# Functions.cs

using System;  
using System.Collections.Generic;  
using System.Globalization;  
using System.IO;  
  
namespace ConsoleProjektH1  
{  
 public class Functions  
 {  
 /// <summary>  
 /// Shows the entire current list, fetched from the file  
 /// </summary>  
 private void ShowAll()  
 {  
 int i = 15;  
 Console.WriteLine("Name".PadRight(i) + "Age".PadRight(i) +   
 "Balance".PadRight(i));  
   
 foreach (var person in People.people)  
 {  
 if (person.Name.Length > i)  
 i = person.Name.Length + 1;  
  
 Console.WriteLine(person.Name.PadRight(i) + person.Age.ToString().PadRight(i) +   
 person.Balance.ToString(CultureInfo.InvariantCulture).PadRight(i));  
 }  
 Console.Write(Environment.NewLine);  
 }  
  
 /// <summary>  
 /// Adds a person at the end of the list, then appends the person to the .txt-file  
 /// </summary>  
 /// <param name="name"></param>  
 /// <param name="age"></param>  
 /// <param name="balance"></param>  
 private void AddPerson(string name, int age, double balance)  
 {  
 People.people.Add(new Person(name, age, balance));  
 AppendNames();  
 Console.WriteLine($"{name} was added");  
 }  
  
 /// <summary>  
 /// Removes a person with a specific name, then appends to the .txt-file  
 /// </summary>  
 /// <param name="name"></param>  
 private void DeletePerson(string name)  
 {  
 for (int i = 0; i < People.people.Count; i++)  
 { if (People.people[i].Name == name) { People.people.Remove(People.people[i]); } }  
 AppendNames();  
 Console.WriteLine($"{name} was deleted");  
 }  
   
 /// <summary>  
 /// Changes the person with a specific name, to another name, then appends to the .txt-file  
 /// </summary>  
 /// <param name="oldName"></param>  
 /// <param name="newName"></param>  
 private void ChangeName(string oldName, string newName)  
 {  
 for (int i = 0; i < People.people.Count; i++)  
 { if (People.people[i].Name == oldName) { People.people[i].Name = newName; } }  
 AppendNames();  
 Console.WriteLine($"{oldName}'s name was changed to {newName}");  
 }  
  
 /// <summary>  
 /// Changes the person with a specific name, to a different age, then appends to the .txt-file  
 /// </summary>  
 /// <param name="name"></param>  
 /// <param name="age"></param>  
 private void ChangeAge(string name, int age)  
 {  
 for (int i = 0; i < People.people.Count; i++)  
 { if (People.people[i].Name == name) { People.people[i].Age = age; } }  
 AppendNames();   
 Console.WriteLine($"{name}'s age was changed to {age}");  
 }  
  
 /// <summary>  
 /// Changes the person with a specific name, to a different balance  
 /// </summary>  
 /// <param name="name"></param>  
 /// <param name="balance"></param>  
 private void ChangeBalance(string name, double balance)  
 {  
 for (int i = 0; i < People.people.Count; i++)  
 { if (People.people[i].Name == name) { People.people[i].Balance = balance; } }  
 AppendNames();   
 Console.WriteLine($"{name}'s balance was changed to {balance}");  
 }  
   
 /// <summary>  
 /// Appends the names from the list of people to the .txt-file, separated by ',' and '\n'  
 /// </summary>  
 private void AppendNames()  
 {  
 File.WriteAllText(Environment.CurrentDirectory + "\\NameList.txt", "");  
  
 for (int i = 0; i < People.people.Count; i++)   
 {   
 Person p = new Person  
 (  
 Capitalize  
 (People.people[i].Name),  
 People.people[i].Age,  
 People.people[i].Balance  
 );  
   
 string appendText = p.Name + "," + p.Age + "," + p.Balance + Environment.NewLine;  
  
 File.AppendAllText(Environment.CurrentDirectory + "\\NameList.txt", appendText);  
 }  
 }  
  
 /// <summary>  
 /// Capitalizes the first letter in a string / char array  
 /// </summary>  
 /// <param name="word"></param>  
 /// <returns>A string with the first letter of the string, capitalized</returns>  
 private string Capitalize(string word)  
 {  
 if (word[0] != char.ToUpper(word[0]))  
 {  
 var newCharArray = word.ToCharArray();  
 if (word != "")  
 {  
 newCharArray[0] = char.ToUpper(word[0]);  
 }  
 return new string(newCharArray).Replace(" ", "");  
 }  
 return word.Replace(" ", "");   
 }  
  
 /// <summary>  
 /// Takes the input given by the user  
 /// </summary>  
 /// <param name="input"></param>  
 /// <returns>Returns the input, split up by whitespace</returns>  
 public List<string> FilterInput(string input)  
 {  
 return new List<string>(input.Split(new[] {" "}, StringSplitOptions.RemoveEmptyEntries));  
 }  
   
 /// <summary>  
 /// A method that can read the NameList file, and split up the containing lines by ',' to retrieve the  
 /// information for use  
 /// </summary>  
 public void ReadFile()  
 {  
 foreach (var line in File.ReadAllLines(Environment.CurrentDirectory + "\\NameList.txt"))  
 {  
 string[] splitUp = line.Split(',');  
  
 People.people.Add(new Person  
 (  
 Capitalize(splitUp[0]),   
 int.Parse(splitUp[1]),   
 double.Parse(splitUp[2])  
 ));  
 }  
 }  
   
 /// <summary>  
 /// A method containing a switch, that handles the entire collection of commands.  
 /// </summary>  
 /// <param name="inputList"></param>  
 /// <param name="functions"></param>  
 public void HandleCommands(List<string> inputList, Functions functions)  
 {  
 switch (inputList[0])  
 {  
 case "showall":  
 functions.ShowAll();  
 break;  
 case "addperson":  
 functions.AddPerson(Capitalize(inputList[1]), int.Parse(inputList[2]), double.Parse(inputList[3]));  
 break;  
 case "deleteperson":  
 functions.DeletePerson(Capitalize(inputList[1]));  
 break;  
 case "changeperson":  
 functions.ChangeName(Capitalize(inputList[1]), Capitalize(inputList[2]));  
 break;  
 case "changeage":  
 functions.ChangeAge(Capitalize(inputList[1]), int.Parse(inputList[2]));  
 break;  
 case "changebalance":  
 functions.ChangeBalance(Capitalize(inputList[1]), double.Parse(inputList[2]));  
 break;  
 case "clear":  
 Console.Clear();  
 Console.WriteLine(@"Hello, welcome to this list of people - Type ""help"" to receive a list of commands");  
 break;  
 case "quit":  
 Environment.Exit(0);  
 break;  
 case "help":  
 Console.WriteLine("These are the available commands: \n\n" +  
 " \"showall\" - Shows the current list of people\n" +   
 " \"addperson\" <name> <age> <balance> - Adds a person to the current list of people\n" +   
 " \"deleteperson\" <name> - Deletes a person from the current list of people\n" +  
 " \"changeperson\" <oldname> <newname> - changes the name of a person from the current list of people\n" +   
 " \"changeage\" <name> <newage> - changes the age of a person from the current list of people\n" +  
 " \"changebalance\" <name> <newbalance> - changes the balance of a person from the current list of people\n" +   
 " \"quit\" - Quits the console\n" +   
 " \"help\" - Shows this list of available commands");  
 break;  
 default:  
 Console.WriteLine("That is not a command");  
 break;  
 }  
 }  
 }  
}

# Program.cs

using System;  
using System.Collections.Generic;  
  
namespace ConsoleProjektH1  
{  
 class Program  
 {  
 private static void Main()  
 {  
 new Program().Run();  
 }  
   
 /// <summary>  
 /// Handles user interface/experience and catches user errors  
 /// </summary>  
 private void Run()  
 {  
 try  
 {  
 Functions functions = new Functions();  
 functions.ReadFile();  
 Console.WriteLine("Hello, welcome to this list of people - Type \"help\" to " +  
 "receive a list of commands");  
 while (true)  
 {  
 Console.Write(":>");  
 List<string> inputList = functions.FilterInput(Console.ReadLine()?.ToLower());  
 try  
 {  
 Console.Clear();  
 functions.HandleCommands(inputList, functions);  
 Console.WriteLine("\nPlease enter a command");  
 }  
 catch (Exception e)  
 {  
 if (inputList[0] == "changeperson" || inputList[0] == "changeage" ||  
 inputList[0] == "changebalance" || inputList[0] == "deleteperson" ||  
 inputList[0] == "addperson")  
 {  
 Console.WriteLine("That person is not on the list, or you entered an incorrect value");  
 }  
 else  
 {  
 Console.WriteLine("Please enter a name");  
 Console.WriteLine(e);  
 }  
 }  
 }  
 }  
 catch (Exception nfe)  
 {  
 Console.WriteLine(nfe);  
 }  
 }  
 }  
}

# Information.cs

using System.Collections.Generic;  
  
namespace ConsoleProjektH1  
{  
 /// <summary>  
 /// A class to describe and contain the value of people  
 /// </summary>  
 public class Person  
 {  
 private string name;  
 private int age;  
 private double balance;  
  
 public string Name { get => name; set => name = value; }  
 public int Age { get => age; set => age = value; }  
 public double Balance { get => balance; set => balance = value; }  
  
 //public string Name { get; set; }  
 //public int age { get; set; }  
 //public double balance { get; set; }  
  
 public Person(string name, int age, double balance)  
 {  
 this.name = name;  
 this.age = age;  
 this.balance = balance;  
 }  
  
 }  
  
 public class People  
 {  
 public static List<Person> people = new List<Person>();  
 }  
}