The Lob Client

At Lob, we believe we are only as good as our partner network. That is why supporting our print partners in doing their best work is a core tenant of what we do here. In this question we'll consider what a Lob client (software running on a computer or a tablet used by a print-partner employee) might look like, and particularly the functionality used to render (present) a PDF file inside that client for quality assurance purposes.

When rendering a PDF, it might actually be beneficial not to use a single, pre-defined rendering engine (E.g., Adobe, Foxit, PDFium), but for the software to have the flexibility to dynamically select the engine for best performance and reliability given the capabilities required. For example, Foxit supports annotations, but is known to suffer some loading time issues with heavier files. MuPDF is fast, but its annotation capabilities are fairly rudimentary.

For the purpose of this exercise, we'll assume all that:

1. All PDF engines support the operations:

   > **LoadDocument** - load a PDF document from a file or memory buffer.
   > **CloseDocument** - close a loaded PDF document.
   > **GetPageCount** - get the number of pages in a loaded PDF document.
   > **GetPageSize** - get the size of a specific page in a loaded PDF document.
   > **SaveAs** - save a modified PDF document to a file or memory buffer.
   > **RenderPage** - render a specific page in a loaded PDF document to a bitmap or device context.

2. Different PDF engines **only** differ in whether or not they support *adding and deleting annotations, encryption, decryption, and granular search (search within*

a specific region of the file). When having to render a PDF, some, all or none of these operations might be needed.

Using TypeScript and node.js, your task is to code the class hierarchy that supports the API:

```
function renderPDF(pathToPDFFile: string, fileInfoByte: number): PDFWrapper {
// Implementation goes here
}
```

In this function:

1. pathToPDFile is the location of the file.
2. fileInfoByte is a **binary** number. Its value represents which of **(annotation, encryption/decryption, granular search)** is needed in order to work with the file. It uses its 3 LSBs for that.
3. PDFWrapper is an object that contains a reference to the structure typically returned by the selected engine's LoadDocument method. It also exposes methods reflecting the operations supported by that specific engine.

You are responsible for:

1. A working solution. You will need to provide one or more tests that take the PDF file **we provided** and validate that PDFWrapper contains something you can render. The tests will also have to show how PDFWrapper exposes **only** the methods applicable to the specific rendering engine that was dynamically selected given fileInfoByte.
2. An implementation that allows you to add support for a new rendering engine with specific capabilities that don't necessarily exist in other engines you have, **without altering any code except for the specific abstraction of the new engine.**

During your team interview, you will be asked to demo and explain how your code handles these requirements.

During your team interview, you will be asked to extend your engine collection in a different way. Please make sure your design is properly extensible.

You are not responsible for:

1. The correctness of fileInfoByte given an input PDF file. You can assume that fileInfoByte will always represent the correct set of capabilities required to work with the file.
2. The correct matching of engine to capability. We are interested in your class design to support maintainability of this code. The file we provided doesn't actually require any of those capabilities to render properly. For the purpose of this exercise, you can use the following "fake" collection of PDF engines, with PDF.js as the actual underlying engine for all of them.

Engine Name	Annotation	Encrypt / Decrypt	Search
SOSOPDFENGINE			
OKPDFENGINE			X
ALSOOKPDFENGINE		X	
BETTERPDFENGINE	X		X
BESTPDFENGINE	X	X	X

We provided you with a starter pack to get you going, which can be found here.

Good luck!