DAY-8

OUTPUT SCREENSHOTS:

PROGRAM.CS

using RequestTrackerAppBLLibrary;

using RequestTrackerAppModelLib;

using System.Collections;

namespace RequestTrackerApp

{

internal class Program

{

private DepartmentBL departmentBL = new DepartmentBL();

void AddDepartment()

{

while (true)

{

try

{

Console.WriteLine("Please enter the department name or 'stop' to exit");

string name = Console.ReadLine();

// Exit the loop if the user enters 'stop'

if (name.ToLower() == "stop")

{

break;

}

// Check if department with the same name already exists

if (departmentBL.DepartmentExistsByName(name))

{

Console.WriteLine("A department with this name already exists. Please enter a unique name.");

}

else

{

Department department = new Department() { Name = name };

int id = departmentBL.AddDepartment(department);

Console.WriteLine("Congrats. We have created the department for you and the Id is " + id);

}

}

catch (DuplicateDepartmentNameException ddne)

{

Console.WriteLine(ddne.Message);

}

}

}

void UpdateDepartment()

{

while (true)

{

try

{

Console.WriteLine("Please enter the department name you want to update or 'stop' to exit");

string oldName = Console.ReadLine();

// Exit the loop if the user enters 'stop'

if (oldName.ToLower() == "stop")

{

break;

}

// Check if department with the old name exists

if (departmentBL.DepartmentExistsByName(oldName))

{

Console.WriteLine("Please enter the new department name");

string newName = Console.ReadLine();

Department department = departmentBL.ChangeDepartmentName(oldName, newName);

Console.WriteLine("The department name has been updated successfully to " + department.Name);

}

else

{

Console.WriteLine("A department with this name does not exist. Please enter a valid department name.");

}

}

catch (DepartmentNotFoundException dnfe)

{

Console.WriteLine(dnfe.Message);

}

}

}

void DeleteDepartmentByName()

{

while (true)

{

try

{

Console.WriteLine("Please enter the department name to delete or 'stop' to exit");

string name = Console.ReadLine();

// Exit the loop if the user enters 'stop'

if (name.ToLower() == "stop")

{

break;

}

// Check if department with the same name exists

if (departmentBL.DepartmentExistsByName(name))

{

Department department = departmentBL.DeleteDepartmentByName(name);

Console.WriteLine("The department " + department.Name + " has been deleted successfully.");

}

else

{

Console.WriteLine("A department with this name does not exist. Please enter a valid department name.");

}

}

catch (DepartmentNotFoundException dnfe)

{

Console.WriteLine(dnfe.Message);

}

}

}

void GetAllDepartments()

{

try

{

List<Department> departments = departmentBL.GetDepartmentList();

if (departments.Count == 0)

{

Console.WriteLine("No departments found.");

}

else

{

Console.WriteLine("Here are all the departments:");

foreach (var department in departments)

{

Console.WriteLine("Id: " + department.Id + ", Name: " + department.Name);

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

static void Main(string[] args)

{

Program program = new Program();

bool running = true;

while (running)

{

Console.WriteLine("Please enter your choice:");

Console.WriteLine("1. Add Department");

Console.WriteLine("2. Delete Department");

Console.WriteLine("3. Get All Department");

Console.WriteLine("4. Update Department");

Console.WriteLine("5. Exit");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

program.AddDepartment();

break;

case "2":

program.DeleteDepartmentByName();

break;

case "3":

program.GetAllDepartments();

break;

case "4":

program.UpdateDepartment();

break;

case "5":

running = false;

break;

default:

Console.WriteLine("Invalid choice. Please enter a number between 1 and 5.");

break;

}

}

}

}

}

DepartmentBL.cs

using RequestTrackerAppModelLib;

using RequestTrackerDALLibrary;

namespace RequestTrackerAppBLLibrary

{

public class DepartmentBL : IDepartmentService

{

readonly IRepository<int, Department> \_departmentRepository;

public DepartmentBL()

{

\_departmentRepository = new DepartmentRepository();

}

public int AddDepartment(Department department)

{

// Check if department with the same name already exists

if (\_departmentRepository.DepartmentExistsByName(department.Name))

{

throw new DuplicateDepartmentNameException();

}

var result = \_departmentRepository.Add(department);

if (result != null)

{

return result.Id;

}

else

{

throw new Exception("An error occurred while adding the department.");

}

}

public Department ChangeDepartmentName(string departmentOldName, string departmentNewName)

{

var department = \_departmentRepository.GetDepartmentByName(departmentOldName);

if (department != null)

{

department.Name = departmentNewName;

\_departmentRepository.Update(department);

return department;

}

throw new DepartmentNotFoundException();

}

public Department GetDepartmentById(int id)

{

var department = \_departmentRepository.GetDepartmentById(id);

if (department != null)

{

return department;

}

throw new DepartmentNotFoundException();

}

public Department GetDepartmentByName(string departmentName)

{

var department = \_departmentRepository.GetDepartmentByName(departmentName);

if (department != null)

{

return department;

}

throw new DepartmentNotFoundException();

}

public Department DeleteDepartmentByName(string departmentName)

{

// Get the department by name

var department = \_departmentRepository.GetDepartmentByName(departmentName);

if (department != null)

{

// Delete the department

return \_departmentRepository.Delete(department.Id);

}

else

{

throw new DepartmentNotFoundException();

}

}

public List<Department> GetDepartmentList()

{

return \_departmentRepository.GetAll();

}

public bool DepartmentExists(int id)

{

var department = \_departmentRepository.GetDepartmentById(id);

return department != null;

}

public bool DepartmentExistsByName(string name)

{

var department = \_departmentRepository.GetDepartmentByName(name);

return department != null;

}

public int GetDepartmentHeadId(int departmentId)

{

throw new NotImplementedException();

}

}

}

DepartmentNotFoundException.cs

using System;

namespace RequestTrackerAppBLLibrary

{

public class DepartmentNotFoundException : Exception

{

string msg;

public DepartmentNotFoundException()

{

msg = "Department not found";

}

public override string Message => msg;

}

}

DuplicateDepartmentNameException.cs  
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RequestTrackerAppBLLibrary

{

public class DuplicateDepartmentNameException : Exception

{

string msg;

public DuplicateDepartmentNameException()

{

msg = "Department name already exists";

}

public override string Message => msg;

}

}

IDeparmentService.cs  
using RequestTrackerAppModelLib;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RequestTrackerAppBLLibrary

{

public interface IDepartmentService

{

int AddDepartment(Department department);

Department ChangeDepartmentName(string departmentOldName, string departmentNewName);

Department GetDepartmentById(int id);

Department GetDepartmentByName(string departmentName);

int GetDepartmentHeadId(int departmentId);

bool DepartmentExistsByName(string name);

}

}

Department.cs  
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RequestTrackerAppModelLib

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

public int Department\_Head { get; set; }

public override bool Equals(object? obj)

{

return this.Id.Equals((obj as Department).Id);

}

}

}

Employee.cs  
namespace RequestTrackerAppModelLib

{

public class Employee

{

public Department EmployeeDepartment { get; set; }

int age;

DateTime dob;

public int Id { get; set; }

public string Name { get; set; } = string.Empty;

public int Age

{

get

{

return age;

}

}

public DateTime DateOfBirth

{

get => dob;

set

{

dob = value;

age = ((DateTime.Today - dob).Days) / 365;

}

}

public double Salary { get; set; }

public string Type { get; set; }

public string Role { get; set; }

public Employee()

{

Id = 0;

Name = string.Empty;

Salary = 0.0;

DateOfBirth = new DateTime();

Type = string.Empty;

Role = "Employee";

}

public Employee(int id, string name, DateTime dateOfBirth, string role)

{

Id = id;

Name = name;

DateOfBirth = dateOfBirth;

Role = role;

}

public virtual void BuildEmployeeFromConsole()

{

Console.WriteLine("Please enter the Name");

Name = Console.ReadLine() ?? String.Empty;

Console.WriteLine("Please enter the Date of birth");

DateOfBirth = Convert.ToDateTime(Console.ReadLine());

Role = "Employee";

}

public override string ToString()

{

return "Employee Type : " + Type

+ "\nEmployee Id : " + Id

+ "\nEmployee Name " + Name

+ "\nDate of birth : " + DateOfBirth

+ "\nEmployee Age : " + Age

+ "\nEmployee Role " + Role;

}

public override bool Equals(object? obj)

{

Employee e1, e2;

e1 = this;

//e2 = (Employee)obj;//Casting

e2 = obj as Employee;//Casting in a more symanctic way

return e1.Id.Equals(e2.Id);

}

public static bool operator ==(Employee a, Employee b)

{

return a.Id == b.Id;

}

public static bool operator !=(Employee a, Employee b)

{

return a.Id != b.Id;

}

}

}

Request.cs  
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RequestTrackerAppModelLib

{

public class Request : IEquatable<Request>

{

public int Id { get; set; }

public string RequestText { get; set; }

public int Raised\_By { get; set; }

public string Status { get; set; }

public int Closed\_By { get; set; }

public DateTime RaisedDate { get; set; } = DateTime.Now;

public DateTime ClosedDate { get; set; }

public bool Equals(Request? other)

{

return Id.Equals(other.Id);

}

//public bool Equals(Request? other) => Id.Equals(other.Id);

}

}

DepartmentRepository.cs  
using RequestTrackerAppModelLib;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RequestTrackerDALLibrary

{

public class DepartmentRepository : IRepository<int, Department>

{

readonly Dictionary<int, Department> \_departments;

public DepartmentRepository()

{

\_departments = new Dictionary<int, Department>();

}

int GenerateId()

{

if (\_departments.Count == 0)

return 1;

int id = \_departments.Keys.Max();

return ++id;

}

public Department Add(Department item)

{

if (\_departments.Values.Any(department => department.Name == item.Name))

{

return null;

}

int newId = GenerateId();

item.Id = newId; // Set the Id of the department

\_departments.Add(newId, item);

return item;

}

public Department Delete(int key)

{

if (\_departments.ContainsKey(key))

{

var department = \_departments[key];

\_departments.Remove(key);

return department;

}

return null;

}

public Department Get(int key)

{

return \_departments.ContainsKey(key) ? \_departments[key] : null;

}

public List<Department> GetAll()

{

// If \_departments.Count is 0, return an empty list instead of null

if (\_departments.Count == 0)

return new List<Department>();

return \_departments.Values.ToList();

}

public Department Update(Department item)

{

if (\_departments.ContainsKey(item.Id))

{

\_departments[item.Id] = item;

return item;

}

return null;

}

public Department GetDepartmentByName(string name)

{

return \_departments.Values.FirstOrDefault(department => department.Name == name);

}

public Department GetDepartmentById(int id)

{

throw new NotImplementedException();

}

public bool DepartmentExistsByName(string name)

{

return \_departments.Values.Any(department => department.Name == name);

}

}

}

Irepository.cs  
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RequestTrackerDALLibrary

{

public interface IRepository<K, T> where T : class

{

List<T> GetAll();

T Get(K key);

T Add(T item);

T Update(T item);

T Delete(K key);

T GetDepartmentByName(string name);

T GetDepartmentById(int id);

bool DepartmentExistsByName(string name);

}}



