Overview

Understanding custom exceptions

- Overview
- When to use
- Implementing

Define a custom calculation exception

Define a custom calculation operation not supported exception

Add additional custom property

Catch custom exceptions

Understanding Custom Exceptions

Overview

Use existing predefined .NET exception types where applicable, e.g.

- InvalidOperationException if property set/method call is not appropriate for current state
- ArgumentException (or derived) for invalid parameters

Wrap inner exception if appropriate

Don't use custom (or existing) exceptions for normal (non exceptional) logic flow

Understanding Custom Exceptions

When to use

Only create custom exception types if they need to be caught and handled differently from existing predefined .NET exceptions

E.g. want to perform special monitoring of a specific critical exception type

If building a library/framework for use by other developers so consumers can react to errors in your library

Interfacing with external API, DLL, service

Understanding Custom Exceptions

Implementing

Naming convention: ...Exception

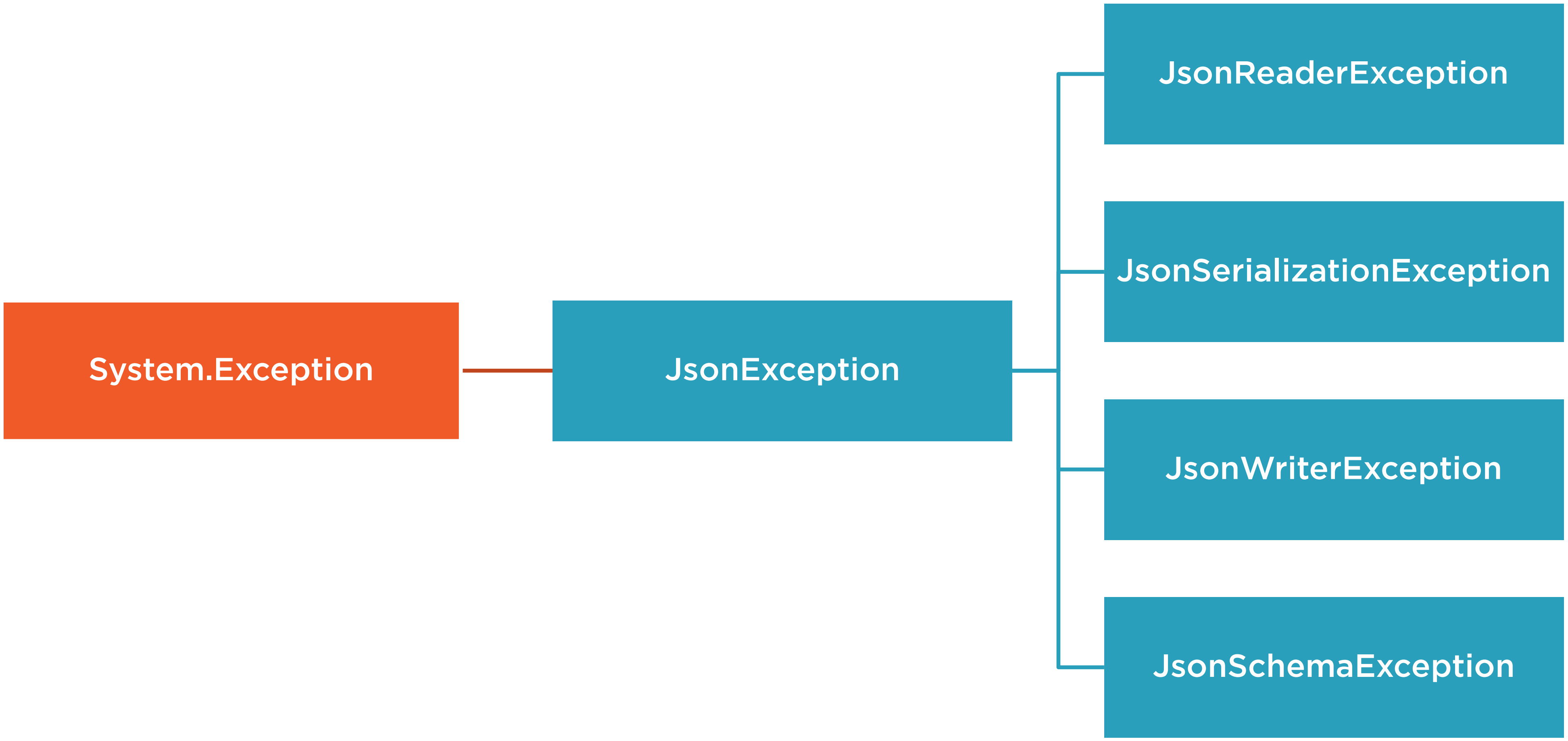
Implement standard 3 constructors

Add additional properties where needed

Never inherit from ApplicationException

Inherit from Exception (or your other custom exception)

Keep the number of custom exception types to a minimum



https://www.newtonsoft.com/json/help/html/T_Newtonsoft_Json_JsonException.htm

Summary

Understanding custom exceptions

- Use existing predefined .NET exception types where applicable
- Only create custom exception types if they need to be caught
- ...Exception

CalculationException

CalculationOperationNotSupportedExceptio n

public string Operation { get; }

Catch custom exceptions