ПМГ „Академик Боян Петканчин“ – Хасково

НП „Обучение за ИТ кариера“

**Документация**

Модул 7: Разработка на софтуер

Тема: Botanical Garden

Изготвили:

Моника Джелепова - <https://github.com/MonikaDzehelepova>

Ферай Фахри - <https://github.com/feray03>

**Хасково 2021г.**

Въведение

* **Какво представлява програмата?**

Програмата е предназначена за ботанически градини, в които се отглеждат, категоризират и документират цветя и растения. Тя поддържа пет вида растения: 🌷цветя, 🌳дървета, 🌿храсти, 🌵кактуси и 🌾треви. И категоризиране по сезон за всяко растение. За всяко едно растение в базата данни се съхранява информация.

* **Какви функционалности поддържа?**
* **Text

  Description automatically generated**Извеждане на списък с всички налични растения
* Добавяне на растение
* Актуализиране на данните за дадено растение
* Търсене на растение по име
* Изтриване на растение по ИД номер

**Код**

* Data
* Models
* Flower.cs

public int Id { get; set; }

        public string Name { get; set; }

        public string Color { get; set; }

        public string LifeExpectancy { get; set; }

        public int SeasonsId { get; set; }

        public virtual Season Seasons { get; set; }

* Tree.cs

public int Id { get; set; }

        public string Name { get; set; }

        public string Type { get; set; }

        public decimal Height { get; set; }

        public decimal StemDiameter { get; set; }

        public int SeasonsId { get; set; }

        public virtual Season Seasons { get; set; }

* Shrub.cs

public int Id { get; set; }

        public string Name { get; set; }

        public string Type { get; set; }

        public decimal Height { get; set; }

        public string LifeExpectancy { get; set; }

        public int SeasonsId { get; set; }

        public virtual Season Seasons { get; set; }

* Cactus.cs

public int Id { get; set; }

        public string Name { get; set; }

        public decimal Height { get; set; }

        public string Thorns { get; set; }

        public int SeasonsId { get; set; }

        public virtual Season Seasons { get; set; }

* Grass.cs

public int Id { get; set; }

        public string Name { get; set; }

        public decimal Height { get; set; }

        public int SeasonsId { get; set; }

        public virtual Season Seasons { get; set; }

* Season.cs

public int Id { get; set; }

        public string Name { get; set; }

        public virtual ICollection<Flower> Flowers { get; set; }

        public virtual ICollection<Tree> Trees { get; set; }

        public virtual ICollection<Shrub> Shrubs { get; set; }

        public virtual ICollection<Cactus> Cactuses { get; set; }

        public virtual ICollection<Grass> Grasses { get; set; }

* Context
* GardenContext.cs

/// <summary>

    /// Context to connect to the database.

    /// </summary>

    public class GardenContext : DbContext

    {

        public virtual DbSet<Season> Seasons { get; set; }

        public virtual DbSet<Flower> Flowers { get; set; }

        public virtual DbSet<Tree> Trees { get; set; }

        public virtual DbSet<Shrub> Shrubs { get; set; }

        public virtual DbSet<Cactus> Cactuses { get; set; }

        public virtual DbSet<Grass> Grasses { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)

        {

            modelBuilder.Entity<Season>()

                .HasMany(p => p.Flowers)

                .WithOne(b => b.Seasons)

                .HasForeignKey(b => b.SeasonsId);

            modelBuilder.Entity<Season>()

                .HasMany(p => p.Trees)

                .WithOne(b => b.Seasons)

                .HasForeignKey(b => b.SeasonsId);

            modelBuilder.Entity<Season>()

                .HasMany(p => p.Shrubs)

                .WithOne(b => b.Seasons)

                .HasForeignKey(b => b.SeasonsId);

            modelBuilder.Entity<Season>()

                .HasMany(p => p.Cactuses)

                .WithOne(b => b.Seasons)

                .HasForeignKey(b => b.SeasonsId);

            modelBuilder.Entity<Season>()

                .HasMany(p => p.Grasses)

                .WithOne(b => b.Seasons)

                .HasForeignKey(b => b.SeasonsId);

        }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

        {

            optionsBuilder.UseSqlServer("Server = .\\SQLEXPRESS; Database = BotanicalGarden; Integrated Security = true ");

        }

    }

* Business
* FlowerBusiness.cs

private GardenContext context;

        /// <summary>

        ///

        /// </summary>

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

        public FlowerBusiness(GardenContext gardenContext)

        {

            context = gardenContext;

        }

        public FlowerBusiness()

        {

            context = new GardenContext();

        }

        /// <summary>

        /// Gives all flowers in the database.

        /// </summary>

        /// <returns>all flowers from the database</returns>

        public List<Flower> GetAllFlowers()

        {

            return context.Flowers.ToList();

        }

        /// <summary>

        /// Gives flower with wanted name.

        /// </summary>

        /// <param name="name">name of the wanted flower</param>

        /// <returns>flower with wanted name</returns>

 public Flower GetFlowerByName(string name)

        {

            return context.Flowers.SingleOrDefault(flower => flower.Name == name);

        }

        /// <summary>

        /// Adds flower in database.

        /// </summary>

        /// <param name="flower">the flower that will be added</param>

        public void Add(Flower flower)

        {

            context.Flowers.Add(flower);

            context.SaveChanges();

        }

        /// <summary>

        /// Updates flower.

        /// </summary>

        /// <param name="flower">the flower that will be updated</param>

        public void Update(Flower flower)

        {

            var item = context.Flowers.Find(flower.Id);

            if (item != null)

            {

                context.Entry(item).CurrentValues.SetValues(flower);

                context.SaveChanges();

            }

        }

        /// <summary>

        /// Deletes a flower with wanted id.

        /// </summary>

        /// <param name="id">id of the wanted flower</param>

        public void Delete(int id)

        {

            var item = context.Flowers.FirstOrDefault(m => m.Id == id);

            if (item != null)

            {

                context.Flowers.Remove(item);

                context.SaveChanges();

            }

        }

        public Season GetSeason(string name)

        {

            var Flower = this.GetFlowerByName(name);

            return context.Seasons.Find(Flower.SeasonsId);

        }

        /// <summary>

        /// Returns an array of flowers with the corresponding season.

        /// </summary>

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

        /// <returns></returns>

        public List<Flower> SearchBySeason(int seasonId)

        {

            Season FlowerSeason = context.Seasons.SingleOrDefault(season => season.Id == seasonId);

            return context.Flowers.Where(flower => flower.SeasonsId == FlowerSeason.Id).ToList();

        }

* TreeBusiness.cs

private GardenContext context;

        /// <summary>

        ///

        /// </summary>

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

        public TreeBusiness(GardenContext gardenContext)

        {

            context = gardenContext;

        }

        public TreeBusiness()

        {

            context = new GardenContext();

        }

        /// <summary>

        /// Gives all trees in the database.

        /// </summary>

        /// <returns>all trees from the database</returns>

        public List<Tree> GetAllTrees()

        {

            return context.Trees.ToList();

        }

        /// <summary>

        /// Gives tree with wanted name.

        /// </summary>

        /// <param name="id">id of the wanted tree</param>

        /// <returns>tree with wanted id</returns>

        public Tree GetTreeByName(string name)

        {

            return context.Trees.SingleOrDefault(tree => tree.Name == name);

        }

        /// <summary>

        /// Adds tree in database.

        /// </summary>

        /// <param name="tree">the tree that will be added</param>

        public void Add(Tree tree)

        {

            context.Trees.Add(tree);

            context.SaveChanges();

        }

        /// <summary>

        /// Updates tree.

        /// </summary>

        /// <param name="tree">the tree that will be updated</param>

        public void Update(Tree tree)

        {

            var item = context.Trees.Find(tree.Id);

            if (item != null)

            {

                context.Entry(item).CurrentValues.SetValues(tree);

                context.SaveChanges();

            }

        }

        /// <summary>

        /// Deletes a tree with wanted id.

        /// </summary>

        /// <param name="id">id of the wanted tree</param>

        public void Delete(int id)

        {

            var item = context.Trees.FirstOrDefault(m => m.Id == id);

            if (item != null)

            {

                context.Trees.Remove(item);

                context.SaveChanges();

            }

        }

        public Season GetSeason(string name)

        {

            var Tree = this.GetTreeByName(name);

            return context.Seasons.Find(Tree.SeasonsId);

        }

        /// <summary>

        /// Returns an array of trees with the corresponding season.

        /// </summary>

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

        /// <returns></returns>

        public List<Tree> SearchBySeason(int seasonId)

        {

            Season TreeSeason = context.Seasons.SingleOrDefault(season => season.Id == seasonId);

            return context.Trees.Where(tree => tree.SeasonsId == TreeSeason.Id).ToList();

        }

* ShrubBusiness.cs

private GardenContext context;

        /// <summary>

        ///

        /// </summary>

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

        public ShrubBusiness(GardenContext gardenContext)

        {

            context = gardenContext;

        }

        public ShrubBusiness()

        {

            context = new GardenContext();

        }

        /// <summary>

        /// Gives shrub with wanted name.

        /// </summary>

        /// <param name="id">id of the wanted shrub</param>

        /// <returns>shrub with wanted id</returns>

        public Shrub GetShrubByName(string name)

        {

            return context.Shrubs.SingleOrDefault(shrub => shrub.Name == name);

        }

        /// <summary>

        /// Gives all shrubs in the database.

        /// </summary>

        /// <returns>all shrubs from the database</returns>

        public List<Shrub> GetAllShrubs()

        {

            return context.Shrubs.ToList();

        }

        /// <summary>

        /// Adds shrub in database.

        /// </summary>

        /// <param name="shrub">the shrub that will be added</param>

        public void Add(Shrub shrub)

        {

            context.Shrubs.Add(shrub);

            context.SaveChanges();

        }

        /// <summary>

        /// Updates shrub.

        /// </summary>

        /// <param name="shrub">the shrub that will be updated</param>

        public void Update(Shrub shrub)

        {

            var item = context.Shrubs.Find(shrub.Id);

            if (item != null)

            {

                context.Entry(item).CurrentValues.SetValues(shrub);

                context.SaveChanges();

            }

        }

        /// <summary>

        /// Deletes a shrub with wanted id.

        /// </summary>

        /// <param name="id">id of the wanted shrub</param>

        public void Delete(int id)

        {

            var item = context.Shrubs.FirstOrDefault(m => m.Id == id);

            if (item != null)

            {

                context.Shrubs.Remove(item);

                context.SaveChanges();

            }

        }

        public Season GetSeason(string name)

        {

            var Shrub = this.GetShrubByName(name);

            return context.Seasons.Find(Shrub.SeasonsId);

        }

        /// <summary>

        /// Returns an array of shrubs with the corresponding season.

        /// </summary>

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

        /// <returns></returns>

        public List<Shrub> SearchBySeason(int seasonId)

        {

            Season ShrubSeason = context.Seasons.SingleOrDefault(season => season.Id == seasonId);

            return context.Shrubs.Where(shrub => shrub.SeasonsId == ShrubSeason.Id).ToList();

        }

* CactusBusiness.cs

private GardenContext context;

        /// <summary>

        ///

        /// </summary>

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

        public CactusBusiness(GardenContext gardenContext)

        {

            context = gardenContext;

        }

        public CactusBusiness()

        {

            context = new GardenContext();

        }

        /// <summary>

        /// Gives cactus with wanted name.

        /// </summary>

        /// <param name="id">id of the wanted cactus</param>

        /// <returns>cactus with wanted id</returns>

        public Cactus GetCactusByName(string name)

        {

            return context.Cactuses.SingleOrDefault(cactus => cactus.Name == name);

        }

        /// <summary>

        /// Gives all cactuses in the database.

        /// </summary>

        /// <returns>all cactuses from the database</returns>

        public List<Cactus> GetAllCactuses()

        {

            return context.Cactuses.ToList();

        }

        /// <summary>

        /// Adds cactus in database.

        /// </summary>

        /// <param name="cactus">the cactus that will be added</param>

        public void Add(Cactus cactus)

        {

            context.Cactuses.Add(cactus);

            context.SaveChanges();

        }

        /// <summary>

        /// Updates cactus.

        /// </summary>

        /// <param name="cactus">the cactus that will be updated</param>

        public void Update(Cactus cactus)

        {

            var item = context.Cactuses.Find(cactus.Id);

            if (item != null)

            {

                context.Entry(item).CurrentValues.SetValues(cactus);

                context.SaveChanges();

            }

        }

        /// <summary>

        /// Deletes a cactus with wanted id.

        /// </summary>

        /// <param name="id">id of the wanted cactus</param>

        public void Delete(int id)

        {

            var item = context.Cactuses.FirstOrDefault(m => m.Id == id);

            if (item != null)

            {

                context.Cactuses.Remove(item);

                context.SaveChanges();

            }

        }

        public Season GetSeason(string name)

        {

            var Cactus = this.GetCactusByName(name);

            return context.Seasons.Find(Cactus.SeasonsId);

        }

        /// <summary>

        /// Returns an array of cactuses with the corresponding season.

        /// </summary>

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

        /// <returns></returns>

        public List<Cactus> SearchBySeason(int seasonId)

        {

            Season CactusSeason = context.Seasons.SingleOrDefault(season => season.Id == seasonId);

            return context.Cactuses.Where(cactus => cactus.SeasonsId == CactusSeason.Id).ToList();

        }

* GrassBusiness.cs

private GardenContext context;

        /// <summary>

        ///

        /// </summary>

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

        public GrassBusiness(GardenContext gardenContext)

        {

            context = gardenContext;

        }

        public GrassBusiness()

        {

            context = new GardenContext();

        }

        /// <summary>

        /// Gives grass with wanted name.

        /// </summary>

        /// <param name="id">id of the wanted grass</param>

        /// <returns>grass with wanted id</returns>

        public Grass GetGrassByName(string name)

        {

            return context.Grasses.SingleOrDefault(grass => grass.Name == name);

        }

        /// <summary>

        /// Gives all Grasses in the database.

        /// </summary>

        /// <returns>all grases from the database</returns>

        public List<Grass> GetAllGrasses()

        {

            return context.Grasses.ToList();

        }

        /// <summary>

        /// Adds Grass in database.

        /// </summary>

        /// <param name="grass">the grass that will be added</param>

        public void Add(Grass grass)

        {

            context.Grasses.Add(grass);

            context.SaveChanges();

        }

        /// <summary>

        /// Updates Grass.

        /// </summary>

        /// <param name="grass">the grass that will be updated</param>

        public void Update(Grass grass)

        {

            var item = context.Grasses.Find(grass.Id);

            if (item != null)

            {

                context.Entry(item).CurrentValues.SetValues(grass);

                context.SaveChanges();

            }

        }

        /// <summary>

        /// Deletes a grass with wanted id.

        /// </summary>

        /// <param name="id">id of the wanted grass</param>

        public void Delete(int id)

        {

            var item = context.Grasses.FirstOrDefault(m => m.Id == id);

            if (item != null)

            {

                context.Grasses.Remove(item);

                context.SaveChanges();

            }

        }

        public Season GetSeason(string name)

        {

            var Grass = this.GetGrassByName(name);

            return context.Seasons.Find(Grass.SeasonsId);

        }

        /// <summary>

        /// Returns an array of grasses with the corresponding season.

        /// </summary>

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

        /// <returns></returns>

        public List<Grass> SearchBySeason(int seasonId)

        {

            Season GrassSeason = context.Seasons.SingleOrDefault(season => season.Id == seasonId);

            return context.Grasses.Where(grass => grass.SeasonsId == GrassSeason.Id).ToList();

        }

* Presentation
* FlowerDisplay.cs

private int closeOperationId = 7;

        private FlowerBusiness flowerBusiness;

        private void ShowMenu()

        {

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

            Console.WriteLine(new string(' ', 18) + "FLOWER MENU" + new string(' ', 11));

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

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

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

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

            Console.WriteLine("4. Fetch flower by Name");

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

            Console.WriteLine("6. Search flower by season");

            Console.WriteLine("7. Back");

        }

        private void Input()

        {

            var operation = -1;

            do

            {

                ShowMenu();

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

                switch (operation)

                {

                    case 1:

                        ListAllFlowers();

                        break;

                    case 2:

                        Add();

                        break;

                    case 3:

                        Update();

                        break;

                    case 4:

                        Fetch();

                        break;

                    case 5:

                        Delete();

                        break;

                    case 6:

                        SearchFlowerBySeason();

                        break;

                    default:

                        break;

                }

            }

            while (operation != closeOperationId);

        }

        public FlowerDisplay()

        {

            flowerBusiness = new FlowerBusiness();

            Input();

        }

        private void ListAllFlowers()

        {

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

            Console.WriteLine(new string(' ', 16) + "Flowers" + new string(' ', 16));

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

            var flowers = flowerBusiness.GetAllFlowers();

            foreach (var flower in flowers)

            {

                var Seasons = flowerBusiness.GetSeason(flower.Name);

                Console.WriteLine($"{flower.Id} - {flower.Name}, {flower.Color}, {flower.LifeExpectancy}, {Seasons.Name}");

            }

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

        }

        private void Add()

        {

            Flower flower = new Flower();

            Console.WriteLine("Enter name: ");

            flower.Name = Console.ReadLine();

            Console.WriteLine("Enter color: ");

            flower.Color = Console.ReadLine();

            Console.WriteLine("Enter life expectancy: ");

            flower.LifeExpectancy = Console.ReadLine();

            Console.WriteLine("Enter seasons Id: ");

            flower.SeasonsId = int.Parse(Console.ReadLine());

            flowerBusiness.Add(flower);

            Console.WriteLine("The flower was successfully added!");

        }

        private void Update()

        {

            Console.WriteLine("Enter Name to update: ");

            string name = Console.ReadLine();

            Flower flower = flowerBusiness.GetFlowerByName(name);

            if (flower != null)

            {

                Console.WriteLine("Enter name: ");

                flower.Name = Console.ReadLine();

                Console.WriteLine("Enter color: ");

                flower.Color = Console.ReadLine();

                Console.WriteLine("Enter life expectancy: ");

                flower.LifeExpectancy = Console.ReadLine();

                Console.WriteLine("Enter seasons Id: ");

                flower.SeasonsId = int.Parse(Console.ReadLine());

                flowerBusiness.Update(flower);

                Console.WriteLine("The flower was updated successfully!");

            }

            else

            {

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

            }

        }

        private void Fetch()

        {

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

            string name = Console.ReadLine();

            var flower = flowerBusiness.GetFlowerByName(name);

            if (flower != null)

            {

                var FlowerSeason = flowerBusiness.GetSeason(flower.Name);

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

                Console.WriteLine("ID: " + flower.Id);

                Console.WriteLine("Name: " + flower.Name);

                Console.WriteLine("Color: " + flower.Color);

                Console.WriteLine("Life Expectancy: " + flower.LifeExpectancy);

                Console.WriteLine("Seasons: " + FlowerSeason.Name);

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

            }

            else

            {

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

            }

        }

        private void Delete()

        {

            Console.WriteLine("Enter Id to delete: ");

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

            flowerBusiness.Delete(id);

            Console.WriteLine("Done.");

        }

        private void SearchFlowerBySeason()

        {

            Console.WriteLine("Enter season id: ");

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

            List<Flower> flowers = flowerBusiness.SearchBySeason(seasonId);

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

            Console.WriteLine(new string(' ', 16) + "Flowers" + new string(' ', 16));

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

            foreach (var flower in flowers)

            {

                Console.WriteLine($"{flower.Id} - {flower.Name}");

            }

        }

* TreeDisplay.cs

private int closeOperationId = 7;

        private TreeBusiness treeBusiness;

        private void ShowMenu()

        {

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

            Console.WriteLine(new string(' ', 18) + "TREE MENU" + new string(' ', 18));

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

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

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

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

            Console.WriteLine("4. Fetch tree by Name");

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

            Console.WriteLine("6. Search tree by season");

            Console.WriteLine("7. Back");

        }

        private void Input()

        {

            var operation = -1;

            do

            {

                ShowMenu();

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

switch (operation)

                {

                    case 1:

                        ListAllTrees();

                        break;

                    case 2:

                        Add();

                        break;

                    case 3:

                        Update();

                        break;

                    case 4:

                        Fetch();

                        break;

                    case 5:

                        Delete();

                        break;

                    case 6:

                        SearchTreeBySeason();

                        break;

                    default:

                        break;

                }

            }

            while (operation != closeOperationId);

        }

        public TreeDisplay()

        {

            treeBusiness = new TreeBusiness();

            Input();

        }

        private void ListAllTrees()

        {

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

            Console.WriteLine(new string(' ', 16) + "Trees" + new string(' ', 16));

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

            var trees = treeBusiness.GetAllTrees();

            foreach (var tree in trees)

            {

                var Seasons = treeBusiness.GetSeason(tree.Name);

                Console.WriteLine($"{tree.Id} - {tree.Name}, {tree.Type}, {tree.Height}, {tree.StemDiameter}, {Seasons.Name}");

            }

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

        }

        private void Add()

        {

            Tree tree = new Tree();

            Console.WriteLine("Enter name: ");

            tree.Name = Console.ReadLine();

            Console.WriteLine("Enter type: ");

            tree.Type = Console.ReadLine();

            Console.WriteLine("Enter height: ");

            tree.Height = decimal.Parse(Console.ReadLine());

            Console.WriteLine("Enter stem diameter: ");

            tree.StemDiameter = decimal.Parse(Console.ReadLine());

            Console.WriteLine("Enter seasons Id: ");

            tree.SeasonsId = int.Parse(Console.ReadLine());

            treeBusiness.Add(tree);

            Console.WriteLine("The tree was successfully added!");

        }

        private void Update()

        {

            Console.WriteLine("Enter Name to update: ");

            string name = Console.ReadLine();

            Tree tree = treeBusiness.GetTreeByName(name);

            if (tree != null)

            {

                Console.WriteLine("Enter name: ");

                tree.Name = Console.ReadLine();

                Console.WriteLine("Enter type: ");

                tree.Type = Console.ReadLine();

                Console.WriteLine("Enter height: ");

                tree.Height = decimal.Parse(Console.ReadLine());

                Console.WriteLine("Enter stem diameter: ");

                tree.StemDiameter = decimal.Parse(Console.ReadLine());

                Console.WriteLine("Enter seasons Id: ");

                tree.SeasonsId = int.Parse(Console.ReadLine());

                treeBusiness.Update(tree);

                Console.WriteLine("The tree was updated successfully!");

            }

            else

            {

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

            }

        }

        private void Fetch()

        {

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

            string name = Console.ReadLine();

            var tree = treeBusiness.GetTreeByName(name);

            if (tree != null)

            {

                var TreeSeason = treeBusiness.GetSeason(tree.Name);

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

                Console.WriteLine("ID: " + tree.Id);

                Console.WriteLine("Name: " + tree.Name);

                Console.WriteLine("Type: " + tree.Type);

                Console.WriteLine("Height: " + tree.Height);

                Console.WriteLine("Stem Diameter: " + tree.StemDiameter);

                Console.WriteLine("Seasons: " + TreeSeason.Name);

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

            }

            else

            {

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

            }

        }

        private void Delete()

        {

            Console.WriteLine("Enter Id to delete: ");

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

            treeBusiness.Delete(id);

            Console.WriteLine("Done.");

        }

        private void SearchTreeBySeason()

        {

            Console.WriteLine("Enter season id: ");

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

            List<Tree> trees = treeBusiness.SearchBySeason(seasonId);

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

            Console.WriteLine(new string(' ', 16) + "Trees" + new string(' ', 16));

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

            foreach (var tree in trees)

            {

                Console.WriteLine($"{tree.Id} - {tree.Name}");

            }

        }

* ShrubDisplay.cs

private int closeOperationId = 7;

        private ShrubBusiness shrubBusiness;

        private void ShowMenu()

        {

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

            Console.WriteLine(new string(' ', 18) + "SHRUB MENU" + new string(' ', 18));

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

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

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

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

            Console.WriteLine("4. Fetch shrub by Name");

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

            Console.WriteLine("6. Search shrub by season");

            Console.WriteLine("7. Back");

        }

        private void Input()

        {

            var operation = -1;

            do

            {

                ShowMenu();

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

                switch (operation)

                {

                    case 1:

                        ListAllShrubs();

                        break;

                    case 2:

                        Add();

                        break;

                    case 3:

                        Update();

                        break;

                    case 4:

                        Fetch();

                        break;

                    case 5:

                        Delete();

                        break;

                    case 6:

                        SearchShrubBySeason();

                        break;

                    default:

                        break;

                }

            }

            while (operation != closeOperationId);

        }

        public ShrubDisplay()

        {

            shrubBusiness = new ShrubBusiness();

            Input();

        }

        private void ListAllShrubs()

        {

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

            Console.WriteLine(new string(' ', 16) + "Shrubs" + new string(' ', 16));

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

            var shrubs = shrubBusiness.GetAllShrubs();

            foreach (var shrub in shrubs)

            {

                var Seasons = shrubBusiness.GetSeason(shrub.Name);

                Console.WriteLine($"{shrub.Id} - {shrub.Name}, {shrub.Type}, {shrub.Height}, {shrub.LifeExpectancy}, {Seasons.Name}");

            }

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

        }

        private void Add()

        {

            Shrub shrub = new Shrub();

            Console.WriteLine("Enter name: ");

            shrub.Name = Console.ReadLine();

            Console.WriteLine("Enter type: ");

            shrub.Type = Console.ReadLine();

            Console.WriteLine("Enter height: ");

            shrub.Height = decimal.Parse(Console.ReadLine());

            Console.WriteLine("Enter life expectancy: ");

            shrub.LifeExpectancy = Console.ReadLine();

            Console.WriteLine("Enter seasons Id: ");

            shrub.SeasonsId = int.Parse(Console.ReadLine());

            shrubBusiness.Add(shrub);

            Console.WriteLine("The shrub was successfully added!");

        }

        private void Update()

        {

            Console.WriteLine("Enter Name to update: ");

            string name = Console.ReadLine();

            Shrub shrub = shrubBusiness.GetShrubByName(name);

            if (shrub != null)

            {

                Console.WriteLine("Enter name: ");

                shrub.Name = Console.ReadLine();

                Console.WriteLine("Enter type: ");

                shrub.Type = Console.ReadLine();

                Console.WriteLine("Enter height: ");

                shrub.Height = decimal.Parse(Console.ReadLine());

                Console.WriteLine("Enter life expectancy: ");

                shrub.LifeExpectancy = Console.ReadLine();

                Console.WriteLine("Enter seasons Id: ");

                shrub.SeasonsId = int.Parse(Console.ReadLine());

                shrubBusiness.Update(shrub);

                Console.WriteLine("The shrub was updated successfully!");

            }

            else

            {

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

            }

        }

        private void Fetch()

        {

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

            string name = Console.ReadLine();

            var shrub = shrubBusiness.GetShrubByName(name);

            if (shrub != null)

            {

                var ShrubSeason = shrubBusiness.GetSeason(shrub.Name);

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

                Console.WriteLine("ID: " + shrub.Id);

                Console.WriteLine("Name: " + shrub.Name);

                Console.WriteLine("Type: " + shrub.Type);

                Console.WriteLine("Height: " + shrub.Height);

                Console.WriteLine("Life Expectancy: " + shrub.LifeExpectancy);

                Console.WriteLine("Seasons: " + ShrubSeason.Name);

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

            }

            else

            {

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

            }

        }

        private void Delete()

        {

            Console.WriteLine("Enter Id to delete: ");

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

            shrubBusiness.Delete(id);

            Console.WriteLine("Done.");

        }

        private void SearchShrubBySeason()

        {

            Console.WriteLine("Enter season id: ");

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

            List<Shrub> shrubs = shrubBusiness.SearchBySeason(seasonId);

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

            Console.WriteLine(new string(' ', 16) + "Shrubs" + new string(' ', 16));

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

            foreach (var shrub in shrubs)

            {

                Console.WriteLine($"{shrub.Id} - {shrub.Name}");

            }

        }

* CactusDisplay.cs

private int closeOperationId = 7;

        private CactusBusiness cactusBusiness;

        private void ShowMenu()

        {

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

            Console.WriteLine(new string(' ', 18) + "CACTUS MENU" + new string(' ', 18));

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

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

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

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

            Console.WriteLine("4. Fetch cactus by Name");

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

            Console.WriteLine("6. Search cactus by season");

            Console.WriteLine("7. Back");

        }

        private void Input()

        {

            var operation = -1;

            do

            {

                ShowMenu();

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

                switch (operation)

                {

                    case 1:

                        ListAllCactuses();

                        break;

                    case 2:

                        Add();

                        break;

                    case 3:

                        Update();

                        break;

                    case 4:

                        Fetch();

                        break;

                    case 5:

                        Delete();

                        break;

                    case 6:

                        SearchCactusBySeason();

                        break;

                    default:

                        break;

                }

            }

            while (operation != closeOperationId);

        }

        public CactusDisplay()

        {

            cactusBusiness = new CactusBusiness();

            Input();

        }

        private void ListAllCactuses()

        {

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

            Console.WriteLine(new string(' ', 16) + "Cactus" + new string(' ', 16));

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

            var cactuses = cactusBusiness.GetAllCactuses();

            foreach (var cactus in cactuses)

            {

                var Seasons = cactusBusiness.GetSeason(cactus.Name);

                Console.WriteLine($"{cactus.Id} - {cactus.Name}, {cactus.Height}, {cactus.Thorns}, {Seasons.Name}");

            }

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

        }

        private void Add()

        {

            Cactus cactus = new Cactus();

            Console.WriteLine("Enter name: ");

            cactus.Name = Console.ReadLine();

            Console.WriteLine("Enter height: ");

            cactus.Height = decimal.Parse(Console.ReadLine());

            Console.WriteLine("Enter thorns: ");

            cactus.Thorns = Console.ReadLine();

            Console.WriteLine("Enter seasons Id: ");

            cactus.SeasonsId = int.Parse(Console.ReadLine());

            cactusBusiness.Add(cactus);

            Console.WriteLine("The cactus was successfully added!");

        }

        private void Update()

        {

            Console.WriteLine("Enter Name to update: ");

            string name = Console.ReadLine();

            Cactus cactus = cactusBusiness.GetCactusByName(name);

            if (cactus != null)

            {

                Console.WriteLine("Enter name: ");

                cactus.Name = Console.ReadLine();

                Console.WriteLine("Enter height: ");

                cactus.Height = decimal.Parse(Console.ReadLine());

                Console.WriteLine("Enter thorns: ");

                cactus.Thorns = Console.ReadLine();

                Console.WriteLine("Enter seasons Id: ");

                cactus.SeasonsId = int.Parse(Console.ReadLine());

                cactusBusiness.Update(cactus);

                Console.WriteLine("The cactus was updated successfully!");

            }

            else

            {

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

            }

        }

        private void Fetch()

        {

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

            string name = Console.ReadLine();

            var cactus = cactusBusiness.GetCactusByName(name);

            if (cactus != null)

            {

                var CactusSeason = cactusBusiness.GetSeason(cactus.Name);

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

                Console.WriteLine("ID: " + cactus.Id);

                Console.WriteLine("Name: " + cactus.Name);

                Console.WriteLine("Height: " + cactus.Height);

                Console.WriteLine("Thorns: " + cactus.Thorns);

                Console.WriteLine("Seasons: " + CactusSeason.Name);

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

            }

            else

            {

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

            }

        }

        private void Delete()

        {

            Console.WriteLine("Enter Id to delete: ");

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

            cactusBusiness.Delete(id);

            Console.WriteLine("Done.");

        }

        private void SearchCactusBySeason()

        {

            Console.WriteLine("Enter season id: ");

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

            List<Cactus> cactuses = cactusBusiness.SearchBySeason(seasonId);

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

            Console.WriteLine(new string(' ', 16) + "Cactuses" + new string(' ', 16));

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

            foreach (var cactus in cactuses)

            {

                Console.WriteLine($"{cactus.Id} - {cactus.Name}");

            }

        }

* GrassDisplay.cs

private int closeOperationId = 7;

        private GrassBusiness grassBusiness;

        private void ShowMenu()

        {

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

            Console.WriteLine(new string(' ', 18) + "GRASS MENU" + new string(' ', 18));

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

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

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

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

            Console.WriteLine("4. Fetch grass by Name");

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

            Console.WriteLine("6. Search grass by season");

            Console.WriteLine("7. Back");

        }

        private void Input()

        {

            var operation = -1;

            do

            {

                ShowMenu();

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

                switch (operation)

                {

                    case 1:

                        ListAllGrasses();

                        break;

                    case 2:

                        Add();

                        break;

                    case 3:

                        Update();

                        break;

                    case 4:

                        Fetch();

                        break;

                    case 5:

                        Delete();

                        break;

                    case 6:

                        SearchGrassBySeason();

                        break;

                    default:

                        break;

                }

            }

            while (operation != closeOperationId);

        }

        public GrassDisplay()

        {

            grassBusiness = new GrassBusiness();

            Input();

        }

        private void ListAllGrasses()

        {

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

            Console.WriteLine(new string(' ', 16) + "Grasses" + new string(' ', 16));

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

            var grasses = grassBusiness.GetAllGrasses();

            foreach (var grass in grasses)

            {

                var Seasons = grassBusiness.GetSeason(grass.Name);

                Console.WriteLine($"{grass.Id } - {grass.Name}, {grass.Height}, {Seasons.Name}");

            }

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

        }

        private void Add()

        {

            Grass grass = new Grass();

            Console.WriteLine("Enter name: ");

            grass.Name = Console.ReadLine();

            Console.WriteLine("Enter height: ");

            grass.Height = decimal.Parse(Console.ReadLine());

            Console.WriteLine("Enter seasons Id: ");

            grass.SeasonsId = int.Parse(Console.ReadLine());

            grassBusiness.Add(grass);

            Console.WriteLine("The grass was successfully added!");

        }

        private void Update()

        {

            Console.WriteLine("Enter Name to update: ");

            string name = Console.ReadLine();

            Grass grass = grassBusiness.GetGrassByName(name);

            if (grass != null)

            {

                Console.WriteLine("Enter name: ");

                grass.Name = Console.ReadLine();

                Console.WriteLine("Enter height: ");

                grass.Height = decimal.Parse(Console.ReadLine());

                Console.WriteLine("Enter seasons Id: ");

                grass.SeasonsId = int.Parse(Console.ReadLine());

                grassBusiness.Update(grass);

                Console.WriteLine("The grass was updated successfully!");

            }

            else

            {

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

            }

        }

        private void Fetch()

        {

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

            string name = Console.ReadLine();

            var grass = grassBusiness.GetGrassByName(name);

            if (grass != null)

            {

                var GrassSeason = grassBusiness.GetSeason(grass.Name);

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

                Console.WriteLine("ID: " + grass.Id);

                Console.WriteLine("Name: " + grass.Name);

                Console.WriteLine("Height: " + grass.Height);

                Console.WriteLine("Seasons: " + GrassSeason.Name);

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

            }

            else

            {

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

            }

        }

        private void Delete()

        {

            Console.WriteLine("Enter Id to delete: ");

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

            grassBusiness.Delete(id);

            Console.WriteLine("Done.");

        }

        private void SearchGrassBySeason()

        {

            Console.WriteLine("Enter season id: ");

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

            List<Grass> grasses = grassBusiness.SearchBySeason(seasonId);

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

            Console.WriteLine(new string(' ', 16) + "Grasses" + new string(' ', 16));

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

            foreach (var grass in grasses)

            {

                Console.WriteLine($"{grass.Id} - {grass.Name}");

            }

        }

* Display.cs

private int closeOperationId = 6;

        private void ShowMenu()

        {

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

            Console.WriteLine(new string(' ', 18) + "MAIN MENU" + new string(' ', 18));

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

            Console.WriteLine("1. Flower Menu");

            Console.WriteLine("2. Tree Menu");

            Console.WriteLine("3. Shrub Menu");

            Console.WriteLine("4. Cactus Menu");

            Console.WriteLine("5. Grass Menu");

            Console.WriteLine("6. Exit");

        }

        private void Input()

        {

            var operation = -1;

            do

            {

                ShowMenu();

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

                switch (operation)

                {

                    case 1:

                        CreateFlowerDisplay();

                        break;

                    case 2:

                        CreateTreeDisplay();

                        break;

                    case 3:

                        CreateShrubDisplay();

                        break;

                    case 4:

                        CreateCactusDisplay();

                        break;

                    case 5:

                        CreateGrassDisplay();

                        break;

                    default:

                        break;

                }

            } while (operation != closeOperationId);

        }

        public Display()

        {

            Input();

        }

        private void CreateFlowerDisplay()

        {

            FlowerDisplay display = new FlowerDisplay();

        }

        private void CreateTreeDisplay()

        {

            TreeDisplay display = new TreeDisplay();

        }

        private void CreateShrubDisplay()

        {

            ShrubDisplay display = new ShrubDisplay();

        }

        private void CreateCactusDisplay()

        {

            CactusDisplay display = new CactusDisplay();

        }

        private void CreateGrassDisplay()

        {

            GrassDisplay display = new GrassDisplay();

        }

* Program.cs

static void Main(string[] args)

        {

            Information();

            Display display = new Display();

        }

        static void Information()

        {

            Console.WriteLine(new string('-', 42));

            Console.WriteLine(new string(' ', 5) + "NP\"Obuchenie IT kariera\"-2021" + new string(' ', 5));

            Console.WriteLine();

            Console.WriteLine(new string(' ', 10) + "Botanical Garden" + new string(' ', 10));

            Console.WriteLine(new string(' ', 5) + "Created by Monika and Feray" + new string(' ', 5));

            Console.WriteLine(new string('-', 42));

            Console.ReadKey();

            Console.Clear();

        }

**Тестове**

* **FlowerTests.cs**

[TestCase]

        public void Gives\_All\_Flowers()

        {

            var data = new List<Flower>

            {

                new Flower { Name="First" },

                new Flower { Name="Second" },

                new Flower { Name="Third" },

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Flower>>();

            mockSet.As<IQueryable<Flower>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Flower>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Flower>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Flower>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(c => c.Flowers).Returns(mockSet.Object);

            var business = new FlowerBusiness(mockContext.Object);

            var Flowers = business.GetAllFlowers();

            Assert.AreEqual(3, Flowers.Count);

            Assert.AreEqual("First", Flowers[0].Name);

            Assert.AreEqual("Second", Flowers[1].Name);

            Assert.AreEqual("Third", Flowers[2].Name);

        }

        [TestCase]

        public void Add\_Flower()

        {

            var mockSet = new Mock<DbSet<Flower>>();

            var flower = new Flower();

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(m => m.Flowers).Returns(mockSet.Object);

            var business = new FlowerBusiness(mockContext.Object);

            business.Add(flower);

            mockSet.Verify(m => m.Add(It.IsAny<Flower>()), Times.Once());

            mockContext.Verify(m => m.SaveChanges(), Times.Once());

        }

        [TestCase]

        public void Gives\_Flower\_By\_Name()

        {

            var data = new List<Flower>()

            {

                new Flower{Id=1, Name="Flower1"},

                new Flower{Id=2, Name="Flower2" },

                new Flower{Id=3, Name="Flower3"},

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Flower>>();

            mockSet.As<IQueryable<Flower>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Flower>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Flower>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Flower>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(c => c.Flowers).Returns(mockSet.Object);

            var business = new FlowerBusiness(mockContext.Object);

            var flower = business.GetFlowerByName("Flower1");

            Assert.AreEqual("Flower1", flower.Name);

        }

        [TestCase]

        public void Update\_Flower()

        {

            var mockContext = new Mock<GardenContext>(); ;

            var flowerBusiness=new FlowerBusiness();

            var Flower = new Flower() { Name = "Flower1" };

            try { flowerBusiness.Update(Flower); }

            catch { mockContext.Verify(m => m.Entry(It.IsAny<Flower>()), Times.Once()); }

        }

        [TestCase]

        public void Remove\_Flower()

        {

            var data = new List<Flower>()

            {

                new Flower{Id=1, Name="Flower1"},

                new Flower{Id=2, Name="Flower2" },

                new Flower{Id=3, Name="Flower3"},

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Flower>>();

            mockSet.As<IQueryable<Flower>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Flower>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Flower>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Flower>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(x => x.Flowers).Returns(mockSet.Object);

            var business = new FlowerBusiness(mockContext.Object);

            var flowers = business.GetAllFlowers();

            int deletedFlowerId = 1; business.Delete(flowers[0].Id);

            Assert.IsNull(business.GetAllFlowers().FirstOrDefault(x => x.Id == deletedFlowerId));

        }

* TreeTests.cs

[TestCase]

        public void Gives\_All\_Flowers()

        {

            var data = new List<Tree>

            {

                new Tree { Name="First" },

                new Tree { Name="Second" },

                new Tree { Name="Third" },

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Tree>>();

            mockSet.As<IQueryable<Tree>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Tree>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Tree>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Tree>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(c => c.Trees).Returns(mockSet.Object);

            var business = new TreeBusiness(mockContext.Object);

            var Trees = business.GetAllTrees();

            Assert.AreEqual(3, Trees.Count);

            Assert.AreEqual("First", Trees[0].Name);

            Assert.AreEqual("Second", Trees[1].Name);

            Assert.AreEqual("Third", Trees[2].Name);

        }

        [TestCase]

        public void Add\_Tree()

        {

            var mockSet = new Mock<DbSet<Tree>>();

            var tree = new Tree();

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(m => m.Trees).Returns(mockSet.Object);

            var business = new TreeBusiness(mockContext.Object);

            business.Add(tree);

            mockSet.Verify(m => m.Add(It.IsAny<Tree>()), Times.Once());

            mockContext.Verify(m => m.SaveChanges(), Times.Once());

        }

        [TestCase]

        public void Gives\_Tree\_By\_Name()

        {

            var data = new List<Tree>()

            {

                new Tree{Id=1, Name="Tree1"},

                new Tree{Id=2, Name="Tree2" },

                new Tree{Id=3, Name="Tree3"},

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Tree>>();

            mockSet.As<IQueryable<Tree>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Tree>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Tree>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Tree>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(c => c.Trees).Returns(mockSet.Object);

            var business = new TreeBusiness(mockContext.Object);

            var tree = business.GetTreeByName("Tree1");

            Assert.AreEqual("Tree1", tree.Name);

        }

        [TestCase]

        public void Update\_Tree()

        {

            var mockContext = new Mock<GardenContext>(); ;

            var treeBusiness = new TreeBusiness();

            var Tree = new Tree() { Name = "Tree1" };

            try { treeBusiness.Update(Tree); }

            catch { mockContext.Verify(m => m.Entry(It.IsAny<Tree>()), Times.Once()); }

        }

        [TestCase]

        public void Remove\_Tree()

        {

            var data = new List<Tree>()

            {

                new Tree{Id=1, Name="Tree1"},

                new Tree{Id=2, Name="Tree2" },

                new Tree{Id=3, Name="Tree3"},

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Tree>>();

            mockSet.As<IQueryable<Tree>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Tree>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Tree>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Tree>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(x => x.Trees).Returns(mockSet.Object);

            var business = new TreeBusiness(mockContext.Object);

            var trees = business.GetAllTrees();

            int deletedTreeId = 1; business.Delete(trees[0].Id);

            Assert.IsNull(business.GetAllTrees().FirstOrDefault(x => x.Id == deletedTreeId));

        }

* ShrubTests.cs

[TestCase]

        public void Gives\_All\_Shrubs()

        {

            var data = new List<Shrub>

            {

                new Shrub { Name="First" },

                new Shrub { Name="Second" },

                new Shrub { Name="Third" },

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Shrub>>();

            mockSet.As<IQueryable<Shrub>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Shrub>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Shrub>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Shrub>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(c => c.Shrubs).Returns(mockSet.Object);

            var business = new ShrubBusiness(mockContext.Object);

            var Shrubs = business.GetAllShrubs();

            Assert.AreEqual(3, Shrubs.Count);

            Assert.AreEqual("First", Shrubs[0].Name);

            Assert.AreEqual("Second", Shrubs[1].Name);

            Assert.AreEqual("Third", Shrubs[2].Name);

        }

        [TestCase]

        public void Add\_Shrub()

        {

            var mockSet = new Mock<DbSet<Shrub>>();

            var shrub = new Shrub();

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(m => m.Shrubs).Returns(mockSet.Object);

            var business = new ShrubBusiness(mockContext.Object);

            business.Add(shrub);

            mockSet.Verify(m => m.Add(It.IsAny<Shrub>()), Times.Once());

            mockContext.Verify(m => m.SaveChanges(), Times.Once());

        }

        [TestCase]

        public void Gives\_Shrub\_By\_Name()

        {

            var data = new List<Shrub>()

            {

                new Shrub{Id=1, Name="Shrub1"},

                new Shrub{Id=2, Name="Shrub2" },

                new Shrub{Id=3, Name="Shrub3"},

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Shrub>>();

            mockSet.As<IQueryable<Shrub>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Shrub>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Shrub>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Shrub>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(c => c.Shrubs).Returns(mockSet.Object);

            var business = new ShrubBusiness(mockContext.Object);

            var shrub = business.GetShrubByName("Shrub1");

            Assert.AreEqual("Shrub1", shrub.Name);

        }

        [TestCase]

        public void Update\_Shrub()

        {

            var mockContext = new Mock<GardenContext>(); ;

            var shrubBusiness = new ShrubBusiness();

            var Shrub = new Shrub() { Name = "Shrub1" };

            try { shrubBusiness.Update(Shrub); }

            catch { mockContext.Verify(m => m.Entry(It.IsAny<Shrub>()), Times.Once()); }

        }

        [TestCase]

        public void Remove\_Shrub()

        {

            var data = new List<Shrub>()

            {

                new Shrub{Id=1, Name="Shrub1"},

                new Shrub{Id=2, Name="Shrub2" },

                new Shrub{Id=3, Name="Shrub3"},

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Shrub>>();

            mockSet.As<IQueryable<Shrub>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Shrub>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Shrub>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Shrub>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(x => x.Shrubs).Returns(mockSet.Object);

            var business = new ShrubBusiness(mockContext.Object);

            var shrubs = business.GetAllShrubs();

            int deletedShrubId = 1; business.Delete(shrubs[0].Id);

            Assert.IsNull(business.GetAllShrubs().FirstOrDefault(x => x.Id == deletedShrubId));

        }

* CactusTests.cs

[TestCase]

        public void Gives\_All\_Flowers()

        {

            var data = new List<Cactus>

            {

                new Cactus{ Name="First" },

                new Cactus { Name="Second" },

                new Cactus { Name="Third" },

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Cactus>>();

            mockSet.As<IQueryable<Cactus>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Cactus>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Cactus>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Cactus>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(c => c.Cactuses).Returns(mockSet.Object);

            var business = new CactusBusiness(mockContext.Object);

            var Cactuses = business.GetAllCactuses();

            Assert.AreEqual(3, Cactuses.Count);

            Assert.AreEqual("First", Cactuses[0].Name);

            Assert.AreEqual("Second", Cactuses[1].Name);

            Assert.AreEqual("Third", Cactuses[2].Name);

        }

        [TestCase]

        public void Add\_Cactus()

        {

            var mockSet = new Mock<DbSet<Cactus>>();

            var cactus = new Cactus();

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(m => m.Cactuses).Returns(mockSet.Object);

            var business = new CactusBusiness(mockContext.Object);

            business.Add(cactus);

            mockSet.Verify(m => m.Add(It.IsAny<Cactus>()), Times.Once());

            mockContext.Verify(m => m.SaveChanges(), Times.Once());

        }

        [TestCase]

        public void Gives\_Cactus\_By\_Name()

        {

            var data = new List<Cactus>()

            {

                new Cactus{Id=1, Name="Cactus1"},

                new Cactus{Id=2, Name="Cactus2" },

                new Cactus{Id=3, Name="Cactus3"},

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Cactus>>();

            mockSet.As<IQueryable<Cactus>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Cactus>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Cactus>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Cactus>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(c => c.Cactuses).Returns(mockSet.Object);

            var business = new CactusBusiness(mockContext.Object);

            var cactus = business.GetCactusByName("Cactus1");

            Assert.AreEqual("Cactus1", cactus.Name);

        }

        [TestCase]

        public void Update\_Cactus()

        {

            var mockContext = new Mock<GardenContext>(); ;

            var cactusBusiness = new CactusBusiness();

            var Cactus = new Cactus() { Name = "Cactus1" };

            try { cactusBusiness.Update(Cactus); }

            catch { mockContext.Verify(m => m.Entry(It.IsAny<Cactus>()), Times.Once()); }

        }

        [TestCase]

        public void Remove\_Cactus()

        {

            var data = new List<Cactus>()

            {

                new Cactus{Id=1, Name="Cactus1"},

                new Cactus{Id=2, Name="Cactus2" },

                new Cactus{Id=3, Name="Cactus3"},

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Cactus>>();

            mockSet.As<IQueryable<Cactus>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Cactus>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Cactus>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Cactus>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(x => x.Cactuses).Returns(mockSet.Object);

            var business = new CactusBusiness(mockContext.Object);

            var cactuses = business.GetAllCactuses();

            int deletedCactusId = 1; business.Delete(cactuses[0].Id);

            Assert.IsNull(business.GetAllCactuses().FirstOrDefault(x => x.Id == deletedCactusId));

        }

* GrassTests.cs

[TestCase]

        public void Gives\_All\_Flowers()

        {

            var data = new List<Grass>

            {

                new Grass{ Name="First" },

                new Grass { Name="Second" },

                new Grass { Name="Third" },

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Grass>>();

            mockSet.As<IQueryable<Grass>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Grass>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Grass>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Grass>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(c => c.Grasses).Returns(mockSet.Object);

            var business = new GrassBusiness(mockContext.Object);

            var Grasses = business.GetAllGrasses();

            Assert.AreEqual(3, Grasses.Count);

            Assert.AreEqual("First", Grasses[0].Name);

            Assert.AreEqual("Second", Grasses[1].Name);

            Assert.AreEqual("Third", Grasses[2].Name);

        }

        [TestCase]

        public void Add\_Grass()

        {

            var mockSet = new Mock<DbSet<Grass>>();

            var grass = new Grass();

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(m => m.Grasses).Returns(mockSet.Object);

            var business = new GrassBusiness(mockContext.Object);

            business.Add(grass);

            mockSet.Verify(m => m.Add(It.IsAny<Grass>()), Times.Once());

            mockContext.Verify(m => m.SaveChanges(), Times.Once());

        }

        [TestCase]

        public void Gives\_Grass\_By\_Name()

        {

            var data = new List<Grass>()

            {

                new Grass{Id=1, Name="Grass1"},

                new Grass{Id=2, Name="Grass2" },

                new Grass{Id=3, Name="Grass3"},

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Grass>>();

            mockSet.As<IQueryable<Grass>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Grass>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Grass>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Grass>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(c => c.Grasses).Returns(mockSet.Object);

            var business = new GrassBusiness(mockContext.Object);

            var grass = business.GetGrassByName("Grass1");

            Assert.AreEqual("Grass1", grass.Name);

        }

        [TestCase]

        public void Update\_Grass()

        {

            var mockContext = new Mock<GardenContext>(); ;

            var grassBusiness = new GrassBusiness();

            var Grass = new Grass() { Name = "Grass1" };

            try { grassBusiness.Update(Grass); }

            catch { mockContext.Verify(m => m.Entry(It.IsAny<Grass>()), Times.Once()); }

        }

        [TestCase]

        public void Remove\_Grass()

        {

            var data = new List<Grass>()

            {

                new Grass{Id=1, Name="Grass1"},

                new Grass{Id=2, Name="Grass2" },

                new Grass{Id=3, Name="Grass3"},

            }.AsQueryable();

            var mockSet = new Mock<DbSet<Grass>>();

            mockSet.As<IQueryable<Grass>>().Setup(m => m.Provider).Returns(data.Provider);

            mockSet.As<IQueryable<Grass>>().Setup(m => m.Expression).Returns(data.Expression);

            mockSet.As<IQueryable<Grass>>().Setup(m => m.ElementType).Returns(data.ElementType);

            mockSet.As<IQueryable<Grass>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());

            var mockContext = new Mock<GardenContext>();

            mockContext.Setup(x => x.Grasses).Returns(mockSet.Object);

            var business = new GrassBusiness(mockContext.Object);

            var grasses = business.GetAllGrasses();

            int deletedGrassId = 1; business.Delete(grasses[0].Id);

            Assert.IsNull(business.GetAllGrasses().FirstOrDefault(x => x.Id == deletedGrassId));

        }