FUNCTIONAL REQUIREMENTS

Name	Create user
Identifier	RF1
Operation summary	It asks for the required information that allows to create the user (name, password and age), once it asks for the name it verifies if it is a valid one (without spaces) and if it is already taken by another user. After the validation it creates a new User and adds it to the users array if there is space available (It is important to specify that once a new User its initialized, its category is "NEWBIE"). Finally it prints a message that informs the user if the operation was a succeed or not.
Inputs	String name (username of the user) String password (password of the user) String age (age of the user)
Expected outputs	Create a new User which it's going to be located in one of the 10 positions of the users array. If the operation is a succeed, a variable named "control" keeps track of how many users are being created.

Name	Show user(s)
Identifier	RF2
Operation summary	It prints the name, age and category of all users in the users array. This thanks to the "control" variable mentioned in the RF1. This method has a loop and sends the advance variable (of the loop) into a method in charge of asking the User object in the position of the index, all the information and returning it into a message that later, is printed in the User interface.
Inputs	
Expected outputs	It prints the name, age and category of all the existing users up to that moment in a specific format.

Name	Add a song to the pool
Identifier	RF3
Operation summary	It asks for the title, artist, release date, genre and duration of a song and then it adds it to the pool array if there is space available. There is some a selection menu that allows the user to select (with a number) the genre of the song, to avoid errors. At the end of the process it prints if the operation was a succeed or a failure. It also allows for (if there are at least one user) to say which user uploaded the song and updates its category.
Inputs	String title (title of the song) String artist (whoever made the song) String date (release date of the song) int genre (option from the menu that sets the String for the genre of the song) int duration (duration in seconds of the song)
Expected outputs	Create a new song and add it to the pool array. There is also a variable "controlS" that keeps track of how many songs are being created.

Name	Show song(s)
Identifier	RF4
Operation summary	It prints the title, artist duration (in a specified format) and genre of all songs in the pool array. This thanks to the "controlS" variable mentioned in the RF1. This method has a loop and sends the advance variable (of the loop) into a method in charge of asking the Song object in the position of the index, all the information and returning it into a message that later, is printed in the User interface.
Inputs	
Expected outputs	It prints the title, artist, duration and genre of all the existing songs up to that moment in a specific format.

Name	Create playlists
Identifier	RF5
Operation summary	It asks for the name of the playlist and then, for the type of playlist the user wants to create (with a menu selection) whether is private, restricted or public. If its private, it asks for the name of the only user that will have access to that playlist. If the user doesn't exist, the playlist isn't initialized. If its restricted it will ask for the name of the 5 users that will have access to that playlist (the user has to type at least one username). In this case it doesn't affect if the user exists or not, it simply doesn't add it to the array of users. If the playlist is public, it just creates it if there is space available. In this case, the methods overcharge the preexisting one in the Playlist class in order to create each type. Additionally, when a new playlist is created its duration is 0 and its genre "UNKNOWN".
Inputs	String name (name of the playlist) String username (if it's a private Pl) String [] userNames (if it's a restricted Pl)
Expected outputs	Create one of the 3 options of playlist and add it to the collection array. There is also a "control" variable that keeps track of how many playlist are being created.

Name	Add song to playlist
Identifier	RF6
Operation summary	It asks for which song the user wants to add to a playlist and if it exists, it asks for the name of the playlist. Once again, if it exists, it takes the reference of that Song object and send it to the corresponding playlist who receives it into an array of Song objects called songs. It also updates automatically the duration and genre list of the playlist that receives the song.
Inputs	String title (title of the song) String name (name of the playlist)
Expected outputs	Add a specific song object from the pool array to the songs array of a given playlist.

Name	Show playlist(s)
Identifier	RF7
Operation summary	It prints the name, duration (in a specified format) and genres of all playlist (in general) and: For a private playlist, it also prints the owner of the playlist. For a restricted playlist, the names of the owners (in that moment it could be none, one, or anything up to 5). For a public playlist, its rate (a grade gave by the users). This is made with a general method in the Playlist class which is inherit by all the 3 types and then, overwritten.
Inputs	
Expected outputs	Display the name, duration and genres of all playlist (created up to that moment) and the owners or rating, depending on the playlist type.

Name	Grade a playlist
Identifier	RF8
Operation summary	It allows to grade a public playlist in any moment. It accumulates the scores values and the amount of evaluations and divides the first by the second one in order to get the total score of the playlist.
Inputs	int score (positive integer between 1 and 5)
Expected outputs	Display the name, duration and genres of all playlist (created up to that moment) and the owners or rating, depending on the playlist type.