**Паттерны программирования**

Задание 1 Создать CacheManager, который будет хранить временные данные в оперативной памяти. Метод AddToCache(string key, object value): добавляет данные. Метод GetFromCache(string key): получает данные. Доступ к кэшу должен осуществляться через единственный экземпляр класса.

Листинг программы:

public class CacheManager

{

private static readonly CacheManager \_instance = new CacheManager();

private readonly Dictionary<string, object> \_cache;

private CacheManager()

{

\_cache = new Dictionary<string, object>();

}

public static CacheManager Instance

{

get { return \_instance; }

}

public void AddToCache(string key, object value)

{

if (!\_cache.ContainsKey(key))

{

\_cache[key] = value;

}

}

public T GetFromCache<T>(string key)

{

\_cache.TryGetValue(key, out var value);

return (T)value;

}

}

public class User

{

public string Name { get; set; }

public int Age { get; set; }

}

class Program

{

static void Main(string[] args)

{

CacheManager cache = CacheManager.Instance;

var user1 = new User { Name = "Иван", Age = 11 };

var user2 = new User { Name = "Пашок", Age = 12 };

cache.AddToCache("user1", user1);

cache.AddToCache("user2", user2);

var userFromCache1 = cache.GetFromCache<User>("user1");

var userFromCache2 = cache.GetFromCache<User>("user2");

Console.WriteLine($"Имя: {userFromCache1.Name}, Возраст: {userFromCache1.Age}");

Console.WriteLine($"Имя: {userFromCache2.Name}, Возраст: {userFromCache2.Age}");

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| user1: { Name = Иван, Age = 11 }, user2: { Name = Пашок, Age = 12 } | user1: { Name = Иван, Age = 11 }, user2: { Name = Пашок, Age = 12 } |

Анализ результатов:

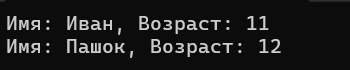


Рисунок 1.1 – Результат работы программы