**Лабораторна робота № 6  
Минко Ярослав**

**1. Дженеричний калькулятор**using System;

public class Calculator<T>

{

public delegate T ArithmeticOperation(T x, T y);

public ArithmeticOperation Add { get; set; }

public ArithmeticOperation Subtract { get; set; }

public ArithmeticOperation Multiply { get; set; }

public ArithmeticOperation Divide { get; set; }

public Calculator(ArithmeticOperation add, ArithmeticOperation subtract, ArithmeticOperation multiply, ArithmeticOperation divide)

{

Add = add;

Subtract = subtract;

Multiply = multiply;

Divide = divide;

}

public T PerformOperation(T x, T y, ArithmeticOperation operation)

{

if (operation == null)

{

throw new ArgumentNullException(nameof(operation), "Arithmetic operation cannot be null.");

}

return operation(x, y);

}

}

class Program

{

static void Main()

{

Console.WriteLine("Минко Ярослав");

Calculator<int> intCalculator = new Calculator<int>(

(x, y) => x + y,

(x, y) => x - y,

(x, y) => x \* y,

(x, y) => x / y

);

Console.WriteLine("Результат додавання: " + intCalculator.PerformOperation(5, 3, intCalculator.Add));

Console.WriteLine("Результат віднімання: " + intCalculator.PerformOperation(5, 3, intCalculator.Subtract));

Console.WriteLine("Результат множення: " + intCalculator.PerformOperation(5, 3, intCalculator.Multiply));

Console.WriteLine("Результат ділення: " + intCalculator.PerformOperation(5, 3, intCalculator.Divide));

Calculator<double> doubleCalculator = new Calculator<double>(

(x, y) => x + y,

(x, y) => x - y,

(x, y) => x \* y,

(x, y) => x / y

);

Console.WriteLine("Результат додавання: " + doubleCalculator.PerformOperation(5.5, 3.2, doubleCalculator.Add));

Console.WriteLine("Результат віднімання: " + doubleCalculator.PerformOperation(5.5, 3.2, doubleCalculator.Subtract));

Console.WriteLine("Результат множення: " + doubleCalculator.PerformOperation(5.5, 3.2, doubleCalculator.Multiply));

Console.WriteLine("Результат ділення: " + doubleCalculator.PerformOperation(5.5, 3.2, doubleCalculator.Divide));

}

}

**2.** **Дженеричний репозиторій з критеріями**

using System;

using System.Collections.Generic;

public class Repository<T>

{

private List<T> items;

public Repository()

{

items = new List<T>();

}

public delegate bool Criteria<T>(T item);

public void Add(T item)

{

items.Add(item);

}

public List<T> Find(Criteria<T> criteria)

{

if (criteria == null)

{

throw new ArgumentNullException(nameof(criteria), "Критерії не можуть бути нульовими.");

}

List<T> result = new List<T>();

foreach (T item in items)

{

if (criteria(item))

{

result.Add(item);

}

}

return result;

}

}

class Program

{

static void Main()

{

Console.WriteLine("Минко Ярослав");

Repository<int> intRepository = new Repository<int>();

intRepository.Add(1);

intRepository.Add(2);

intRepository.Add(3);

intRepository.Add(4);

intRepository.Add(5);

Repository<int>.Criteria<int> evenCriteria = x => x % 2 == 0;

List<int> evenNumbers = intRepository.Find(evenCriteria);

Console.WriteLine("Парні числа:");

foreach (int number in evenNumbers)

{

Console.WriteLine(number);

}

}

}

**3. Дженеричний кеш з викликами користувацьких функцій**

using System;

using System.Collections.Generic;

public class FunctionCache<TKey, TResult>

{

private Dictionary<TKey, CacheItem> cache = new Dictionary<TKey, CacheItem>();

public delegate TResult FuncDelegate(TKey key);

private class CacheItem

{

public TResult Result { get; set; }

public DateTime ExpirationTime { get; set; }

}

public TResult GetOrAdd(TKey key, FuncDelegate function, TimeSpan cacheDuration)

{

if (cache.TryGetValue(key, out var cacheItem) && DateTime.Now < cacheItem.ExpirationTime)

{

return cacheItem.Result;

}

else

{

TResult result = function(key);

cache[key] = new CacheItem { Result = result, ExpirationTime = DateTime.Now.Add(cacheDuration) };

return result;

}

}

}

class Program

{

static void Main()

{

Console.WriteLine("Минко Ярослав");

FunctionCache<int, string> stringCache = new FunctionCache<int, string>();

FunctionCache<int, string>.FuncDelegate function = key => key.ToString();

string result1 = stringCache.GetOrAdd(5, function, TimeSpan.FromSeconds(5));

Console.WriteLine(result1);

System.Threading.Thread.Sleep(3000);

string result2 = stringCache.GetOrAdd(5, function, TimeSpan.FromSeconds(5));

Console.WriteLine(result2);

System.Threading.Thread.Sleep(3000);

string result3 = stringCache.GetOrAdd(5, function, TimeSpan.FromSeconds(5));

Console.WriteLine(result3);

}

}

**4.** **Дженеричний планувальник завдань**

using System;

using System.Collections.Generic;

public class TaskScheduler<TTask, TPriority> where TPriority : IComparable<TPriority>

{

private class TaskItem

{

public TTask Task { get; set; }

public TPriority Priority { get; set; }

}

private readonly Queue<TaskItem> taskQueue = new Queue<TaskItem>();

public delegate void TaskExecution(TTask task);

public void AddTask(TTask task, TPriority priority)

{

taskQueue.Enqueue(new TaskItem { Task = task, Priority = priority });

}

public void ExecuteNext(TaskExecution executeTask)

{

if (taskQueue.Count > 0)

{

TaskItem nextTask = taskQueue.Dequeue();

executeTask(nextTask.Task);

}

else

{

Console.WriteLine("Немає завдань у черзі.");

}

}

public TTask GetFromPool(Func<TTask> initializeTask)

{

return initializeTask();

}

public void ReturnToPool(TTask task, Action<TTask> resetTask)

{

resetTask(task);

}

}

class Program

{

static void Main()

{

Console.WriteLine("Минко Ярослав");

TaskScheduler<string, int> scheduler = new TaskScheduler<string, int>();

Console.WriteLine("Введіть завдання з пріоритетами (натисніть Enter, щоб завершити):");

while (true)

{

Console.Write("завдання: ");

string task = Console.ReadLine();

if (string.IsNullOrEmpty(task))

break;

Console.Write("Пріоритет: ");

if (int.TryParse(Console.ReadLine(), out int priority))

{

scheduler.AddTask(task, priority);

}

else

{

Console.WriteLine("Недійсний пріоритет. Введіть дійсне ціле число.");

}

}

scheduler.ExecuteNext(Console.WriteLine);

string pooledTask = scheduler.GetFromPool(() => "Об'єднане завдання");

Console.WriteLine("Об'єднане завдання: " + pooledTask);

scheduler.ReturnToPool(pooledTask, t => Console.WriteLine("Скидання об’єднаного завдання: " + t));

}

}