Task1:

CRUD application on employee request management

Code:

Employee model, created in separate project library

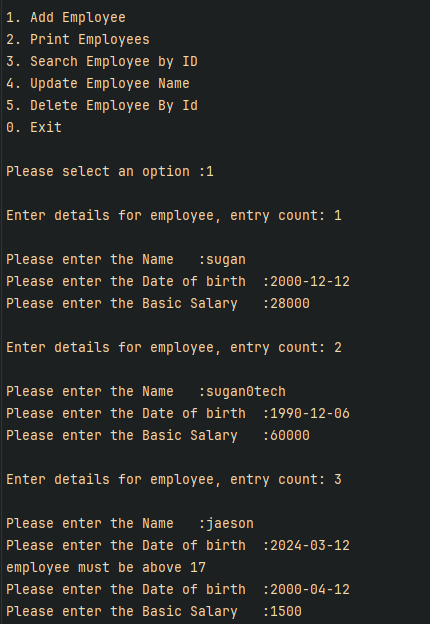
namespace RequestTrackerModelLibrary  
{  
 public class Employee  
 {  
 private int \_age;  
 private DateTime \_dob;  
 public int Id { get; set; }  
 public string Name { get; set; }  
 public int Age => \_age;  
  
 */// <summary>*  
 */// directly updates age with the given Date of Birth.*  
 */// </summary>*  
public DateTime DateOfBirth { get =>\_dob;   
 set {  
 \_dob = value;  
 \_age =(DateTime.Today - \_dob).Days/365;  
 } }  
 public double Salary { get; set; }  
  
 public Employee()  
 {  
 Id = 0;  
 Name = string.Empty;  
 Salary = 0.0;  
 DateOfBirth = new DateTime();  
 }  
 public Employee(int id, string name, DateTime dateOfBirth, double salary)  
 {  
 Id = id;  
 Name = name;  
 DateOfBirth = dateOfBirth;  
 Salary = salary;  
 }  
  
 */// <summary>*  
 */// Gets input from user and updates directly.*  
 */// </summary>*  
public void BuildEmployeeFromConsole()  
 {  
 Console.Write("Please enter the Name\t:");  
 Name = Console.ReadLine()??String.Empty;  
 Console.Write("Please enter the Date of birth\t:");  
 DateOfBirth = Convert.ToDateTime(Console.ReadLine());  
 while (Age < 18)  
 {  
 Console.WriteLine("employee must be above 17");  
 Console.Write("Please enter the Date of birth\t:");  
 DateOfBirth = Convert.ToDateTime(Console.ReadLine());  
 }  
  
  
 Console.Write("Please enter the Basic Salary\t:");  
 Salary = Convert.ToDouble(Console.ReadLine());  
 }  
  
 public void PrintEmployeeDetails()  
 {  
 Console.Write($"Employee {Id} Details:\n");  
 Console.Write($"\tEmployee Name\t:\t{Name}\n");  
 Console.Write($"\tEmployee DOB\t:\t{DateOfBirth}\n");  
 Console.Write($"\tEmployee Age\t:\t{Age}\n");  
 Console.Write($"\tEmployee Salary\t:\t{Salary}\n\n");  
 }  
 }  
}

Application program in separate project

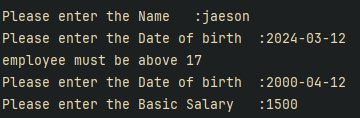
using RequestTrackerModelLibrary;  
  
namespace RequestTrackerApplication  
{  
 internal class Program  
 {  
 Employee[] employees;  
 public Program()  
 {  
 employees = new Employee[3];  
 }  
  
 */// <summary>*  
 */// For displaying options list for Employees CRUD.*  
 */// </summary>*  
void PrintMenu()  
 {  
 Console.WriteLine("\n1. Add Employee\n2. Print Employees\n3. Search Employee by ID\n4. Update Employee Name\n5. Delete Employee By Id\n0. Exit\n");  
 }  
  
 */// <summary>*  
 */// Primary method to route user options to Employee CRUD operations.*  
 */// </summary>*  
void EmployeeInteraction()  
 {  
 int choice;  
 do  
 {  
 PrintMenu();  
 Console.Write("Please select an option\t:");  
 choice = Convert.ToInt32(Console.ReadLine());  
 switch (choice)  
 {  
 case 0:  
 Console.WriteLine("Bye.....");  
 break;  
 case 1:  
 AddEmployee();  
 break;  
 case 2:  
 PrintAllEmployees();  
 break;  
 case 3:  
 SearchAndPrintEmployee();  
 break;  
 case 4:  
 UpdateNameById();  
 break;  
 case 5:  
 DeleteEmployeeById();  
 break;  
 default:  
 Console.WriteLine("Invalid choice. Try again");  
 break;  
 }  
 } while (choice !=0);  
 }  
   
 */// <summary>*  
 */// to add employees, gets input from cli*  
 */// </summary>*  
void AddEmployee()  
 {  
 if(employees[employees.Length - 1] != null)  
 {  
 Console.WriteLine("\nSorry we have reached the maximum number of employees\n");  
 return;  
 }  
  
 var counter = 1;  
 for(var i = 0; i < employees.Length; i++)  
 {  
 if (employees[i] == null)  
 {  
 Console.WriteLine($"\nEnter details for employee, entry count: {counter}\n");  
 employees[i] = CreateEmployee(i);  
 counter++;  
 }  
 }  
   
 }  
 */// <summary>*  
 */// To display all the employees details that are present in the record*  
 */// </summary>*  
void PrintAllEmployees()  
 {  
 if (employees[0] == null)  
 {  
 Console.WriteLine("No Employees available !!!\n");  
 return;  
 }  
  
 foreach (var employee in employees)  
 {  
 if (employee != null)  
 employee.PrintEmployeeDetails();  
 }  
 }  
 */// <summary>*  
 */// Base for creating employee from cli*  
 */// </summary>*  
 */// <param name="id"></param>*  
 */// <returns></returns>*  
Employee CreateEmployee(int id)  
 {  
 Employee employee = new Employee();  
 employee.Id = 101 + id;  
 try  
 {  
 employee.BuildEmployeeFromConsole();  
 }  
 catch (FormatException e)  
 {  
 Console.WriteLine(e);  
 Console.WriteLine("Then Date of Birth should be in the form of YYYY-MM-DD");  
 return null;  
 }  
 return employee;  
 }  
   
 */// <summary>*  
 */// To display the given employee with proper decoration.*  
 */// </summary>*  
 */// <param name="employee"></param>*  
void PrintEmployee(Employee employee)  
 {  
 Console.WriteLine("---------------------------");  
 employee.PrintEmployeeDetails();  
 Console.WriteLine("---------------------------");  
 }  
   
 */// <summary>*  
 */// Helper for getting Id from console. With added error handeling.*  
 */// </summary>*  
 */// <returns> int representation of employee id</returns>*  
int GetIdFromConsole()  
 {  
 int id;  
 Console.Write($"Please enter the employee Id\t:");  
 while(!int.TryParse(Console.ReadLine(), out id))  
 Console.WriteLine("\nInvalid entry. Please try again\n");  
   
 return id;  
 }  
   
 */// <summary>*  
 */// Gets Employee Id from CLI and displays that employee.*  
 */// </summary>*  
void SearchAndPrintEmployee()  
 {  
 Console.WriteLine("Print One employee");  
 int id = GetIdFromConsole();  
 Employee employee = SearchEmployeeById(id);  
 if(employee == null)  
 {  
 Console.WriteLine("\nNo such Employee is present\n");  
 return;  
 }  
 PrintEmployee(employee);  
 }  
   
 */// <summary>*  
 */// Fetches employee with the given Employee Id.*  
 */// </summary>*  
 */// <param name="id">Employee Id</param>*  
 */// <returns>Employee object if presents</returns>*  
Employee SearchEmployeeById(int id)  
 {  
 Employee employee = null;  
 for (int i = 0; i < employees.Length; i++)  
 {  
 if (employees[i] != null && employees[i].Id == id)  
 {  
 employee = employees[i];  
 break;  
 }  
 }  
 return employee;  
 }  
  
 */// <summary>*  
 */// Updates name of employee with the given Employee Id.*  
 */// </summary>*  
void UpdateNameById()  
 {  
 int id = GetIdFromConsole();  
 var employee = SearchEmployeeById(id);  
 Console.WriteLine($"Enter the new name to be updated for {employee.Name}\t:");  
 employee.Name = Console.ReadLine() ?? string.Empty;  
 Console.WriteLine($"Scuccessly updated as {SearchEmployeeById(id).Name}!!!\n");  
 }  
   
 */// <summary>*  
 */// Deletes Employee with the given Id.*  
 */// </summary>*  
void DeleteEmployeeById()  
 {  
 int id = GetIdFromConsole();  
 for (int i = 0; i < employees.Length; i++)  
 {  
 if (employees[i] != null && employees[i].Id == id)  
 {  
 employees[i] = null;  
 return;  
 }  
 }  
   
 }  
  
 static void Main(string[] args)  
 {  
 var program = new Program();  
 program.EmployeeInteraction();  
 }  
 }   
}

Output:

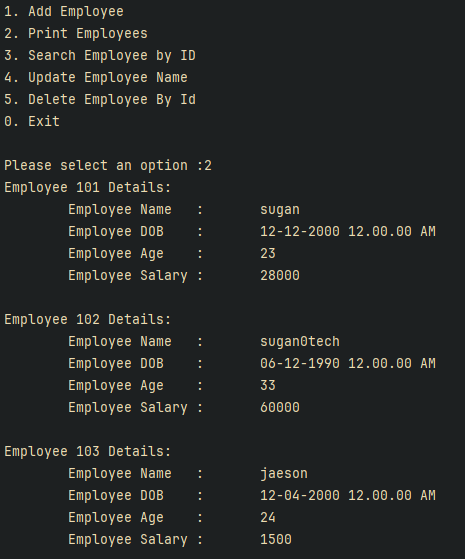
Adding employees



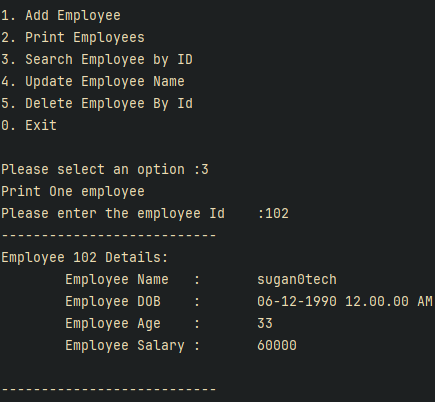
Invalid age:



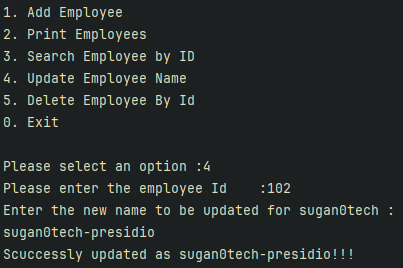
Print all employees



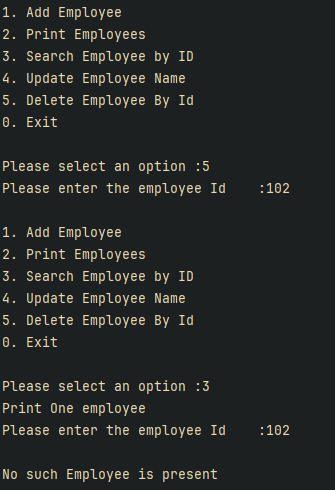
Search by Id:



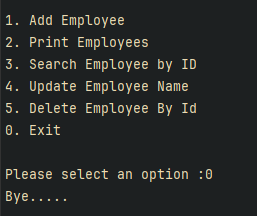
Update name by Id



Delete by Id:



Exit:



Task 2:

Bulls and cow problem

Create an application that will allow the user to play cow an bull(only for 4 char word)

Same char same position - cow

Same char diff position - bull

Example - If the word is - golf

Start the guess

heap

cows - 0, bulls - 0

kite

cows - 0, bulls -0

girl

cows - 1, bulls -1

like

cows -0, bulls 1

milk

cows -1, bull -0

goat

cows -2, bulls - 0

gold

cows -3, bulls-0

golf

cows -4, bulls -0

Congrats!!! you won!!!!!

Code:

namespace BullAndCow;  
  
class Program  
{  
 static void Main(string[] args)  
 {  
 Console.WriteLine("\t\tWelcome to the Cows and Bulls Game!!\nRules: word should be in 4 characters only consists of alphabets\n");  
  
 var secret = "golf";  
 var program = new Program();  
 do  
 {  
 Console.Write("Enter your guess\t:");  
 var guess = (Console.ReadLine()??string.Empty).ToLower();  
 if (guess.Length != 4)  
 {  
 Console.WriteLine("Please enter a valid guess\n");  
 continue;  
   
 }  
 if (guess == secret)  
 {  
 Console.WriteLine("Whola that's a fine guess");  
 return;  
 }  
   
 Console.WriteLine($"The hint is\t:\t{program.GetHint(secret, guess)}\n");  
 } while (true);  
 }  
  
 string GetHint(string secret, string guess)  
 {  
 var bullsCount = 0;  
 var cowsCount = 0;  
 var freq = new int[26];  
 var n = secret.Length;  
   
 for(int i = 0; i < n; i++){  
 if(secret[i] == guess[i])  
 bullsCount++;  
 else  
 freq[secret[i] - 'a']++;  
 }  
   
 for(int i = 0; i < n; i++){  
 if(secret[i] != guess[i]){  
 if(freq[guess[i] - 'a'] > 0){  
 cowsCount++;  
 freq[guess[i] - 'a']--;  
 }  
 }  
 }  
   
 return $"cows - {cowsCount}, bulls - {bullsCount}";  
  
 }  
}

Output:

