Module **setup**

▶ EXPAND SOURCE CODE

Classes

```
class FileHandler (main_path: str)
```

Handles files, specifically those related to the recorded songs wavs and zips.

Creates a new `FileHandler` and sets the path for the zips folder and unzipped songs folder.

Args:

`main_path` : Path of the main script from which the relative paths of resources are calculated.

▶ EXPAND SOURCE CODE

Methods

```
def get_dest_folder_path(self) -> str
```

Getter for the `_dest_folder_path` attribute.

Returns:

The path in which to put the unzipped files.

▶ EXPAND SOURCE CODE

```
def get_list_of_songs(self)
```

Gets the list of available songs so that the user can choose the one he prefers.

Returns:

A `list` of strings containing the name if the songs and their indexes.

Raises:

`FileHandlingException` : if no songs have been found.

▶ EXPAND SOURCE CODE

```
def get_number_of_songs(self) -> int
```

Getter for the `_number_of_songs` attribute.

Returns:

The number of songs.

▶ EXPAND SOURCE CODE

```
def get_number_of_tracks(self, song_index: int) -> int
```

Gets the number of tracks given the index of a song.

Args:

`song_index` : Index of the song of which to get the number of tracks.

Returns:

The number of tracks of a song.

▶ EXPAND SOURCE CODE

```
def get_source_folder_path(self) -> str
```

Getter for the `_source_folder_path` attribute.

Returns:

The path of the `src` folder.

▶ EXPAND SOURCE CODE

```
def get_tracks(self, song_index: int) -> tuple
```

Gets all the tracks of a song as numpy arrays, ready to be processed. Each track has its own stereo .wav file, so each file has to be read separately and converted to mono.

Args:

`song_index` : Index of the song of which to get the tracks.

Returns:

A `list` of `np.ndarray` containing the tracks and the sample rate of the wave file read (assuming all files of the song have the same sample rate).

▶ EXPAND SOURCE CODE

```
def unzip_files(self)
```

Unzips any zip file found in the `resources/test_songs` folder. If the zip file has already been unzipped, it skips it.

Raises:

`FileHandlingException` if there are no .zip files in the songs folder.

▶ EXPAND SOURCE CODE

```
class SetupHandler
```

Singleton that handles the setup phase of the application and stores audio parameters.

Constructor to not be accessed directly (Singleton pattern). It initializes the `FileHandler` to `None` and creates an empty `dict` that will contain the audio parameters.

Class Attributes:

`__main_path` : Absolute path from root of the `src` folder from which to gather files.

`__file_handler` : Object to handle file reads and writes.

`__audio_producer` : Object used to produce audio in chunks.

`__audio_parameters` : Audio parameters used for processing and during setup.

▶ EXPAND SOURCE CODE

Static methods

```
def get_instance() -> SetupHandler
```

Returns the instance of the singleton, creating it if the method has never been called before.

Returns:

The currently running class instance.

▶ EXPAND SOURCE CODE

Methods

```
def get_audio_parameters(self) -> dict
```

Getter for the `__audio_parameters` attribute.

Returns:

A `dict` containing the audio parameters.

▶ EXPAND SOURCE CODE

```
def get_audio_producer(self)
                       -> modules.audio_producer.AudioProducer
```

Getter for the `__audio_producer` attribute.

Returns:

The `AudioProducer` object attribute.

▶ EXPAND SOURCE CODE

```
def get_audio_streams(self) -> tuple
```

Creates a new audio stream based on the audio parameters of the `SetupHandler` . Specifically, it creates an input stream if the user chose to use live audio, or an output stream if the user chose to use recorded audio and wants to hear the song while it's being processed.

Returns:

A `tuple` containing the input and output stream (`None` if they aren't created).

▶ EXPAND SOURCE CODE

```
def set_audio_parameters(self, audio_parameters: dict)
```

Sets the audio parameters to a given dictionary.

Args:

`audio_parameters` : new dictionary to set.

▶ EXPAND SOURCE CODE

```
def set_main_path(self, main_path: str) -> None
```

Creates the `FileHandler` instance with a given string path. The path has to be fed by the main script as it is the point from here each relative path is calculated (see `FileHandler` docs for more info).

Args:

`main_path` : Path of the `src` folder.

▶ EXPAND SOURCE CODE

```
def setup(self) -> dict
```

Gets the info needed as user input and fills the audio parameters dictionary accordingly, along with `AudioProducer` object. See the specific private input functions of this class for more info about the user input.

Returns:

The audio parameters dictionary filled according to user input.

▶ EXPAND SOURCE CODE