1. Does it have only necessary code? (400 pts)
Score: 300/400
The code is functional but contains some redundancy. For example:
MakeVideo has multiple methods for similar tasks (e.g., convertImageToVideo and convertImageToVideoWithSilencedAudio). These could be merged with parameters to toggle audio.
Some utility methods (e.g., deleteFile) are repeated or could be further generalized.
Unused or experimental code (e.g., generateAudioFrom64) should be removed.

2. Does it properly use OOP? (600 pts)
Score: 500/600
Strengths:
Classes are well-segregated (e.g., MakeVideo, IaFunctions, ExifFunctions).
Static methods are used appropriately for utility functions.
Weaknesses:
Over-reliance on static methods limits polymorphism and inheritance.
Some classes (e.g., IaFunctions) are overly monolithic and could be split into smaller, focused classes (e.g., AudioGenerator, ImageProcessor).

3. Is the code reusable? (1,000 pts)
Score: 800/1,000
Pros:
Modular design allows methods like normalizeVideo or generateAudioFromText to be reused across projects.
FFmpeg and OpenAI API wrappers are decoupled from business logic.
Cons:
Hardcoded values (e.g., video duration 5 seconds in convertImageToVideo) reduce reusability.
Tight coupling between IaFunctions and MakeVideo (e.g., direct calls to base64ToDescription).

4. Is the code flexible? (1,000 pts)
Score: 800/1,000
Flexible Aspects:
Configurable parameters (e.g., width, height) allow adaptation.
Supports multiple file types (images/videos).
Inflexible Aspects:
Audio/video processing logic is rigid (e.g., fixed sampling rates, codecs).
No dependency injection (e.g., ChatGPTKey is hardwired).

5. Does it have bugs? (800 pts)
Score: 70/800
Potential Issues:
No null checks for file paths (e.g., path + "/" + fileName could fail).
Error handling in IaFunctions is inconsistent (e.g., empty returns instead of exceptions).
Memory leaks: Open streams (e.g., ProcessBuilder) are not always closed.
Robustness:
Most edge cases (e.g., missing files) are handled gracefully.

6. Is the code scalable? (800 pts)
Score: 700/800
Scalability:
Threading/parallelism is absent; large batches of files could slow down execution.
OpenAI API calls are synchronous.
Improvements:
Use async/await for API calls.
Implement a task queue for FFmpeg processes.

7. Does it have comments? (800 pts)
Score: 500/800
Missing:
No JavaDoc or method-level comments explaining purpose/parameters.
Complex logic (e.g., generateCollage FFmpeg filters) lacks inline explanations.
Good:
Some debug prints (e.g., System.out.println("Collage made")) act as pseudo-comments.

8. Is the code a huge mess or neat? (1,000 pts)
Score: 900/1,000
Organization:
Clean separation of concerns (e.g., ExifFunctions for metadata, MakeVideo for FFmpeg).
Consistent naming conventions.
Messy Parts:
Long methods (e.g., generateCollage).
Mixed abstraction levels (e.g., IaFunctions handles both HTTP calls and string normalization).