using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

// Failu sistemas objekta klase

public class FailuSistObjekt

{

public string Nosaukums { get; set; }

public FailuSistObjekt Vecaks { get; set; }

public List<FailuSistObjekt> Berni { get; set; }

public bool IrMape { get; set; }

public string FiziskaDatne { get; set; } // Realas datnes cels

// Konstruktors

public FailuSistObjekt(string nosaukums, bool irMape, FailuSistObjekt vecaks = null)

{

Nosaukums = nosaukums;

IrMape = irMape;

Vecaks = vecaks;

Berni = new List<FailuSistObjekt>();

}

// Paradit metodes

public void Paradit()

{

if (IrMape)

{

Console.WriteLine($"/[{Nosaukums}]");

foreach (var bernis in Berni)

{

bernis.Paradit();

}

}

else

{

Console.WriteLine($"~[{Nosaukums}]~");

}

}

// Dzesanas metode

public void Dzest()

{

if (IrMape && Berni.Count > 0)

{

Console.WriteLine($"Vai tiesam velaties dzest mapi '{Nosaukums}' un visus tas saturus? (y/n)");

var atbilde = Console.ReadLine().ToLower();

if (atbilde == "y")

{

// Dzesam visus bernus rekursivi

for (int i = Berni.Count - 1; i >= 0; i--)

{

Berni[i].Dzest();

}

Vecaks.Berni.Remove(this);

Console.WriteLine($"Mape '{Nosaukums}' dzesta.");

}

}

else

{

if (Vecaks != null)

{

Vecaks.Berni.Remove(this);

}

Console.WriteLine($"Objekts '{Nosaukums}' dzests.");

// Ja ir fiziska datne, dzesam ari to

if (!string.IsNullOrEmpty(FiziskaDatne) && File.Exists(FiziskaDatne))

{

try

{

File.Delete(FiziskaDatne);

}

catch (Exception ex)

{

Console.WriteLine($"Kluda dzesot fizisko datni: {ex.Message}");

}

}

}

}

// Parvietosanas metode

public FailuSistObjekt Parvietoties(string mape)

{

if (!IrMape) return null;

if (mape == "..")

{

return Vecaks ?? this; // Ja nav vecaka, paliekam seit

}

else if (mape == "/")

{

// Atgriezjamies sakne

var sakne = this;

while (sakne.Vecaks != null)

{

sakne = sakne.Vecaks;

}

return sakne;

}

else

{

// Meklejam bernu mapi

foreach (var bernis in Berni)

{

if (bernis.IrMape && bernis.Nosaukums.Equals(mape, StringComparison.OrdinalIgnoreCase))

{

return bernis;

}

}

return null; // Mape nav atrasta

}

}

// Izveides metode

public void Izveidot(bool irMape, string nosaukums)

{

if (!IrMape) return;

// Parbaudam, vai nosaukums jau eksiste

foreach (var bernis in Berni)

{

if (bernis.Nosaukums.Equals(nosaukums, StringComparison.OrdinalIgnoreCase))

{

Console.WriteLine($"Kluda: Objekts ar nosaukumu '{nosaukums}' jau eksiste!");

return;

}

}

var jaunsObjekts = new FailuSistObjekt(nosaukums, irMape, this);

Berni.Add(jaunsObjekts);

Console.WriteLine($"Izveidots jauns {(irMape ? "mape" : "datne")}: '{nosaukums}'");

}

}

// Galvena programmas klase

public class Program

{

private static FailuSistObjekt \_currentDirectory;

private static FailuSistObjekt \_sakne;

// Metode, kas izveido sakuma strukturu

private static void InicializetFailuSistemu()

{

\_sakne = new FailuSistObjekt("c:", true);

\_currentDirectory = \_sakne;

}

// Metode, kas paradis pasreizejo atrasanas vietu

private static void ParaditCurrentPath()

{

var path = new List<string>();

var current = \_currentDirectory;

while (current != null)

{

path.Insert(0, current.Nosaukums);

current = current.Vecaks;

}

Console.Write(string.Join("/", path) + "/> ");

}

// Metode, kas apstrada komandas

private static void ApstradatKomandu(string komanda)

{

var parts = komanda.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

if (parts.Length == 0) return;

var cmd = parts[0].ToLower();

var arguments = new List<string>();

for (int i = 1; i < parts.Length; i++)

{

arguments.Add(parts[i]);

}

try

{

switch (cmd)

{

case "mkdir":

if (arguments.Count == 0)

{

Console.WriteLine("Kluda: Noradiet mapes nosaukumu!");

return;

}

foreach (var mape in arguments.Take(3))

{

\_currentDirectory.Izveidot(true, mape);

}

break;

case "create":

if (arguments.Count == 0)

{

Console.WriteLine("Kluda: Noradiet datnes nosaukumu!");

return;

}

foreach (var datne in arguments.Take(3))

{

\_currentDirectory.Izveidot(false, datne);

}

break;

case "rm":

if (arguments.Count == 0)

{

Console.WriteLine("Kluda: Noradiet mapes nosaukumu!");

return;

}

foreach (var mape in arguments.Take(3))

{

var found = false;

foreach (var bernis in \_currentDirectory.Berni.ToList())

{

if (bernis.IrMape && bernis.Nosaukums.Equals(mape, StringComparison.OrdinalIgnoreCase))

{

bernis.Dzest();

found = true;

break;

}

}

if (!found)

{

Console.WriteLine($"Kluda: Mape '{mape}' nav atrasta!");

}

}

break;

case "dir":

if (arguments.Count == 0)

{

Console.WriteLine($"Satura saraksts mape '{\_currentDirectory.Nosaukums}':");

foreach (var bernis in \_currentDirectory.Berni)

{

bernis.Paradit();

}

}

else if (arguments.Count == 1 && arguments[0].StartsWith(">"))

{

var datnesNosaukums = arguments[0].Substring(1).Trim();

if (string.IsNullOrEmpty(datnesNosaukums))

{

Console.WriteLine("Kluda: Noradiet datnes nosaukumu!");

return;

}

// Meklejam vai izveidojam datni

FailuSistObjekt datne = null;

foreach (var bernis in \_currentDirectory.Berni)

{

if (!bernis.IrMape && bernis.Nosaukums.Equals(datnesNosaukums, StringComparison.OrdinalIgnoreCase))

{

datne = bernis;

break;

}

}

if (datne == null)

{

datne = new FailuSistObjekt(datnesNosaukums, false, \_currentDirectory);

\_currentDirectory.Berni.Add(datne);

}

// Izveidojam fizisko datni

try

{

var path = Path.Combine(Directory.GetCurrentDirectory(), datnesNosaukums);

using (var writer = new StreamWriter(path))

{

foreach (var bernis in \_currentDirectory.Berni)

{

writer.WriteLine(bernis.Nosaukums + (bernis.IrMape ? " [MAPE]" : " [DATNE]"));

}

}

datne.FiziskaDatne = path;

Console.WriteLine($"Satura saraksts saglabats datne '{datnesNosaukums}'");

}

catch (Exception ex)

{

Console.WriteLine($"Kluda rakstot datne: {ex.Message}");

}

}

else

{

Console.WriteLine("Kluda: Nepareiza 'dir' komandas sintakse!");

}

break;

case "cd":

if (arguments.Count != 1)

{

Console.WriteLine("Kluda: Noradiet mapes nosaukumu vai '..' vai '/'!");

return;

}

var newDir = \_currentDirectory.Parvietoties(arguments[0]);

if (newDir != null)

{

\_currentDirectory = newDir;

}

else

{

Console.WriteLine($"Kluda: Mape '{arguments[0]}' nav atrasta!");

}

break;

case "edit":

if (arguments.Count != 1)

{

Console.WriteLine("Kluda: Noradiet datnes nosaukumu!");

return;

}

var foundFile = false;

foreach (var bernis in \_currentDirectory.Berni)

{

if (!bernis.IrMape && bernis.Nosaukums.Equals(arguments[0], StringComparison.OrdinalIgnoreCase))

{

foundFile = true;

if (!string.IsNullOrEmpty(bernis.FiziskaDatne) && File.Exists(bernis.FiziskaDatne))

{

Console.WriteLine($"Datnes '{bernis.Nosaukums}' saturs:");

Console.WriteLine(File.ReadAllText(bernis.FiziskaDatne));

}

else

{

Console.WriteLine($"Datne '{bernis.Nosaukums}' ir tuksa vai nav izveidota fiziski.");

}

break;

}

}

if (!foundFile)

{

Console.WriteLine($"Kluda: Datne '{arguments[0]}' nav atrasta!");

}

break;

case "del":

if (arguments.Count == 0)

{

Console.WriteLine("Kluda: Noradiet datnes nosaukumu!");

return;

}

foreach (var datne in arguments.Take(3))

{

var found = false;

foreach (var bernis in \_currentDirectory.Berni.ToList())

{

if (!bernis.IrMape && bernis.Nosaukums.Equals(datne, StringComparison.OrdinalIgnoreCase))

{

bernis.Dzest();

found = true;

break;

}

}

if (!found)

{

Console.WriteLine($"Kluda: Datne '{datne}' nav atrasta!");

}

}

break;

case "exit":

case "by":

Environment.Exit(0);

break;

default:

Console.WriteLine($"Kluda: Nezinama komanda '{cmd}'!");

break;

}

}

catch (Exception ex)

{

Console.WriteLine($"Kluda: {ex.Message}");

}

}

// Galvena metode

public static void Main(string[] args)

{

Console.WriteLine("Failu sistemas simulators");

Console.WriteLine("Pieejamas komandas: mkdir, create, rm, dir, cd, edit, del, exit/by");

InicializetFailuSistemu();

while (true)

{

ParaditCurrentPath();

var komanda = Console.ReadLine();

if (string.IsNullOrWhiteSpace(komanda)) continue;

ApstradatKomandu(komanda);

}

}

}