Folder Synchronization Application

A simple C# console application that performs one-way synchronization between two folders, maintaining an identical copy of a source folder in a replica folder.

Features

- **One-way synchronization**: Replica folder becomes an exact copy of source folder
- **Periodic execution**: Automatically syncs at specified intervals
- **Comprehensive logging**: All operations logged to both console and file
- **MD5 hash comparison**: Efficiently detects file changes
- **Recursive directory support**: Handles nested folder structures
- **Command-line interface**: Easy to use and automate

Requirements

- .NET 6.0 or higher
- Windows, Linux, or macOS

How It Works

The application performs the following operations during each synchronization cycle:

1. **Copy new files** from source to replica
2. **Update modified files** in replica (detected via MD5 hash comparison)
3. **Create missing directories** in replica
4. **Remove extra files** from replica that don't exist in source
5. **Remove extra directories** from replica that don't exist in source

All operations are logged with timestamps to both the console and a log file.

Building the Project

Using Visual Studio

1. Open the solution file in Visual Studio
2. Build the solution (Ctrl + Shift + B)

Using .NET CLI

```
bash
```

```
dotnet build
```

Usage

```bash
FolderSync <source_path> <replica_path> <interval_seconds> <log_file_path>
```

Parameters

- `source_path` - Path to the source folder
- `replica_path` - Path to the replica folder (created if doesn't exist)
- `interval_seconds` - Synchronization interval in seconds (must be positive)
- `log_file_path` - Path to the log file (directory created if doesn't exist)

Example

```bash
FolderSync.exe "C:\MyDocuments" "D:\Backup\MyDocuments" 60 "C:\Logs\sync.log"
```

This command will:

- Synchronize `C:\MyDocuments` to `D:\Backup\MyDocuments`
- Run synchronization every 60 seconds
- Log all operations to `C:\Logs\sync.log`

The application will run continuously until you press ENTER to stop it.

Example Output

```
[10/27/2025 2:30:15 PM] Application started.
[10/27/2025 2:30:15 PM] Synchronization interval is 60 seconds
[10/27/2025 2:30:15 PM] Starting synchronization...
[10/27/2025 2:30:15 PM] Created replica folder: D:\Backup\MyDocuments
[10/27/2025 2:30:15 PM] Copied: C:\MyDocuments\file1.txt -> D:\Backup\MyDocuments\file1.txt
[10/27/2025 2:30:15 PM] Created directory: D:\Backup\MyDocuments\Photos
[10/27/2025 2:30:15 PM] Copied: C:\MyDocuments\Photos\image.jpg -> D:\Backup\MyDocuments\Photos\image.jpg
[10/27/2025 2:30:15 PM] Synchronization completed successfully.
Press ENTER to stop...
```

Project Structure

```
Folder_Synchronization_App/
├── Program.cs           # Main entry point and Timer setup
├── Logger.cs            # Logging functionality
├── FileComparer.cs      # MD5 hash calculation and file comparison
└── FolderSynchronizer.cs  # Core synchronization logic
```

Error Handling

The application includes comprehensive error handling:

- **Invalid arguments**: Shows usage instructions
- **Missing source folder**: Logs error and skips synchronization
- **File access errors**: Logged but doesn't crash the application
- **Log file errors**: Falls back to console-only logging

Technical Details

- **File comparison**: Uses MD5 hashing for efficient change detection
- **Size optimization**: Compares file sizes before calculating hashes
- **Thread safety**: Logger uses locks to prevent concurrent write issues
- **Recursive sync**: Automatically handles nested directory structures
- **Timer-based**: Uses `System.Threading.Timer` for periodic execution

Notes

- This is a one-way synchronization tool - changes in replica are overwritten
- Large files are handled efficiently through stream-based hashing
- The application creates necessary directories automatically
- Synchronization runs immediately on startup, then at specified intervals

License

This project is created as a test task demonstration.

Author

Created as part of a Junior QA in Dev job application.