>

/// Adds a town in the database.

/// </summary>

/// <param name="town">The town that is being added.</param>

public void AddTown(Towns town)

{

olympicGamesDBContext.Towns.Add(town);

olympicGamesDBContext.SaveChanges();

}

/// <summary>

/// Updates a town in the database.

/// </summary>

/// <param name="town">The town that is being updated.</param>

public void UpdateTown(Towns town)

{

var item = olympicGamesDBContext.Towns.Find(town.Id);

if (item != null)

{

olympicGamesDBContext.Entry(item).CurrentValues.SetValues(town);

olympicGamesDBContext.SaveChanges();

}

}

/// <summary>

/// Deletes a town from the database by given id.

/// </summary>

/// <param name="id">The id of the town wanted to be deleted.</param>

public void DeleteTownById(int id)

{

var item = olympicGamesDBContext.Towns.Find(id);

if (item != null)

{

olympicGamesDBContext.Towns.Remove(item);

olympicGamesDBContext.SaveChanges();

}

}

}

* Presentation Folder
  + **ClubsDisplay.cs**

/// <summary>

/// Тhe <c>ClubsDisplay</c> class in Presentation.

/// This is the layer which is directly connected to ClubsBusiness.

/// </summary>

/// <remarks>

/// This class receive information from the user.

/// Then it is passed to ClubsBusiness.

/// </remarks>

class ClubsDisplay

{

private ClubsBusiness clubsBusiness;

/// <summary>

/// Constructor for ClubsDisplay class.

/// </summary>

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

public ClubsDisplay(OlympicGamesDBContext context)

{

clubsBusiness = new ClubsBusiness(context);

}

/// <summary>

/// "Calls" method "GetAllTowns" from ClubsBusiness.

/// Then it shows all clubs in table Clubs.

/// </summary>

public void GetAllClubs()

{

Console.WriteLine("Clubs:");

List<Clubs> clubs = clubsBusiness.GetAllClubs();

if(clubs.Count == 0)

{

Console.WriteLine("There are no clubs in the table!");

}

else

{

Console.WriteLine("Id" + new string(' ', 4)//6

+ "ClubName" + new string(' ', 36));

Console.WriteLine(new string('-', 50));

foreach (var club in clubs)

{

string output = $"{club.Id}" + new string(' ', 6 - club.Id.ToString().Length)

+ $"{club.Name}" + new string(' ', 44 - club.Name.Length);

Console.WriteLine(output);

}

Console.WriteLine(new string('-', 50));

}

}

/// <summary>

/// After the user has inputed id, the program "Calls" method "GetClubById" from ClubsBusiness.

/// Shows the Club who has this id.

/// </summary>

public void GetClubById()

{

Console.Write("Enter Club ID to fetch: ");

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

Clubs club = clubsBusiness.GetClubById(id);

if(club != null)

{

PrintClub(club);

}

else

{

Console.WriteLine($"There is no club with ID = {id} in the table!");

}

}

/// <summary>

/// After the user has inputed idname the program "Calls" method "GetClubById" from ClubsBusiness.

/// Shows the Club who has this name.

/// </summary>

public void GetClubByName()

{

Console.Write("Enter Club Name to fetch: ");

string name = Console.ReadLine();

Clubs club = clubsBusiness.GetClubByName(name);

if (club != null)

{

PrintClub(club);

}

else

{

Console.WriteLine($"There is no club with name = {name} in the table!");

}

}

private void PrintClub(Clubs club)

{

Console.WriteLine(new string('-', 40));

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

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

Console.WriteLine(new string('-', 40));

}

/// <summary>

/// Makes the user to input data about the club.

/// Passes the information to ClubsBusiness, using the method "AddClub"

/// </summary>

public void AddClub()

{

Clubs club = new Clubs();

Console.Write("Enter Club Name: ");

club.Name = Console.ReadLine();

clubsBusiness.AddClub(club);

Console.WriteLine($"New club successfully added!");

}

/// <summary>

/// Finds the club wished to be updated.

/// Makes the user to enter the new information.

/// Then passes it to ClubsBusiness, using the method "UpdateClub".

/// </summary>

public void UpdateClub()

{

Console.Write("Enter Club ID to update: ");

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

Clubs club = clubsBusiness.GetClubById(id);

if(club == null)

{

Console.WriteLine($"There is no club with ID = {id} in the table!");

}

else

{

Console.WriteLine("Enter Club Name: ");

club.Name = Console.ReadLine();

clubsBusiness.UpdateClub(club);

Console.WriteLine("Club successfully updated!");

}

}

/// <summary>

/// Finds the club wished to be deleted.

/// Passes the information to ClubsBusiness, using the method "DeleteClubById".

/// </summary>

public void DeleteClubById()

{

Console.Write("Enter Club Id to delete: ");

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

if(clubsBusiness.GetClubById(id) == null)

{

Console.WriteLine($"There is no club with ID = {id} in the table!");

}

else

{

clubsBusiness.DeleteClubById(id);

Console.WriteLine("Done!");

}

}

}

* + **CoachesDisplay.cs**

/// <summary>

/// Тhe <c>CoachesDisplay</c> class in Presentation.

/// This is the layer which is directly connected to CoachesBusiness.

/// </summary>

/// <remarks>

/// This class receive information from the user.

/// Then it is passed to CoachesBusiness.

/// </remarks>

class CoachesDisplay

{

private CoachesBusiness coachesBusiness;

private SportsBusiness sportsBusiness ;

/// <summary>

/// Constructor for CoachesDisplay class.

/// </summary>

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

public CoachesDisplay(OlympicGamesDBContext context)

{

coachesBusiness = new CoachesBusiness(context);

sportsBusiness = new SportsBusiness(context);

}

/// <summary>

/// "Calls" method "GetAllCoaches" from CoachesBusiness.

/// Then it shows all coaches in table Coaches.

/// </summary>

public void GetAllCoaches()

{

Console.WriteLine("Coaches: ");

List<Coaches> coaches = coachesBusiness.GetAllCoaches();

if(coaches.Count == 0)

{

Console.WriteLine("There are no coaches in the table!");

}

else

{

Console.WriteLine("Id" + new string(' ', 4) + "Name" + new string(' ', 25) + "Sport Name");

Console.WriteLine(new string('-', 56));

foreach (var coach in coaches)

{

var sport = sportsBusiness.GetSportById(coach.SportId);

string output = $"{coach.Id}" + new string(' ', 6 - coach.Id.ToString().Length)

+ $"{coach.Name}" + new string(' ', 29 - coach.Name.Length)

+ $"{sport.Name}" + new string(' ', 21 - sport.Name.Length);

Console.WriteLine(output);

}

Console.WriteLine(new string('-', 56));

}

}

/// <summary>

/// After the user has inputed id, the program "Calls" method "GetCoachById" from CoachesBusiness.

/// Shows the Coach who has this id.

/// </summary>

public void GetCoachById()

{

Console.Write("Enter Coach ID to fetch: ");

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

var coach = coachesBusiness.GetCoachById(id);

var sport = sportsBusiness.GetSportById(coach.SportId);

if (coach != null)

{

PrintCoach(coach, sport);

}

else

{

Console.WriteLine($"There is no coach with ID = {id} in the table!");

}

}

/// <summary>

/// After the user has inputed name, the program "Calls" method "GetCoachById" from CoachesBusiness.

/// Shows the Coach who has this name.

/// </summary>

public void GetCoachByName()

{

Console.Write("Enter Coach Name to fetch: ");

string name = Console.ReadLine();

var coach = coachesBusiness.GetCoachByName(name);

var sport = sportsBusiness.GetSportById(coach.SportId);

if (coach != null)

{

PrintCoach(coach, sport);

}

else

{

Console.WriteLine($"There is no coach with name = {name} in the table!");

}

}

private void PrintCoach(Coaches coach, Sports sport)

{

Console.WriteLine(new string('-', 40));

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

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

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

Console.WriteLine(new string('-', 40));

}

/// <summary>

/// Makes the user to input data about the coach.

/// Passes the information to CoachesBusinessiness, using the method "AddCoach"

/// </summary>

public void AddCoach()

{

var coach = new Coaches();

Console.Write("Enter Coach Name: ");

coach.Name = Console.ReadLine();

Console.Write("Enter Sport Name: ");

string sportName = Console.ReadLine();

coach.SportId = sportsBusiness.GetSportByName(sportName).Id;

coachesBusiness.AddCoach(coach);

Console.WriteLine($"New coach successfully added!");

}

/// <summary>

/// Finds the coach wished to be updated.

/// Makes the user to enter the new information.

/// Then passes it to CoachesBusinessiness, using the method "UpdateCoach".

/// </summary>

public void UpdateCoach()

{

Console.Write("Enter ID to update: ");

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

Coaches coach = coachesBusiness.GetCoachById(id);

if(coach == null)

{

Console.WriteLine($"There is no club with ID = {id} in the table!");

}

else

{

Console.Write("Enter Coach Name: ");

coach.Name = Console.ReadLine();

Console.Write("Enter Sport Name: ");

string sportName = Console.ReadLine();

coach.SportId = sportsBusiness.GetSportByName(sportName).Id;

coachesBusiness.UpdateCoach(coach);

Console.WriteLine("Coach successfully updated!");

}

}

/// <summary>

/// Finds the town wished to be deleted.

/// Passes the information to CoachesBusinessiness, using the method "DeleteCoachById".

/// </summary>

public void DeleteCoachById()

{

Console.Write("Enter ID to delete: ");

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

if (coachesBusiness.GetCoachById(id) == null)

{

Console.WriteLine($"There is no club with ID = {id} in the table!");

}

else

{

coachesBusiness.DeleteCoachById(id);

Console.WriteLine("Done!");

}

}

}

* + **CompetitorsDisplay.cs**
  + **CountriesDisplay.cs**

/// <summary>

/// Тhe <c>CountriesDisplay</c> class in Presentation.

/// This is the layer which is directly connected to CountriesBusiness.

/// </summary>

/// <remarks>

/// This class receive information from the user.

/// Then it is passed to CountriesBusiness.

/// </remarks>

class CountriesDisplay

{

private CountriesBusiness countriesBusiness;

private TownsBusiness towns;

/// <summary>

/// Constructor for CountriesDisplay class.

/// </summary>

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

public CountriesDisplay(OlympicGamesDBContext context)

{

countriesBusiness = new CountriesBusiness(context);

towns = new TownsBusiness(context);

}

/// <summary>

/// "Calls" method "GetAllCountries" from CountriesBusiness.

/// Then it shows all countries in table Countries.

/// </summary>

public void GetAllCountries()

{

Console.WriteLine("Countries: ");

List<Countries> countries = countriesBusiness.GetAllCountries();

if (countries.Count == 0)

{

Console.WriteLine("There are no countries in the table. ");

}

else

{

Console.WriteLine("Id" + new string(' ', 4) + "CountyName");

Console.WriteLine(new string('-', 35));

foreach (var country in countries)

{

string output = $"{country.Id}" + new string(' ', 6 - country.Id.ToString().Length)

+ $"{country.Name}" + new string(' ', 34 - country.Name.Length);

Console.WriteLine(output);

}

Console.WriteLine(new string('-', 35));

}

}

/// <summary>

/// After the user has inputed id, the program "Calls" method "GetCountryById" from CountriesBusiness.

/// Shows the Country who has this id.

/// </summary>

public void GetCountryById()

{

Console.WriteLine("Enter Country Id to fetch: ");

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

Countries country = countriesBusiness.GetCountryById(id);

if (country != null)

{

PrintCountry(country);

}

else

{

Console.WriteLine($"There is no country with id = {id} in the table!");

}

}

/// <summary>

/// After the user has inputed name, the program "Calls" method "GetCountryByName" from CountriesBusiness.

/// Shows the Country who has this name.

/// </summary>

public void GetCountryByName()

{

Console.WriteLine("Enter Country Name to fetch: ");

string name = Console.ReadLine();

Countries country = countriesBusiness.GetCountryByName(name);

if (country != null)

{

PrintCountry(country);

}

else

{

Console.WriteLine($"There is no counrty with name = {name} in the table!");

}

}

private void PrintCountry(Countries country)

{

Console.WriteLine(new string('-', 40));

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

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

Console.WriteLine(new string('-', 40));

}

}

* + **SportsDisplay.sc**

/// <summary>

/// Тhe <c>SportsDisplay</c> class in Presentation.

/// This is the layer which is directly connected to SportsBusiness.

/// </summary>

/// <remarks>

/// This class receive information from the user.

/// Then it is passed to SportsBusiness.

/// </remarks>

class SportsDisplay

{

private SportsBusiness sportsBusiness;

/// <summary>

/// Constructor for SportsDisplay class.

/// </summary>

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

public SportsDisplay(OlympicGamesDBContext context)

{

sportsBusiness = new SportsBusiness(context);

}

/// <summary>

/// "Calls" method "GetAllSports" from SportsBusiness.

/// Then it shows all sports in table Sports.

/// </summary>

public void GetAllSports()

{

Console.WriteLine("Sports: ");

List<Sports> sports = sportsBusiness.GetAllSports();

if (sports.Count == 0)

{

Console.WriteLine("There are no sports in the table!");

}

else

{

Console.WriteLine("Id" + new string(' ', 6) + "Name");

Console.WriteLine(new string('-', 27));

foreach (var sport in sports)

{

string output = $"{sport.Id}" + new string(' ', 6 - sport.Id.ToString().Length)

+ $"{sport.Name}" + new string(' ', 21 - sport.Name.Length);

Console.WriteLine(output);

}

Console.WriteLine(new string('-', 27));

}

}

/// <summary>

/// After the user has inputed id, the program "Calls" method "GetSportById" from SportsBusiness.

/// Shows the Sport who has this id.

/// </summary>

public void GetSportById()

{

Console.Write("Enter Sport Id to fetch: ");

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

Sports sport = sportsBusiness.GetSportById(id);

if (sport != null)

{

PrintSport(sport);

}

else

{

Console.WriteLine($"There is no sport with id = {id} in the table!");

}

}

/// <summary>

/// After the user has inputed name, the program "Calls" method "GetSportByName" from SportsBusiness.

/// Shows the Sport who has this name.

/// </summary>

public void GetSportByName()

{

Console.Write("Enter Sport Name to fetch: ");

string name = Console.ReadLine();

Sports sport = sportsBusiness.GetSportByName(name);

if (sport != null)

{

PrintSport(sport);

}

else

{

Console.WriteLine($"There is no sport with name = {name} in the table!");

}

}

private void PrintSport(Sports sport)

{

Console.WriteLine(new string('-', 40));

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

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

Console.WriteLine(new string('-', 40));

}

}

* + **TownsDisplay.cs**

/// <summary>

/// Тhe <c>TownsDisplay</c> class in Presentation.

/// This is the layer which is directly connected to TownsBusiness.

/// </summary>

/// <remarks>

/// This class receive information from the user.

/// Then it is passed to TownsBusiness.

/// </remarks>

public class TownsDisplay

{

private TownsBusiness townsBusiness;

private CountriesBusiness countriesBusiness;

/// <summary>

/// Constructor for TownsDisplay class.

/// </summary>

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

public TownsDisplay(OlympicGamesDBContext context)

{

townsBusiness = new TownsBusiness(context);

countriesBusiness = new CountriesBusiness(context);

}

/// <summary>

/// "Calls" method "GetAllTowns" from TownsBusiness.

/// Then it shows all towns in table Towns.

/// </summary>

public void GetAllTowns()

{

Console.WriteLine("Towns: ");

List<Towns> towns = townsBusiness.GetAllTowns();

if (towns.Count == 0)

{

Console.WriteLine("There are no towns in the table!");

}

else

{

Console.WriteLine("Id" + new string(' ', 4)

+ "Name" + new string(' ', 28) + "Country");

Console.WriteLine(new string('-', 68));

foreach (var town in towns)

{

var country = countriesBusiness.GetCountryById(town.CountryId);

string output = $"{town.Id}" + new string(' ', 6 - town.Id.ToString().Length)

+ $"{town.Name}" + new string(' ', 28 - town.Name.Length)

+ $"{country.Name}" + new string(' ', 34 - country.Name.Length);

Console.WriteLine(output);

}

Console.WriteLine(new string('-', 68));

}

}

/// <summary>

/// After the user has inputed id, the program "Calls" method "GetTownById" from TownsBusiness.

/// Shows the Town who has this id.

/// </summary>

public void GetTownById()

{

Console.WriteLine("Enter Town Id to fetch: ");

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

Towns town = townsBusiness.GetTownById(id);

if (town != null)

{

PrintTown(town);

}

else

{

Console.WriteLine($"There is no town with id = {id} in the table!");

}

}

/// <summary>

/// After the user has inputed name, the program "Calls" method "GetTownByName" from TownsBusiness.

/// Shows the Town who has this name.

/// </summary>

public void GetTownByName()

{

Console.WriteLine("Enter Town Name to fetch: ");

string name = Console.ReadLine();

Towns town = townsBusiness.GetTownByName(name);

if (town != null)

{

PrintTown(town);

}

else

{

Console.WriteLine($"There is no town with Name = {name} in the table!");

}

}

private void PrintTown(Towns town)

{

Console.WriteLine(new string('-', 40));

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

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

Console.WriteLine(new string('-', 40));

}

/// <summary>

/// Makes the user to input data about the town.

/// Passes the information to TownsBusiness, using the method "AddTown"

/// </summary>

public void AddTown()

{

Towns town = new Towns();

Console.Write("Enter name: ");

town.Name = Console.ReadLine();

Console.Write("Enter Country Name: ");

string countryName = Console.ReadLine();

town.CountryId = countriesBusiness.GetCountryByName(countryName).Id;

townsBusiness.AddTown(town);

Console.WriteLine($"New town successfully added!");

}

/// <summary>

/// Finds the town wished to be updated.

/// Makes the user to enter the new information.

/// Then passes it to TownsBusiness, using the method "UpdateTown".

/// </summary>

public void UpdateTown()

{

Console.Write("Enter Id: ");

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

Towns town = townsBusiness.GetTownById(id);

if (town != null)

{

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

town.Name = Console.ReadLine();

Console.Write("Enter country name: ");

town.CountryId = countriesBusiness.GetCountryByName(Console.ReadLine()).Id;

townsBusiness.UpdateTown(town);

}

else

{

Console.WriteLine("Town not found!");

}

Console.WriteLine("Town successfully updated!");

}

/// <summary>

/// Finds the town wished to be deleted.

/// Passes the information to TownsBusiness, using the method "DeleteTownById".

/// </summary>

public void DeleteTownById()

{

Console.Write("Enter id: ");

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

if (townsBusiness.GetTownById(id) == null)

{

Console.WriteLine($"There is no town with ID = {id} in the table!");

}

else

{

townsBusiness.DeleteTownById(id);

Console.WriteLine("Done!");

}

}

}

* + **Display.cs**