using Newtonsoft.Json;

using System.Collections.Concurrent;

using System.Xml;

namespace RepositoryManager

{

public class Class1

{

private readonly ConcurrentDictionary<string, RepositoryItem> \_repository;

private bool \_initialized;

private static readonly object \_lock = new object();

public Class1()

{

\_repository = new ConcurrentDictionary<string, RepositoryItem>();

\_initialized = false;

Init();

}

// Represents a repo(db) with content and type

private class RepositoryItem

{

public string Content { get; }

public int Type { get; }

public RepositoryItem(string content, int type)

{

Content = content;

Type = type;

}

}

// Ensures repository initialization happens only once

public void Init()

{

if (!\_initialized)

{

lock (\_lock)

{

if (!\_initialized)

{

\_initialized = true;

}

}

}

}

// Registers an item into the repository with validation

public void Register(string itemName, string itemContent, int itemType)

{

Validate(itemContent, itemType); // Validate the content based on the type

var newItem = new RepositoryItem(itemContent, itemType);

// Prevent overwriting existing items

if (!\_repository.TryAdd(itemName, newItem))

{

throw new InvalidOperationException($"Item with name '{itemName}' already exists.");

}

}

// Retrieves an item's content from the repository

public string Retrieve(string itemName)

{

if (\_repository.TryGetValue(itemName, out var item))

{

return item.Content;

}

throw new KeyNotFoundException($"Item with name '{itemName}' not found.");

}

// Retrieves the type of an item (JSON or XML)

public int GetType(string itemName)

{

itemName = itemName.Trim(); // Remove leading/trailing whitespaces

// Check if it's JSON (object or array)

if ((itemName.StartsWith("{") && itemName.EndsWith("}")) || // Object

(itemName.StartsWith("[") && itemName.EndsWith("]"))) // Array

{

return 1; // JSON type

}

// Check if it's XML

else if (itemName.StartsWith("<") && itemName.EndsWith(">"))

{

return 2; // XML type

}

else

{

throw new InvalidDataException("Item content is neither JSON nor XML.");

}

}

// Removes an item from the repository

public void Deregister(string itemName)

{

if (!\_repository.TryRemove(itemName, out \_))

{

throw new KeyNotFoundException($"Item with name '{itemName}' not found or could not be removed.");

}

}

// Validates item content based on the item type (logic should be implemented)

public void Validate(string itemContent, int itemType)

{

if (itemType == 1)

{

try

{

// Attempt to parse the JSON content

var parsedJson = JsonConvert.DeserializeObject(itemContent.Trim());

if (parsedJson == null)

{

throw new FormatException("JSON content is invalid.");

}

}

catch (Newtonsoft.Json.JsonException ex)

{

throw new FormatException("Invalid JSON format.", ex);

}

}

else if (itemType == 2)

{

try

{

// Attempt to load the XML content

var xmlDoc = new XmlDocument();

xmlDoc.LoadXml(itemContent.Trim()); // This will throw if the XML is invalid

}

catch (XmlException ex)

{

throw new FormatException("Invalid XML format.", ex);

}

}

else

{

throw new ArgumentException("Invalid item type provided.");

}

}

}

}