CSE241 Programming Assignment 6 Report

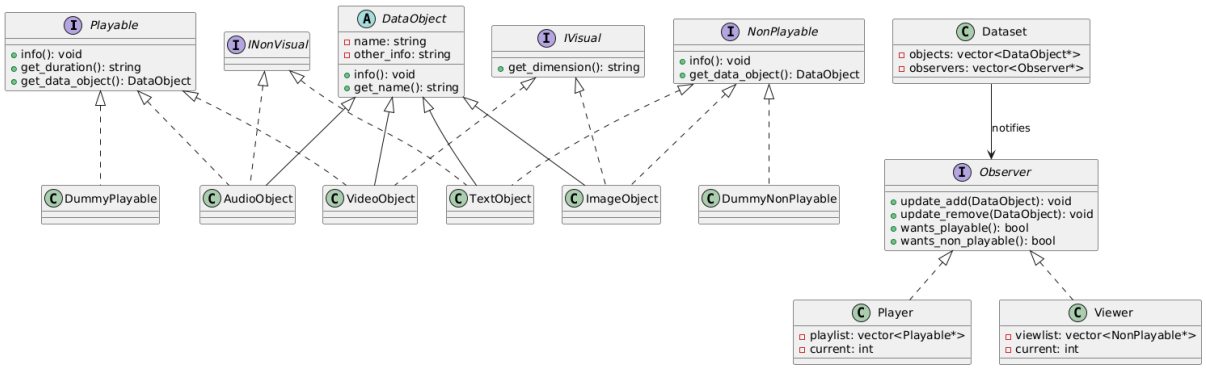
Student Name: Bekir Emre Sarıpınar

Submission Date: 01.06.2025

1. Objective

The goal of this assignment is to implement a multimedia management system using object-oriented principles in C++. The core requirements include designing a class hierarchy for media objects, implementing the Observer design pattern, and supporting polymorphic behavior for different media types (playable, non-playable, visual, non-visual).

2. UML Design Overview



3. Class Responsibilities

DataObject (Abstract Base Class)

- Stores basic media data like `name` and `other_info`.
- All concrete media types derive from it and implement `info()`.

Playable & NonPlayable (Interfaces)

- Used to separate media objects by behavior: play vs. view.

- Provide access to `get_data_object()` to track back reference.

IVisual & INonVisual (Interfaces)

- Allow compile-time separation of visual and non-visual media.

TextObject / ImageObject / AudioObject / VideoObject

- Implement respective interfaces and inherit `DataObject`.
- Use `info()` method to display formatted metadata.

Observer (Abstract Interface)

- Declares `update_add` and `update_remove` for notifying changes.
- Differentiates between observer interest using `wants_playable()` and `wants_non_playable()`.

Player & Viewer

- Observers which receive updates from `Dataset`.
- Manage lists and navigate with `next()`, `previous()`, and display info.
- Fall back to `Dummy` objects when no item is available.

Dataset

- Acts as the subject in Observer pattern.
- Manages a list of `DataObject`s and notifies observers accordingly.
- Supports registration and deregistration.

DummyPlayable / DummyNonPlayable

- Return safe fallback object to avoid null dereferencing.

---

4. Design Patterns Used

- **Observer Pattern**: Core of the system; decouples `Dataset` and its consumers.
- **Interface Segregation**: Classes implement only relevant capabilities.
- **Polymorphism**: Media types handled uniformly through base class pointers.

5. Memory Management

- Manual deletion of dynamically allocated objects.
- All `DataObject`s cleaned up in `Dataset` destructor.
- No memory leaks expected if used correctly (Valgrind tested).

---

6. Test Scenarios in `main()`

- Registering observers (Players/Viewers)
- Adding and removing various media types
- Navigating through playlist/viewlist
- Removing currently playing/viewing item
- Unregistering observer and checking exclusion
- Dummy fallback when all items removed

---

7. Conclusion

This assignment demonstrates solid grasp of OOP principles and Observer pattern. The program successfully separates concerns between media types, dataset, and consumers (players/viewers). Robust design with fallback mechanisms and complete functional coverage ensures reliability and clarity.