Simple Document store

Context

A certain web application regularly generates and store documents (can be of any type including but not restricted to pdf, doc or text). In the initial implementation of the application, the reports were directly stored on the local filesystem.

Currently, the application is deployed on AWS. To save storage cost and to increase reliability, developers decided to use S3 rather than the file system to store the documents.

In the future, the application may be deployed to other cloud systems (like Azure, Google Cloud) and even the client's local server. Anticipating this need, developers want to use a library/interface to store files and don't want to directly use a particular storage technology (like S3 or even local storage).

Task

Core Library

You are given the task to design and implement a simple document/file storage library. Note that the task requires a library and not a web service.

The library must abstract out and encapsulate the underlying storage technology so that the users of the library (like the aforementioned web application developers) do not have to worry about it.

The user of the library must be able to switch between different storage technologies (like S3, filesystem, Azure Blob Storage, Database like MongoDB, SQLite, etc.) with minimum changes. For example, the aforementioned web application was storing documents in the local filesystem and the developers now want to change to storing to S3. The developer should be able to do it by simply making some changes to a configuration file. It should not need other modification in their main application.

The library must also be extensible i.e. a future developer must be able to introduce new storage engine (like a different database or completely different cloud service) without modifying your core library.

The library should support common storage-related operations like - saving the document, open document, get document attributes (like creation time, last update time, size, etc.), rename the document, delete the document, etc.

Prefered language is Java but you may use other languages if necessary.

Local FileSystem Storage Engine

You must Implement at least a storage engine based on a local filesystem to demonstrate that your library works properly.

One Alternate Storage Engine

You should also implement an alternate storage engine to show that your library is actually extensible. You can choose any other storage backend like S3, Azure Blob storage, MongoDB, RDBMS like SQLite, MySQL, etc.

Documentation and Tests

Please also add documentation and tests. While documenting please focus on what your library does and how can be used including its initialization, configuration, general usage, and any exceptions which may need to be handled. Specifically, mention how are you are fulfilling the following two requirements mentioned before:

i) The user of the library must be able to switch between different storage technologies (like S3, filesystem, Azure Blob Storage, Database like MongoDB, SQLite, etc.) with minimum changes.
ii) The library must also be extensible i.e. a future developer must be able to introduce new storage engine (like a different database or completely different cloud service) **without modifying** your core library.

Note: We give a lot of weight to the quality of the documentation you provide. So please give it adequate importance yourself. For more read this.

Bonus task - Directory

Implement support for directories/folder.

FAQ

Q. What is the naming convention of the document to be stored?

A. These are normal documents and hence name can be any valid filename.

Q. I have implemented the library and it's mostly extensible but add more storage engine a future developer will have to make a couple of small changes. Is that fine?

A. An ideal solution will require the future developer to make absolutely no changes to the core library you provided to add more storage options.

   Think of storage engines like addon or plugins. You do not have to make changes to Chrome or Firefox codebase to install add-ons. Similarly, to install more storage engines, a developer should not need to modify your library.

   The library should follow the open-closed principle.
   https://en.wikipedia.org/wiki/Open%E2%80%93closed_principle

Q. What will happen if I try to store a file which is already stored? Should file names be unique?

A. The decision is up to you. You can choose the replace the old file, or throw an error, or store the new content while preserving the old content (version control), or any other solutions you can think of.