What are Exceptions?

- **Definition**: An event that disrupts the normal flow of a program.
- Indicates an error or unexpected condition during execution.* **Examples**: `DivideByZeroException`, `NullReferenceException`, `FileNotFoundException`.
- **Why Handle?** Robustness, graceful degradation, separation of concerns.

The `try-catch-finally` Block

- **`try`**: Code that might throw an exception.
- **`catch`**: Catches and handles specific exception types.
- Multiple **`catch`** blocks (specific to general).
- **`catch (Exception ex)`**: Catches any exception.
- **`finally`**: Code that ***always*** executes (for cleanup).
- **Code Example**:

```
try { /* risky code */ }
catch (SpecificException ex) { /* handle */ }
catch (Exception ex) { /* general handle */ }
finally { /* cleanup */ }
```

Throwing Exceptions

- **`throw` keyword**: Used to explicitly raise an exception.
- Can re-throw a caught exception.
- **Example**:

```
throw new ArgumentException("Invalid input.");
```

Custom Exceptions

- **Purpose**: Provide specific, meaningful error information for application-specific problems.
- **How to Create**:
  - Inherit from `System.Exception`.
  - Follow naming convention (e.g., `MyCustomException`).
  - Provide standard constructors.
  - Mark with [`Serializable`] attribute.
- **Code Example**:

```
public class InvalidAgeException : Exception { /* ... */ }
```

`using` Statement

- **Purpose**: Ensures `IDisposable` objects (like file streams) are correctly disposed.
- Syntactic sugar for `try-finally` block.
- **Example**:

```
using (StreamReader reader = new StreamReader("file.txt"))
{
    // ...
} // reader is automatically disposed here
```

File I/O Operations: Overview

- **Namespace**: `System.IO`
- **`File` Class**: Static methods for common file operations.
- **`Directory` Class**: Static methods for directory operations.
- **Stream-based I/O**: `StreamWriter`, `StreamReader` for text files.

`File` Class Methods

- `File.Exists`(path)
- `File.ReadAllText`(path)
- `File.WriteAllText`(path, content)
- `File.AppendAllText`(path, content)
- `File.Copy`(source, destination)
- `File.Delete`(path)
- **Demo**: Basic file read/write.

Common Operation

- Checking File Existence

```
if (File.Exists("example.txt"))
{
    // File exists
}
else
{
    // File does not exist
}
```

- Creating a File

```
File.Create("example.txt");
```

Common Operation

- Copy File

```
File.Copy("source.txt", "destination.txt", true);
// true overwrites the destination file if it exists
```

- Move File

```
File.Move("source.txt", "destination.txt");
```

Directory Class Methods

- `Directory.Exists`(path)`
- `Directory.CreateDirectory`(path)
- `Directory.Delete`(path)
- `Directory.GetFiles`(path)
- `Directory.GetDirectories`(path)
- **Demo**: Creating/deleting directories.

`StreamWriter` and `StreamReader`

- **`StreamWriter`**: Writes characters to a file.
  - `WriteLine()`, `Write()`.
- **`StreamReader`**: Reads characters from a file.
  - `ReadLine()`, `ReadToEnd()`.
- **Always use with `using` statement!**
- **Demo**: Reading/writing large text files line by line.

Working with Text files

- Write to File
  - `StreamWriter`

```
using (StreamWriter writer = new StreamWriter("example.txt"))
{
  writer.WriteLine("Hello, World!");
  writer.WriteLine("Another line of text.");
}
```

Working with Text files

- Write to File
  - File.WriteAllText

```
File.WriteAllText("example.txt", "Hello, World!\nAnother line of text.");
```

  - File.WriteAllLines

```
string[] lines = { "First line", "Second line", "Third line" };
File.WriteAllLines("example.txt", lines);
```

Working with Text files

- Read From File
  - StreamReader
    - ReadLine

```
using (StreamReader reader = new StreamReader("example.txt"))
{
    string line;
    while ((line = reader.ReadLine()) != null)
    {
        Console.WriteLine(line);
    }
}
```

    - File.ReadAllText

Working with Text files

- File.ReadAllText

```
string content = File.ReadAllText("example.txt");
Console.WriteLine(content);
```

- File.ReadAllLines

```
string[] lines = File.ReadAllLines("example.txt");
foreach (string line in lines)
{
    Console.WriteLine(line);
}
```

Serialization and Deserialization

- **Serialization**: Converting an object's state into a storable/transmittable format.
- **Deserialization**: Reconstructing an object from its serialized form.
- **Purpose**: Persistence, data transfer.

JSON Serialization (`System.Text.Json`)

- **JSON**: JavaScript Object Notation (lightweight, human-readable).
- **`JsonSerializer.Serialize<T>(obj)`**: Object to JSON string.
- **`JsonSerializer.Deserialize<T>(jsonString)`**: JSON string to object.
- **Benefits**: Built-in, high-performance, widely used.
- **Demo**: Serialize/deserialize a custom object (e.g., **`Product`**).

Demo

```
public class DataClass
{
    public int Age { get; set; }
    public string Name { get; set; }
}
```

```
private static void jsonSerializeData()
{
    DataClass data = new DataClass { Age = 30, Name = "John Doe" };
    string jsonOutput = JsonSerializer.Serialize(data, new JsonSerializerOptions
    {
        WriteIndented = true // This option makes the JSON output more readable
    });
    Console.WriteLine("Serialized JSON:\n" + jsonOutput);

    //write it to a file
    string filePath = "data.json";
    File.WriteAllText(filePath, jsonOutput);
}
```

```
private static void jsonDeserializeData()
{
    string filePath = "data.json";
    if (File.Exists(filePath))
    {
        string jsonInput = File.ReadAllText(filePath);
        DataClass data = JsonSerializer.Deserialize<DataClass>(jsonInput);
        if (data != null)
        {
            Console.WriteLine("Deserialized Data:");
            Console.WriteLine($"Name: {data.Name}, Age: {data.Age}");
        }
        else
        {
            Console.WriteLine("Deserialization returned null.");
        }
    }
    else
    {
        Console.WriteLine("File not found: " + filePath);
    }
}
```

Assignment

- In Menu Program
  - Validate data input using **`try catch finally`**
  - Add save Button
    - To save list of employee as a json or text file
  - Add load button
    - To load the list from json or text file