```cpp
// ============================================================================
// GERARCHIA MEDIAITEM - SISTEMA BIBLIOTECA MULTIMEDIALE
// Programmazione ad Oggetti - Università di Padova
// ============================================================================

#ifndef MEDIAITEM_H
#define MEDIAITEM_H

#include <QString>
#include <QDate>
#include <QStringList>
#include <memory>

// Forward declarations per Visitor pattern
class UIVisitor;

// ============================================================================
// CLASSE BASE ASTRATTA
// ============================================================================

class MediaItem {
protected:
    unsigned int id;
    QString title;
    QString description;
    QDate releaseDate;

public:
    MediaItem(unsigned int id, const QString& title, const QString&
description = "",
              const QDate& releaseDate = QDate());
    virtual ~MediaItem() = default;

    // ===== POLIMORFISMO NON BANALE =====

    // 1. Generazione UI specifica per tipo (Visitor pattern)
    virtual void accept(UIVisitor& visitor) = 0;

    // 2. Validazione specifica per tipo di media
    virtual bool isValid() const = 0;
    virtual QStringList getValidationErrors() const = 0;

    // 3. Serializzazione specifica per tipo
    virtual QString getTypeIdentifier() const = 0;  // Solo per persistenza,
NON per controllo flusso

    // ===== INTERFACCIA COMUNE =====
```

```
    unsigned int getId() const { return id; }
    QString getTitle() const { return title; }
    void setTitle(const QString& t) { title = t; }
    QString getDescription() const { return description; }
    void setDescription(const QString& d) { description = d; }
    QDate getReleaseDate() const { return releaseDate; }
    void setReleaseDate(const QDate& date) { releaseDate = date; }
};

// ============================================================
// BOOK - LIBRI
// ============================================================

class Book : public MediaItem {
private:
    QString author;
    QString isbn;
    QString publisher;
    int pageCount;
    QString genre;

public:
    Book(unsigned int id, const QString& title, const QString& author,
         const QString& description = "", const QDate& releaseDate =
QDate());

    // Polimorfismo non banale
    void accept(UIVisitor& visitor) override;
    bool isValid() const override;
    QStringList getValidationErrors() const override;
    QString getTypeIdentifier() const override { return "book"; }

    // Attributi specifici
    QString getAuthor() const { return author; }
    void setAuthor(const QString& a) { author = a; }
    QString getIsbn() const { return isbn; }
    void setIsbn(const QString& i) { isbn = i; }
    QString getPublisher() const { return publisher; }
    void setPublisher(const QString& p) { publisher = p; }
    int getPageCount() const { return pageCount; }
    void setPageCount(int p) { pageCount = p; }
    QString getGenre() const { return genre; }
    void setGenre(const QString& g) { genre = g; }
};

// ============================================================
// COMIC - FUMETTI
// ============================================================
```

```
class Comic : public MediaItem {
private:
    QString author;
    QString illustrator;
    QString series;
    int issueNumber;
    QString publisher;

public:
    Comic(unsigned int id, const QString& title, const QString& author,
          const QString& description = "", const QDate& releaseDate =
QDate());

    // Polimorfismo non banale
    void accept(UIVisitor& visitor) override;
    bool isValid() const override;
    QStringList getValidationErrors() const override;
    QString getTypeIdentifier() const override { return "comic"; }

    // Attributi specifici
    QString getAuthor() const { return author; }
    void setAuthor(const QString& a) { author = a; }
    QString getIllustrator() const { return illustrator; }
    void setIllustrator(const QString& i) { illustrator = i; }
    QString getSeries() const { return series; }
    void setSeries(const QString& s) { series = s; }
    int getIssueNumber() const { return issueNumber; }
    void setIssueNumber(int n) { issueNumber = n; }
    QString getPublisher() const { return publisher; }
    void setPublisher(const QString& p) { publisher = p; }
};

// ============================================================
// MOVIE - FILM
// ============================================================

class Movie : public MediaItem {
private:
    QString director;
    QString genre;
    int durationMinutes;
    QString rating;
    QStringList cast;

public:
    Movie(unsigned int id, const QString& title, const QString& director,
          const QString& description = "", const QDate& releaseDate =
QDate());

    // Polimorfismo non banale
```

```
    void accept(UIVisitor& visitor) override;
    bool isValid() const override;
    QStringList getValidationErrors() const override;
    QString getTypeIdentifier() const override { return "movie"; }

    // Attributi specifici
    QString getDirector() const { return director; }
    void setDirector(const QString& d) { director = d; }
    QString getGenre() const { return genre; }
    void setGenre(const QString& g) { genre = g; }
    int getDurationMinutes() const { return durationMinutes; }
    void setDurationMinutes(int d) { durationMinutes = d; }
    QString getRating() const { return rating; }
    void setRating(const QString& r) { rating = r; }
    QStringList getCast() const { return cast; }
    void setCast(const QStringList& c) { cast = c; }
};

// ============================================================
// MAGAZINE - RIVISTE
// ============================================================

class Magazine : public MediaItem {
private:
    QString publisher;
    int issueNumber;
    QString category;
    QString frequency; // "monthly", "weekly", etc.
    QString editor;

public:
    Magazine(unsigned int id, const QString& title, const QString&
publisher,
             const QString& description = "", const QDate& releaseDate =
QDate());

    // Polimorfismo non banale
    void accept(UIVisitor& visitor) override;
    bool isValid() const override;
    QStringList getValidationErrors() const override;
    QString getTypeIdentifier() const override { return "magazine"; }

    // Attributi specifici
    QString getPublisher() const { return publisher; }
    void setPublisher(const QString& p) { publisher = p; }
    int getIssueNumber() const { return issueNumber; }
    void setIssueNumber(int n) { issueNumber = n; }
    QString getCategory() const { return category; }
    void setCategory(const QString& c) { category = c; }
    QString getFrequency() const { return frequency; }
```

```
    void setFrequency(const QString& f) { frequency = f; }
    QString getEditor() const { return editor; }
    void setEditor(const QString& e) { editor = e; }
};

// ============================================================
// TVSERIES - SERIE TV
// ============================================================

class TVSeries : public MediaItem {
private:
    QString creator;
    int seasons;
    int episodesTotal;
    QString status; // "Ongoing", "Completed", "Cancelled"
    QString network;
    QString genre;

public:
    TVSeries(unsigned int id, const QString& title, const QString& creator,
             const QString& description = "", const QDate& releaseDate =
QDate());

    // Polimorfismo non banale
    void accept(UIVisitor& visitor) override;
    bool isValid() const override;
    QStringList getValidationErrors() const override;
    QString getTypeIdentifier() const override { return "tvseries"; }

    // Attributi specifici
    QString getCreator() const { return creator; }
    void setCreator(const QString& c) { creator = c; }
    int getSeasons() const { return seasons; }
    void setSeasons(int s) { seasons = s; }
    int getEpisodesTotal() const { return episodesTotal; }
    void setEpisodesTotal(int e) { episodesTotal = e; }
    QString getStatus() const { return status; }
    void setStatus(const QString& s) { status = s; }
    QString getNetwork() const { return network; }
    void setNetwork(const QString& n) { network = n; }
    QString getGenre() const { return genre; }
    void setGenre(const QString& g) { genre = g; }
};

// ============================================================
// VISITOR PATTERN - POLIMORFISMO NON BANALE PER UI
// ============================================================

class UIVisitor {
public:
```

```
    virtual ~UIVisitor() = default;

    // Metodi virtuali puri per ogni tipo di media
    virtual void visit(Book& book) = 0;
    virtual void visit(Comic& comic) = 0;
    virtual void visit(Movie& movie) = 0;
    virtual void visit(Magazine& magazine) = 0;
    virtual void visit(TVSeries& series) = 0;
};

// ============================================================
// REPOSITORY PATTERN - GESTIONE DATI
// ============================================================

#include <QList>
#include <memory>

class MediaRepository {
private:
    QList<std::unique_ptr<MediaItem>> items;
    unsigned int nextId;

public:
    MediaRepository();
    ~MediaRepository() = default;

    // CRUD Operations
    void addItem(std::unique_ptr<MediaItem> item);
    bool removeItem(unsigned int id);
    MediaItem* findById(unsigned int id);
    QList<MediaItem*> getAllItems();

    // Ricerca e filtraggio
    QList<MediaItem*> getItemsByType(const QString& type);
    QList<MediaItem*> searchItems(const QString& query);
    QList<MediaItem*> searchInType(const QString& query, const QString&
type);

    // Utility
    void clear();
    unsigned int generateId();
    size_t count() const { return items.size(); }

    // Per validazione
    bool validateAllItems() const;
    QStringList getAllValidationErrors() const;
};

// ============================================================
// PERSISTENCE LAYER - JSON
```

```
// ============================================================

#include <QJsonDocument>
#include <QJsonObject>
#include <QJsonArray>

class JSONPersistence {
public:
    static bool save(const MediaRepository& repo, const QString& filePath);
    static bool load(MediaRepository& repo, const QString& filePath);

private:
    static QJsonObject itemToJson(const MediaItem& item);
    static std::unique_ptr<MediaItem> jsonToItem(const QJsonObject& json);

    // Serializzazione specifica per tipo
    static QJsonObject bookToJson(const Book& book);
    static QJsonObject comicToJson(const Comic& comic);
    static QJsonObject movieToJson(const Movie& movie);
    static QJsonObject magazineToJson(const Magazine& magazine);
    static QJsonObject tvSeriesToJson(const TVSeries& series);

    // Deserializzazione specifica per tipo
    static std::unique_ptr<Book> jsonToBook(const QJsonObject& json);
    static std::unique_ptr<Comic> jsonToComic(const QJsonObject& json);
    static std::unique_ptr<Movie> jsonToMovie(const QJsonObject& json);
    static std::unique_ptr<Magazine> jsonToMagazine(const QJsonObject&
json);
    static std::unique_ptr<TVSeries> jsonToTVSeries(const QJsonObject&
json);
};

#endif // MEDIAITEM_H
```