**ImageSimilarityChecker Class Documentation**

**Overview**

The **ImageSimilarityChecker** class provides functionality to compare images within a specified folder to find similarities based on perceptual hashing. It calculates similarity scores between pairs of images and groups similar images together. The class uses multi-threading to perform comparisons without blocking the main application thread, ensuring responsive applications.

**Constructors**

* **ImageSimilarityChecker(string folderPath, double similarityThreshold)**
  + Initializes a new instance of the **ImageSimilarityChecker** with a specified folder path and similarity threshold.
  + **Parameters**:
    - **folderPath**: The path to the folder containing the images to be compared. Must not be null or empty.
    - **similarityThreshold**: The threshold for determining similarity, expressed as a percentage (1 to 100). Images with similarity scores above this threshold are considered similar.

**Properties**

* **int Progress**
  + Gets the current progress of the image comparison process as a percentage (0 to 100). This property is updated during the image comparison process.
* **List<Tuple<string, string, double>> Result**
  + Gets the results of the image comparison process. Each tuple contains the file paths of two similar images and their similarity score.

**Methods**

* **void CompareImagesInFolderAsync()**
  + Starts the asynchronous image comparison process in a new thread. Results will be stored in the **Result** property upon completion.
* **void Dispose()**
  + Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

**Events**

* **ProgressChangedHandler OnProgressChanged**
  + Occurs when the progress of the image comparison process is updated.
  + Event data: **ProgressChangedEventArgs** which contains the **Progress** property indicating the current progress percentage.

**Event Data Classes**

* **ProgressChangedEventArgs : EventArgs**
  + Contains data for the **OnProgressChanged** event.
  + **Properties**:
    - **int Progress** - Gets the progress of the image comparison process.

**Usage Example**

using (var checker = new ImageSimilarityChecker("path/to/images", 70))

{

checker.OnProgressChanged += (sender, e) =>

{

Console.WriteLine($"Progress: {e.Progress}%");

};

checker.CompareImagesInFolderAsync();

// Wait for the process to complete in a real application scenario.

// Access the results.

foreach (var result in checker.Result)

{

Console.WriteLine($"Images {result.Item1} and {result.Item2} are similar with a score of {result.Item3}");

}

}

**Notes**

* This class is designed to be used in environments where managing large sets of images is necessary.
* Ensure proper error handling and resource management by using the class within a **using** block or manually calling the **Dispose()** method to free up resources.

This documentation should be integrated into the software development documentation where the **ImageSimilarityChecker** class is utilized, providing a detailed reference for developers.

**ImageSimilarityChecker Class Documentation**

**Overview**

The **ImageSimilarityChecker** class provides functionality to compare images within a specified folder to find similarities based on perceptual hashing. It calculates similarity scores between pairs of images and groups similar images together. The class uses multi-threading to perform comparisons without blocking the main application thread, ensuring responsive applications.

**Constructors**

* **ImageSimilarityChecker(string folderPath, double similarityThreshold)**
  + Initializes a new instance of the **ImageSimilarityChecker** with a specified folder path and similarity threshold.
  + **Parameters**:
    - **folderPath**: The path to the folder containing the images to be compared. Must not be null or empty.
    - **similarityThreshold**: The threshold for determining similarity, expressed as a percentage (1 to 100). Images with similarity scores above this threshold are considered similar.

**Properties**

* **int Progress**
  + Gets the current progress of the image comparison process as a percentage (0 to 100). This property is updated during the image comparison process.
* **List<Tuple<string, string, double>> Result**
  + Gets the results of the image comparison process. Each tuple contains the file paths of two similar images and their similarity score.

**Methods**

* **void CompareImagesInFolderAsync()**
  + Starts the asynchronous image comparison process in a new thread. Results will be stored in the **Result** property upon completion.
* **void Dispose()**
  + Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

**Events**

* **ProgressChangedHandler OnProgressChanged**
  + Occurs when the progress of the image comparison process is updated.
  + Event data: **ProgressChangedEventArgs** which contains the **Progress** property indicating the current progress percentage.

**Event Data Classes**

* **ProgressChangedEventArgs : EventArgs**
  + Contains data for the **OnProgressChanged** event.
  + **Properties**:
    - **int Progress** - Gets the progress of the image comparison process.

**Usage Example**

csharp

Copy code

using (var checker = new ImageSimilarityChecker("path/to/images", 70)) { checker.OnProgressChanged += (sender, e) => { Console.WriteLine($"Progress: {e.Progress}%"); }; checker.CompareImagesInFolderAsync(); // Wait for the process to complete in a real application scenario. // Access the results. foreach (var result in checker.Result) { Console.WriteLine($"Images {result.Item1} and {result.Item2} are similar with a score of {result.Item3}"); } }

**Notes**

* This class is designed to be used in environments where managing large sets of images is necessary.
* Ensure proper error handling and resource management by using the class within a **using** block or manually calling the **Dispose()** method to free up resources.

This documentation should be integrated into the software development documentation where the **ImageSimilarityChecker** class is utilized, providing a detailed reference for developers.