Лабораторная работа 8

Структурная обработка исключений в .NET, проектирование собственных типов исключений

Задание 2: Работа с внутренними исключениями и логированием

Общие требования:

В каждой задаче необходимо:

1. Создать класс, содержащий метод, в котором вызывается сторонний метод (из

другого класса).

2. Вызов метода должен приводить к возникновению исключения, которое

перехватывается и оборачивается во внутреннее исключение (InnerException).

3. Логировать всю информацию о возникшем исключении, включая стек вызовов и

данные о внутреннем исключении.

4. Продемонстрировать работу с исключениями в Main через try-catch.

12. Ошибка при сериализации объекта

* Класс Serializer содержит метод Serialize(object obj), который выбрасывает
* SerializationException, если объект не поддерживает сериализацию.
* Класс DataSerializer перехватывает исключение и логирует его в
* SerializationOperationException.

Листинг программы:

using System;

public class SerializationException : Exception

{

public SerializationException() : base() { }

public SerializationException(string message) : base(message) { }

public SerializationException(string message, Exception innerException) : base(message, innerException) { }

}

public class SerializationOperationException : Exception

{

public SerializationOperationException() : base() { }

public SerializationOperationException(string message) : base(message) { }

public SerializationOperationException(string message, Exception innerException) : base(message, innerException) { }

}

public class Serializer

{

public void Serialize(object obj)

{

if (obj == null)

throw new SerializationException("Объект не может быть null");

if (!obj.GetType().IsSerializable)

throw new SerializationException($"Объект типа {obj.GetType().Name} не сериализуем");

}

}

public class DataSerializer

{

private readonly Serializer serializer;

public DataSerializer()

{

serializer = new Serializer();

}

public void PerformSerialization(object obj)

{

try

{

serializer.Serialize(obj);

Console.WriteLine("Сериализация успешно выполнена");

}

catch (SerializationException ex)

{

string log = $"Ошибка: {ex.Message}\nСтек вызовов: {ex.StackTrace}";

Console.WriteLine("Лог: " + log);

throw new SerializationOperationException("Не удалось выполнить сериализацию", ex);

}

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Запуск теста сериализации...");

DataSerializer dataSerializer = new DataSerializer();

try

{

object nonSerializable = new List<int>();

dataSerializer.PerformSerialization(nonSerializable);

}

catch (SerializationOperationException ex)

{

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

Console.WriteLine($"Детали внутреннего исключения: {ex.InnerException?.Message}");

}

Console.WriteLine("Тест завершен. Нажмите Enter для выхода.");

Console.ReadLine();

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| object nonSerializable = new List<int>(); | Запуск теста сериализации...  Сериализация успешно выполнена  Тест завершен. Нажмите Enter для выхода. |

Анализ результатов:

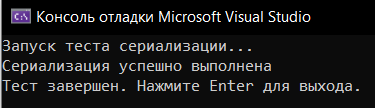


Рисунок 1 – Результат работы программы