Лабораторная работа №6
Организация обработки исключений

1 Цель работы

1.1 Изучить способы организации обработки исключений.

2 Литература

2.1 Гагарина, Л. Г. Технология разработки программного обеспечения : учебное пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул ; под ред. Л.Г. Гагариной. — Москва : ФОРУМ : ИНФРА-М, 2022. — 400 с. — (Среднее профессиональное образование). – URL: https://znanium.com/catalog/product/1794453 . – Режим доступа: по подписке. – Текст : электронный. – п.5.10.

3 Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).

3.2 Изучить описание лабораторной работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Реализация обработки исключений

5.1.1 Написать консольное приложение, которое запрашивает у пользователя два числа, выводит результат их деления и обрабатывает исключения:

- FormatException (если введен текст вместо числа).
- DivideByZeroException (деление на 0).
- общее исключение (Exception) с выводом stack trace.

5.1.2 Добавить логирование ошибок в файл errors.log, используя специализированное ПО (NLog (C#), Log4j (Java), logging (Python)).

5.2 Создание пользовательских исключений

5.2.1 Разработать класс NegativeNumberException для проверки отрицательных чисел.

Пример сценария:

- приложение принимает число (например, возраст),
- если число отрицательное, выбрасывает NegativeNumberException (пользовательский дочерний класс).

5.2.2 Добавить обработку с выводом понятного сообщения пользователю, использовав конструктор с параметром message для создания объекта NegativeNumberException

5.3 Использование finally и using

5.3.1 Написать код, который:

- открывает файл (FileStream в C#, File в Java/Python),
- считывает данные и обрабатывает их (например, ищет четные числа),
- гарантированно закрывает файл в блоке finally или через using (Idisposable).

Способы чтения файла:

- C#: using var file = new StreamReader(...).
- Java: try-with-resources.
- Python: with open(...) as file.

5.3.2 При чтении обработать FileNotFoundException. Проверить, что файл закрывается даже при возникновении ошибки.

5.4 Создание глобального обработчика исключений

5.4.1 Добавить в приложение глобальную обработку неотловленных исключений:

- C#: AppDomain.CurrentDomain.UnhandledException,
- Java: Thread.setDefaultUncaughtExceptionHandler,
- Python: sys.excepthook.

5.4.2 Записывать критичные ошибки в файл crash.log и выводить пользователю сообщение в следующем виде:

Произошла ошибка. Подробности в логах.

5.5 Обработка исключений в REST API

5.5.1 Создать HTTP-сервис (ASP.NET Core, Spring Boot, Flask), который принимает GET-запрос с параметром id. Например:

/user?id=123

Если пользователь не найден, возвращать HTTP 404.

Если id не число, возвращать HTTP 400 с JSON-ошибкой:

{ "error": "Invalid ID format", "statusCode": 400 }

5.5.2 Добавить логирование всех исключений.

6 Порядок выполнения работы

6.1 Выполнить все задания из п.5.

6.2 Ответить на контрольные вопросы.

7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Ответы на контрольные вопросы

7.4 Вывод

8 Контрольные вопросы

8.1 Что такое «исключение»?

8.2 Чем отличается обработка исключений через try-catch от простых условных проверок (if-else)? В каких случаях предпочтительнее использовать исключения?

8.3 Какие типы исключений обычно являются встроенными в объектно-ориентироанных языках программирования?

8.4 Как создать собственный класс исключения?

8.5 Зачем нужен блок finally и как он связан с using?

8.6 Как перехватить необработанное исключение в приложении на глобальном уровне?

8.7 Что такое «подавление исключений» и почему это антипаттерн?