Description of VFS

VFS (Virtual File System) is a file format which makes it possible to store any files and directories just into one file. No compression is used for storing the data, but rather the data is stored normally (clean text). Using VFS in other programming languages and on other operating systems to ease the use of transferring a large amount of files and directories is the main target of this project.

Before I've created VFS there was only a class which could create a new file and extract another file. With VFS there are much more possibilities: Once the file is imported you can read files, create new files, delete existing files and much more, but you don't need to create a new file for applying a change to your existing file.
There are two different versions of this format: Both versions contain a header with information about the directory structure, but both versions differs from each other. The old version (first developed) separates the bytes of the files with a special sequence of bytes which depends on two parameters. Firstly, how much bytes are used (default: 128) and the particular byte which is used to generate this sequence (default: "-"). For instance, a default sequence looks like this:

```
 ----------------------------------------------------------------------------
---------------------------------------------------- (128 x „-“).
```

Meaning that together with the header the length of the final file is greater than the length of all files added together. Consequently, if you have a large amount of files, you're wasting a lot of storage for storing these bytes to separate between the files. Furthermore in some cases a conflict may occur if the file you want to use contains a similar sequence of bytes with the same or a higher length. But it is possible to modify these parameters (the greater the length is, the more storage is used). For little XML-files or files which contains settings you don't need to change these parameters.

The new version of this format is completely different, because the bytes of the files weren't separated anymore, but rather the positions were additionally saved in the header. Now the files can be read at runtime without a special sequence to separate the bytes.
Besides at the beginning of the file there is another special sequence of bytes: `{***VHP***}` which is needed to differentiate between these two versions.
In the opposite to the old version the header is saved in base64-format which makes it easier to get just the header and not content from any files and at the end of the header there is an "^" attached to mark the end of the header.
Another important difference is that the old version is reading the whole content directly into your RAM and changes just affect the content that is stored into the RAM. It's possible to do a "save" call at the end of your session to generate the file new with all changes you made. For files with a low length it's very useful, because otherwise for more changes the file will be written multiple times. Saving at the end of the session just needs one writing access to the physical hard drive. It's also possible to save directly when you done something, e.g. writing a file. You are able to decide which option do you want use.
The process to import the file is a bit different, because just the header will be imported and nothing more. If you want to get the content of a file the start – and end position will be used to read the bytes byte-wise with a buffer, but there is a limit at 1 GB. It's enough for little files with text, because it's not necessary to get the content of big files, e.g. ISO-files in plain text. But VFS can process big files, e.g. extract, because it works buffer-wise (50 GB files and greater are possible too).
If you're adding a new file, this file isn't stored in your memory. It is stored in your workspace (special directory like a temporary directory). Consequently, if you do a save call or the save method is called automatically all files and directory will be extracted to the workspace, except the files you want to add. Then a new file will be generated with all files and directories in the workspace directory. It's important that you don't store data in the workspace directory, because if you do so, the data will be observed too.
Finally, you can choose if you want to save directly or at the end of your session.