SEMINAR 5

Contents

1. OBJECTIVES

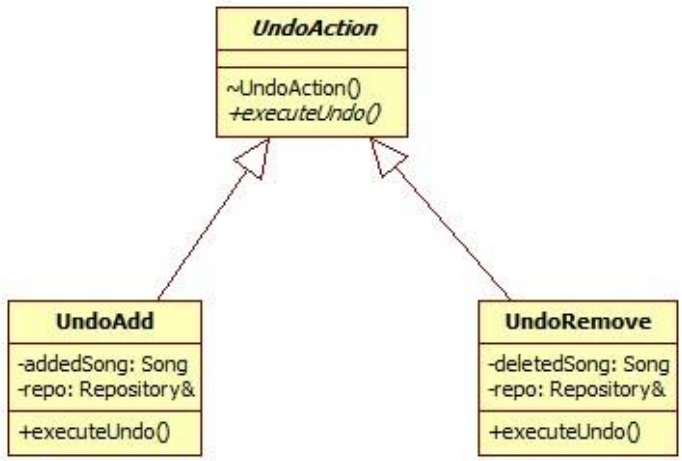
- Use inheritance and polymorphism to add an undo functionality.
- Work with smart pointers: unique_ptr.

2. PROBLEM STATEMENT

Extend your playlist application such that it offers an **undo** functionality: all the operations performed on the song repository can be undone (add, remove, update). Solve this requirement using an abstract class **UndoAction**. For each different action (add, remove, update) you must create a new class, that inherits from UndoAction (see the UML diagram below).

The Controller should be responsible with keeping undo actions. These will be stored as smart pointers (unique_ptr).

3. UML DIAGRAM



4. SOURCE CODE

----------**Undo.h**----------

```
#pragma once

#include "Song.h"
#include "Repository.h"

/*
	Generic class for an undo action.
	For each type of action (add, delete, update), a new class will be created,
inheriting from this UndoAction.
*/
class UndoAction
{
public:
	virtual void executeUndo() = 0;
	// virtual destructor!
	virtual ~UndoAction() {};
};

class UndoAdd : public UndoAction
{
private:
	Song addedSong;
	Repository& repo;	// we keep a reference to the repository to be able to undo
the action
public:
	UndoAdd(Repository& _repo, const Song& s): repo{ _repo }, addedSong { s } {}

	/*
```

```
		For the add operation, the reverse operation that must be executed is
"remove".
	*/
	void executeUndo() override
	{
		this->repo.removeSong(addedSong);
	}
};

class UndoRemove : public UndoAction
{
private:
	Song deletedSong;
	Repository& repo;
public:
	UndoRemove(Repository& _repo, const Song& s): repo{ _repo }, deletedSong{ s } {}

	void executeUndo() override
	{
		this->repo.addSong(deletedSong);
	}
};
```

----------Controller.h----------

```
class Controller
{
private:
	Repository repo;
	FilePlaylist* playList;
	SongValidator validator;
	// a vector of unique_ptr of undo actions;
	// pointers are required, as we need polymorphism;
	// each add/remove action (on the repository) will be recorded in this vector
	std::vector<std::unique_ptr<UndoAction>> undoActions;

public:
	Controller(const Repository& r, FilePlaylist* p, SongValidator v) : repo{ r },
playList{ p }, validator{ v } {}

	Controller(const Controller& ctrl) = delete;		// controller cannot be
copied now, because it contains unique_ptr
	void operator=(const Controller& ctrl) = delete;	// same for assignment

	Repository getRepo() const { return repo; }
	PlayList* getPlaylist() const { return playList; }

	/*
		Adds a song with the given data to the song repository.
		Throws: SongException - if the song is not valid
				DuplicateSongException - if there is another song with the
same artist and title
				Throws: FileException - if the repository file cannot be
opened.
	*/
	void addSongToRepository(const std::string& artist, const std::string& title,
double minutes, double seconds, const std::string& source);
```

```
	void removeSongFromRepository(const std::string& artist, const std::string&
title);

	// undoes the actions performed on the repository
	void undo();


	/*
		Adds a given song to the current playlist.
		Input: song - Song, the song must belong to the repository.
		Output: the song is added to the playlist.
	*/
	void addSongToPlaylist(const Song& song);

	// Adds all the songs from the repository, that have the given artist, to the
current playlist.
	void addAllSongsByArtistToPlaylist(const std::string& artist);

	void startPlaylist();
	void nextSongPlaylist();

	/*
		Saves the playlist.
		Throws: FileException - if the given file cannot be opened.
	*/
	void savePlaylist(const std::string& filename);

	/*
	Opens the playlist, with an appropriate application.
	Throws: FileException - if the given file cannot be opened.
	*/
	void openPlaylist() const;
};
```

----------Controller.cpp----------

```
// ...
void Controller::undo()
{
	if (undoActions.empty())
	{
		throw RepositoryException{ "There are no more actions to undo." };
	}

	try
	{
		undoActions.back()->executeUndo();
		undoActions.pop_back();
	}
	catch (RepositoryException& e)
	{
		cout << e.what() << endl;
	}
}
// ...
```

----------UI.h----------

```
#pragma once
#include "Controller.h"

class UI
{
private:
	Controller& ctrl;	// reference to the controller (controller cannot be copied!)

public:
	UI(Controller& c) : ctrl(c) {}

	void run();

private:
	static void printMenu();
	static void printRepositoryMenu();
	static void printPlayListMenu();

	void addSongToRepo();
	void removeSongFromRepo();
	void displayAllSongsRepo();
	void addSongToPlaylist();
	void addAllSongsByArtistToPlaylist();
	void savePlaylistToFile();
};
```

The entire source code can be found on www.cs.ubbcluj.ro/~iuliana/oop**.**