Papp Bálint - HND2AH

NHF - Filmtár - Pontosított Feladatspecifikáció és Osztályterv

1. Pontosított Feladatspecifikáció

A feladat egy filmnyilvántartó rendszer elkészítése, amely képes különböző típusú filmek adatainak kezelésére. A rendszer egy parancssoros alkalmazás, amely lehetővé teszi filmek hozzáadását, törlését, keresését és listázását, valamint az adatok mentését és betöltését. A megoldáshoz heterogén kollekciót kell használni.

Bemenet és kimenet

Felhasználói parancsok

A program a standard bemenetről fogadja a felhasználói parancsokat:

- add: Új film hozzáadása a megadott típussal (normal, family, documentary)
- delete: Film törlése azonosító alapján
- list: Az összes film listázása
- search: Filmek keresése a megadott feltétel alapján
- save: A filmadatok mentése megadott fájlba
- load: Filmadatok betöltése megadott fájlból
- exit: Kilépés a programból

Filmadatok bevitele

Az add parancs után a program bekéri a film adatait:

- Minden filmtípus esetén: cím, lejátszási idő (percben), kiadási év
- Családi film esetén: korhatár (év)
- Dokumentumfilm esetén: leírás (szöveg)

Keresési feltételek

A search parancs után a következő keresési feltételek adhatók meg:

- title(szöveg): Keresés cím alapján
- year(szám): Keresés kiadási év alapján
- type(osztály): Keresés filmtípus alapján
- A feltételek kombinálhatók & jellel: title=Star&year=1977

Kimenet formátuma

- A list és search parancsok esetén minden film adatai egy-egy sorban jelennek meg, tabulátorral elválasztva:
    - [azonosító] [típus] [cím] [lejátszási idő] [kiadási év] [egyéb specifikus adatok]
- Hibaüzenetek a standard hibakimenetre kerülnek egyértelmű üzenettel
- A parancsok végrehajtása után a program visszajelzést ad a művelet sikerességéről

Adatmodell

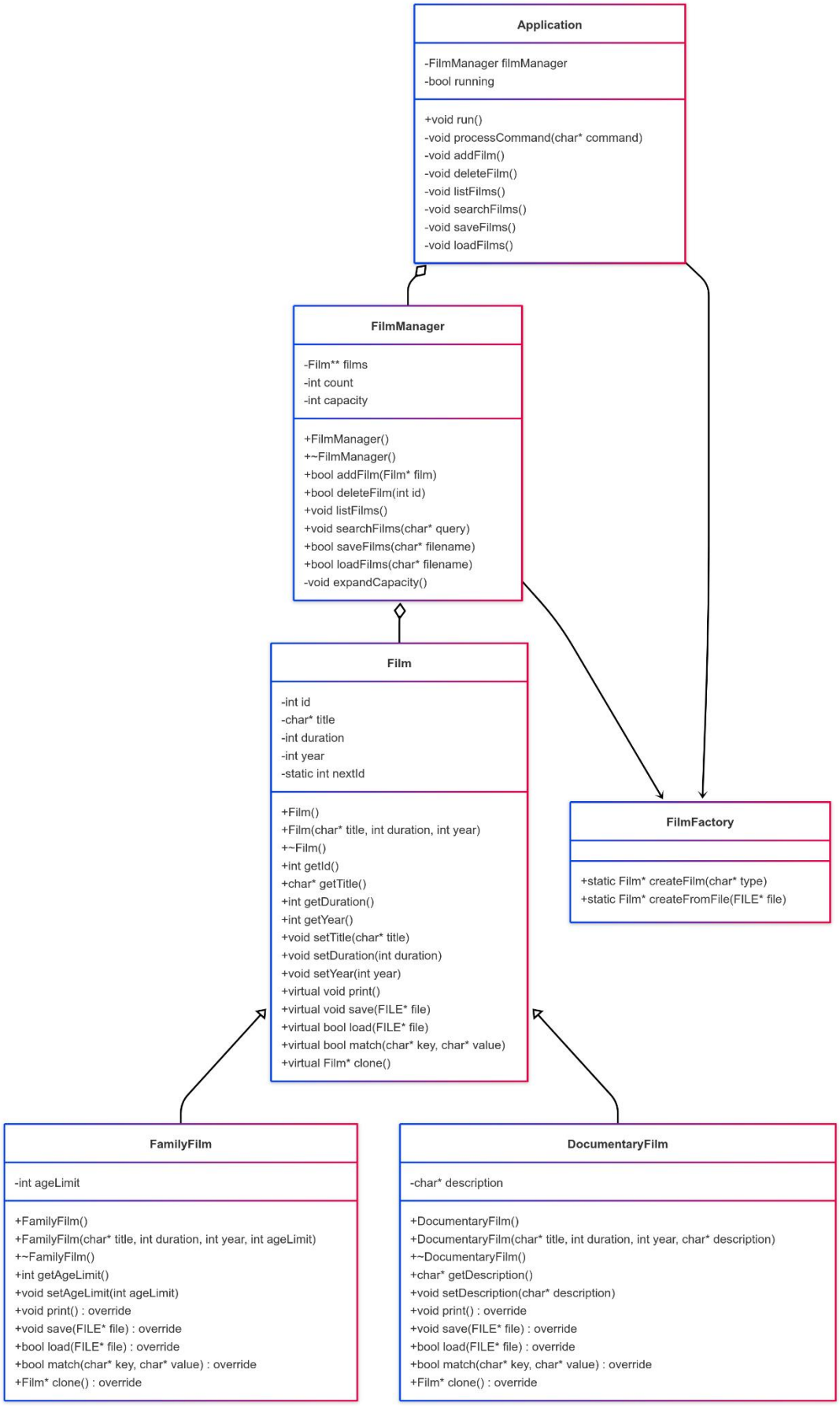
A program a következő adatokat tárolja:

1. **Film:**
    - Azonosító (egész szám)
    - Cím (szöveg)
    - Lejátszási idő (egész szám, percben)
    - Kiadási év (egész szám)
2. **Családi film** (Film leszármazottja):
    - Film adatai
    - Korhatár (egész szám)
3. **Dokumentumfilm** (Film leszármazottja):
    - Film adatai
    - Leírás (szöveg)

Fájlformátum

- Minden sor egy filmet reprezentál
- A sorok felépítése: [típus] [cím] [lejátszási idő] [kiadási év] [típus-specifikus adatok]
- Példa normál filmre: normal Star Wars 121 1977
- Példa családi filmre: family Frozen 102 2013 6
- Példa dokumentumfilmre: documentary Planet Earth 550 2006 Természeti dokumentumfilm

2. Osztálydiagram



3. Osztályok részletes leírása

Film (Absztrakt ősosztály)

Ez az osztály az összes filmtípus közös jellemzőit és viselkedését definiálja.

Attribútumok:

- id: Egyedi azonosító (egész szám)
- title: A film címe (karaktertömb)
- duration: Lejátszási idő percben (egész szám)
- year: Kiadási év (egész szám)
- static nextId: Következő kiosztandó azonosító

Metódusok:

- Konstruktorok és destruktor
- Getter és setter metódusok
- print(): Film adatainak kiírása a standard kimenetre
- save(FILE*): Film adatainak mentése fájlba
- load(FILE*): Film adatainak betöltése fájlból
- match(char* key, char* value): Ellenőrzi, hogy a film megfelel-e a keresési feltételnek
- clone(): Másolat készítése a filmről (virtuális metódus)
- createFromFile(FILE*): Statikus metódus film létrehozására fájlból

FamilyFilm (Film leszármazottja)

Családi filmek specifikus adatait és viselkedését definiálja.

Attribútumok:

- ageLimit: Korhatár (egész szám)

Metódusok:

- Az ősosztály metódusainak felüldefiniálása
- Az ageLimit attribútum getter és setter metódusai

DocumentaryFilm (Film leszármazottja)

Dokumentumfilmek specifikus adatait és viselkedését definiálja.

Attribútumok:

- description: Leírás (karaktertömb)

Metódusok:

- Az ősosztály metódusainak felüldefiniálása
- A description attribútum getter és setter metódusai

FilmManager

Ez az osztály felelős a filmek tárolásáért és a rajtuk végezhető műveletekért. Heterogén kollekciót használ a különböző típusú filmek tárolására.

Attribútumok:

- films: Film pointerek tömbje (Film**)
- count: A tárolt filmek száma
- capacity: A tömb aktuális kapacitása

Metódusok:

- addFilm(Film*): Új film hozzáadása
- addFilm(const char* type, const char* title, int duration, int year, const char* extraData): Film létrehozása és hozzáadása
- deleteFilm(int id): Film törlése azonosító alapján
- listFilms(): Összes film listázása
- searchFilms(char* query): Filmek keresése feltétel alapján
- saveFilms(char* filename): Filmek mentése fájlba
- loadFilms(char* filename): Filmek betöltése fájlból
- expandCapacity(): A tömb kapacitásának növelése (privát metódus)
- createFilm(const char* type, const char* title, int duration, int year, const char* extraData): Új film létrehozása típus alapján (privát metódus)

Application

Ez az osztály felelős a felhasználói interakcióért és a program futásáért.

Attribútumok:

- filmManager: A filmek kezelését végző objektum
- running: Futás jelző (logikai érték)

Metódusok:

- run(): A program fő ciklusa

- processCommand(char*): Parancs feldolgozása
- Különböző parancsokat végrehajtó metódusok

4. Fontosabb algoritmusok

4.1. Dinamikus memóriakezelés (FilmManager::expandCapacity)

```
expandCapacity() {
  // 1. Új, nagyobb kapacitású tömb
  létrehozása újKapacitás = régiKapacitás * 2
  újTömb = új Film*[újKapacitás]

  // 2. Adatok átmásolása
  for i = 0 to count-1 do
    újTömb[i] = régiTömb[i]

  // 3. Régi tömb felszabadítása
  delete[] régiTömb

  // 4. Változók frissítése
  films = újTömb
  capacity = újKapacitás
}
```

4.2. Filmek keresése (FilmManager::searchFilms)

```
searchFilms(query) {
  // 1. Keresési feltételek feldolgozása
  feltételek[] = query.split('&')

  // 2. Összes film ellenőrzése a feltételek
  alapján for i = 0 to count-1 do
    találat = true

    // 3. Minden feltétel ellenőrzése
    for minden feltétel in feltételek
    do kulcs, érték = feltétel.split('=')

      // 4. Ha nem felel meg a feltételnek, nem
      találat if !films[i]->match(kulcs, érték) then
        találat =
        false break

    // 5. Ha minden feltételnek megfelel, kiírjuk
    if találat then
      films[i]->print()
```

```
}
```

4.3. Filmek mentése fájlba (FilmManager::saveFilms)

```
saveFilms(filename) {
  // 1. Fájl megnyitása írásra
  file = fopen(filename, "w") if
  file == NULL then
    return false

  // 2. Filmek mentése egyenként
  for i = 0 to count-1 do
    films[i]->save(file)

  // 3. Fájl bezárása
  fclose(file)
  return true
}
```

4.4. Filmek betöltése fájlból (FilmManager::loadFilms)

```
loadFilms(filename) {
  // 1. Fájl megnyitása olvasásra
  file = fopen(filename, "r")
  if file == NULL
    then return false

  // 2. Jelenlegi filmek törlése
  for i = 0 to count-1 do
    delete films[i]
  count = 0

  // 3. Új filmek betöltése
  while !feof(file) do
    film = FilmFactory::createFromFile(file)
    if film != NULL then
      addFilm(film)

  // 4. Fájl bezárása
  fclose(file) return
  true
}
```

4.5. Parancs feldolgozása (Application::processCommand)

```
processCommand(command) {
  // 1. Parancs részekre
  bontása parts[] =
  command.split(' ')

  // 2. Megfelelő művelet
  végrehajtása if parts[0] ==
  "add" then
    addFilm()
          else if parts[0] ==
               "delete" then
    deleteFilm(parseInt(pa
                       rts[1]))
  else if parts[0] ==
    "list" then
    listFilms()
  else if parts[0] ==
    "search" then
    searchFilms(parts[1])
  else if parts[0] ==
    "save" then if
    parts.length > 1
    then
      saveFilms(p
    arts[1]) else
      saveFilms("filmek.
  dat") else if parts[0]
  == "load" then
    if parts.length
      > 1 then
      loadFilms(pa
      rts[1])
    else
      loadFilms("film
      ek.dat")
  else if parts[0] ==
    "exit" then running
    = false
  else
    print("Ismeretlen parancs")
}
```

5. Doxygen
(csak bemásoltam az rtf-ből amit generált)

NHF

AUTHOR

Version

Table of contents

Hierarchical Index

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

File Index

File List

Here is a list of all documented files with brief descriptions:

Class Documentation

Application Class Reference

Class for handling user interaction.

`#include <application.h>`

Public Member Functions

- **Application** ()
  Constructor.

- **~Application** ()
  Destructor.

- void **run** ()
  Run the application.

Detailed Description

Class for handling user interaction.

Member Function Documentation

void Application::run ()

Run the application.
Main loop that handles user input and program execution.

The documentation for this class was generated from the following files:

application.happlication.cpp

DocumentaryFilm Class Reference

Class representing documentary films.
`#include <documentaryFilm.h>`

Inheritance diagram for DocumentaryFilm:

IMAGE

Public Member Functions

- **DocumentaryFilm** ()
  Default constructor.

- **DocumentaryFilm** (const char *title, int duration, int year, const char *description)
  Parameterized constructor.

- **DocumentaryFilm** (const **DocumentaryFilm** &other)
  Copy constructor.

- **~DocumentaryFilm** ()
  Destructor.

- const char * **getDescription** () const
  Get the documentary description.

- void **setDescription** (const char *description)
  Set the documentary description.

- void **print** () const override
  Print documentary film details to standard output Overrides the base class method to include description.

- bool **save** (FILE *file) const override
  Save documentary film details to a file.

- bool **load** (FILE *file) override
  Load documentary film details from a file.

- bool **match** (const char *key, const char *value) const override
  Check if documentary film matches a search criteria.

- **Film** * **clone** () const override
  Create a deep copy of this documentary film.

Public Member Functions inherited from Film

- **Film** ()

Default constructor.

- **Film** (const char *title, int duration, int year)
  Parameterized constructor.

- **Film** (const **Film** &other)
  Copy constructor.

- virtual ~**Film** ()
  Virtual destructor.

- int **getId** () const
  Get the film ID.

- const char * **getTitle** () const
  Get the film title.

- int **getDuration** () const
  Get the film duration.

- int **getYear** () const
  Get the film year.

- void **setTitle** (const char *title)
  Set the film title.

- void **setDuration** (int duration)
  Set the film duration.

- void **setYear** (int year)
  Set the film year.

Additional Inherited Members

Static Public Member Functions inherited from Film

- static **Film** * **createFromFile** (FILE *file)
  Create a film from file data.

Detailed Description

Class representing documentary films.

Constructor & Destructor Documentation

*DocumentaryFilm::DocumentaryFilm (const char * title, int duration, int year, const char * description)*

Parameterized constructor.

Parameters

title	The film title
duration	The film duration in minutes
year	The release year
description	The documentary description

DocumentaryFilm::DocumentaryFilm (const DocumentaryFilm & other)

Copy constructor.

Parameters

other	The documentary film to copy

Member Function Documentation

*Film * DocumentaryFilm::clone () const* `[override], [virtual]`

Create a deep copy of this documentary film.

Returns

Pointer to a new documentary film object

Reimplemented from **Film** (*p.17*).

*const char * DocumentaryFilm::getDescription () const*

Get the documentary description.

Returns

The description text

*bool DocumentaryFilm::load (FILE * file)* `[override], [virtual]`

Load documentary film details from a file.

Parameters

file	Pointer to an open file

Returns

true if successful, false otherwise

Reimplemented from **Film** (*p.18*).

*bool DocumentaryFilm::match (const char * key, const char * value) const* `[override], [virtual]`

Check if documentary film matches a search criteria.

Parameters

key	The search key
value	The search value

Returns

true if matches, false otherwise

Reimplemented from **Film** (*p.18*).

void DocumentaryFilm::print () const `[override], [virtual]`

Print documentary film details to standard output Overrides the base class method to include description.

Reimplemented from **Film** (*p.18*).

*bool DocumentaryFilm::save (FILE * file) const* `[override], [virtual]`

Save documentary film details to a file.

Parameters

file	Pointer to an open file

Returns

true if successful, false otherwise

Reimplemented from **Film** (*p.18*).

*void DocumentaryFilm::setDescription (const char * description)*

Set the documentary description.

Parameters

description	The new description

The documentation for this class was generated from the following files:

documentaryFilm.hdocumentaryFilm.cpp

FamilyFilm Class Reference

Class representing family films.
`#include <familyFilm.h>`

Inheritance diagram for FamilyFilm:

IMAGE

Public Member Functions

- **FamilyFilm** ()
  Default constructor.

- **FamilyFilm** (const char *title, int duration, int year, int ageLimit)
  Parameterized constructor.

- **FamilyFilm** (const **FamilyFilm** &other)
  Copy constructor.

- **~FamilyFilm** ()
  Destructor.

- int **getAgeLimit** () const
  Get the age limit.

- void **setAgeLimit** (int ageLimit)
  Set the age limit.

- void **print** () const override
  Print family film details to standard output Overrides the base class method to include age limit.

- bool **save** (FILE *file) const override
  Save family film details to a file.

- bool **load** (FILE *file) override
  Load family film details from a file.

- bool **match** (const char *key, const char *value) const override
  Check if family film matches a search criteria.

- **Film** * **clone** () const override
  Create a deep copy of this family film.

Public Member Functions inherited from Film

- **Film** ()

Default constructor.

- **Film** (const char *title, int duration, int year)
  Parameterized constructor.

- **Film** (const **Film** &other)
  Copy constructor.

- virtual ~**Film** ()
  Virtual destructor.

- int **getId** () const
  Get the film ID.

- const char * **getTitle** () const
  Get the film title.

- int **getDuration** () const
  Get the film duration.

- int **getYear** () const
  Get the film year.

- void **setTitle** (const char *title)
  Set the film title.

- void **setDuration** (int duration)
  Set the film duration.

- void **setYear** (int year)
  Set the film year.

Additional Inherited Members

Static Public Member Functions inherited from Film

- static **Film** * **createFromFile** (FILE *file)
  Create a film from file data.

Detailed Description

Class representing family films.

Constructor & Destructor Documentation

*FamilyFilm::FamilyFilm (const char * title, int duration, int year, int ageLimit)*

Parameterized constructor.

Parameters

title	The film title
duration	The film duration in minutes
year	The release year
ageLimit	The minimum age required to watch the film

FamilyFilm::FamilyFilm (const FamilyFilm & other)

Copy constructor.

Parameters

other	The family film to copy

Member Function Documentation

*Film * FamilyFilm::clone () const* `[override], [virtual]`

Create a deep copy of this family film.

Returns

Pointer to a new family film object

Reimplemented from **Film** (*p.17*).

int FamilyFilm::getAgeLimit () const

Get the age limit.

Returns

The age limit in years

*bool FamilyFilm::load (FILE * file)* `[override], [virtual]`

Load family film details from a file.

Parameters

file	Pointer to an open file

Returns

true if successful, false otherwise

Reimplemented from **Film** (*p.18*).

*bool FamilyFilm::match (const char * key, const char * value) const* `[override], [virtual]`

Check if family film matches a search criteria.

Parameters

key	The search key
value	The search value

Returns

true if matches, false otherwise

Reimplemented from **Film** (*p.18*).

void FamilyFilm::print () const `[override], [virtual]`

Print family film details to standard output Overrides the base class method to include age limit.

Reimplemented from **Film** (*p.18*).

*bool FamilyFilm::save (FILE * file) const* `[override], [virtual]`

Save family film details to a file.

Parameters

file	Pointer to an open file

Returns

true if successful, false otherwise

Reimplemented from **Film** (*p.18*).

void FamilyFilm::setAgeLimit (int ageLimit)

Set the age limit.

Parameters

ageLimit	The new age limit

The documentation for this class was generated from the following files:

familyFilm.hfamilyFilm.cpp

Film Class Reference

Base class for all film types.
`#include <film.h>`

Inheritance diagram for Film:

IMAGE

Public Member Functions

- **Film** ()
  Default constructor.
- **Film** (const char *title, int duration, int year)
  Parameterized constructor.
- **Film** (const **Film** &other)
  Copy constructor.
- virtual ~**Film** ()
  Virtual destructor.
- int **getId** () const
  Get the film ID.
- const char * **getTitle** () const
  Get the film title.
- int **getDuration** () const
  Get the film duration.
- int **getYear** () const
  Get the film year.
- void **setTitle** (const char *title)
  Set the film title.
- void **setDuration** (int duration)
  Set the film duration.
- void **setYear** (int year)
  Set the film year.
- virtual void **print** () const
  Print film details to standard output.

- virtual bool **save** (FILE *file) const
  Save film details to a file.

- virtual bool **load** (FILE *file)
  Load film details from a file.

- virtual bool **match** (const char *key, const char *value) const
  Check if film matches a search criteria.

- virtual **Film** * **clone** () const
  Create a deep copy of this film.

Static Public Member Functions

- static **Film** * **createFromFile** (FILE *file)
  Create a film from file data.

Detailed Description

Base class for all film types.
Film is an abstract class that contains common film attributes and methods.

Constructor & Destructor Documentation

*Film::Film (const char * title, int duration, int year)*

Parameterized constructor.

Parameters

title	The film title
duration	The film duration in minutes
year	The release year

Film::Film (const Film & other)

Copy constructor.

Parameters

other	The film to copy

Member Function Documentation

*Film * Film::clone () const* `[virtual]`

Create a deep copy of this film.

Returns

Pointer to a new film object

Reimplemented in **DocumentaryFilm** (*p.7*), and **FamilyFilm** (*p.12*).

*Film * Film::createFromFile (FILE * file)* `[static]`

Create a film from file data.

Parameters

file	Open file pointer to read from

Returns

Pointer to a new film object or nullptr if failed

int Film::getDuration () const

Get the film duration.

Returns

The film duration in minutes

int Film::getId () const

Get the film ID.

Returns

The film ID

*const char * Film::getTitle () const*

Get the film title.

Returns

The film title

int Film::getYear () const

Get the film year.

Returns

The release year

*bool Film::load (FILE * file) [virtual]*

Load film details from a file.

Parameters

file	Pointer to an open file

Returns

true if successful, false otherwise

Reimplemented in **DocumentaryFilm** (*p.8*), and **FamilyFilm** (*p.12*).

*bool Film::match (const char * key, const char * value) const [virtual]*

Check if film matches a search criteria.

Parameters

key	The search key (e.g., "title", "year")
value	The search value

Returns

true if matches, false otherwise

Reimplemented in **DocumentaryFilm** (*p.8*), and **FamilyFilm** (*p.13*).

void Film::print () const [virtual]

Print film details to standard output.

Reimplemented in **DocumentaryFilm** (*p.8*), and **FamilyFilm** (*p.13*).

*bool Film::save (FILE * file) const [virtual]*

Save film details to a file.

Parameters

file	Pointer to an open file

Returns

true if successful, false otherwise

Reimplemented in **DocumentaryFilm** (*p.8*), and **FamilyFilm** (*p.13*).

void Film::setDuration (int duration)

Set the film duration.

Parameters

duration	The new film duration

*void Film::setTitle (const char * title)*

Set the film title.

Parameters

title	The new film title

void Film::setYear (int year)

Set the film year.

Parameters

year	The new release year

The documentation for this class was generated from the following files:

film.hfilm.cpp

FilmManager Class Reference

Class for managing a collection of films.
`#include <filmManager.h>`

Public Member Functions

- **FilmManager** (int initialCapacity=10)
  Constructor.

- **~FilmManager** ()
  Destructor.

- bool **addFilm** (**Film** *film)
  Add a film to the collection.

- bool **addFilm** (const char *type, const char *title, int duration, int year, const char *extraData=nullptr)
  Create and add a film to the collection.

- bool **deleteFilm** (int id)
  Delete a film from the collection.

- void **listFilms** () const
  List all films in the collection.

- void **searchFilms** (const char *query) const
  Search films based on query string.

- bool **saveFilms** (const char *filename) const
  Save all films to a file.

- bool **loadFilms** (const char *filename)
  Load films from a file.

- int **getCount** () const
  Get the number of films in the collection.

Detailed Description

Class for managing a collection of films.
Provides functionality for adding, removing, searching, saving, and loading films.

Constructor & Destructor Documentation

FilmManager::FilmManager (int initialCapacity = `10`)

Constructor.

Parameters

initialCapacity	Initial capacity of the films array

Member Function Documentation

*bool FilmManager::addFilm (const char * type, const char * title, int duration, int year, const char * extraData = `nullptr`)*

Create and add a film to the collection.

Parameters

type	The type of film ("normal", "family", "documentary")
title	The film title
duration	The film duration in minutes
year	The release year
extraData	Additional data specific to the film type

Returns

true if successful, false otherwise

*bool FilmManager::addFilm (Film * film)*

Add a film to the collection.

Parameters

film	Pointer to the film to add

Returns

true if successful, false otherwise

bool FilmManager::deleteFilm (int id)

Delete a film from the collection.

Parameters

id	The ID of the film to delete

Returns

true if successful, false otherwise

int FilmManager::getCount () const

Get the number of films in the collection.

Returns

The number of films

*bool FilmManager::loadFilms (const char * filename)*

Load films from a file.

Parameters

filename	The name of the file to load from

Returns

true if successful, false otherwise

*bool FilmManager::saveFilms (const char * filename) const*

Save all films to a file.

Parameters

filename	The name of the file to save to

Returns

true if successful, false otherwise

*void FilmManager::searchFilms (const char * query) const*

Search films based on query string.

Parameters

query	The search query (format: "key1=value1&key2=value2...")

The documentation for this class was generated from the following files:

- **filmManager.hfilmManager.cpp**

File Documentation

application.cpp File Reference

Implementation of the **Application** class.

```
#include "application.h"
#include <cstdlib>
#include <cstring>
#include <iostream>
```

Detailed Description

Implementation of the **Application** class.

application.h File Reference

Application class declaration.

```
#include "filmmanager.h"
```

Classes

- class **Application***Class for handling user interaction.*

Detailed Description

Application class declaration.

application.h

Go to the documentation of this file.

```
1
5
6 #ifndef APPLICATION_H
7 #define APPLICATION_H
8
9 #include "filmmanager.h"
10
15 class Application {
16 private:
17      FilmManager filmManager;
18      bool running;
19
24      void processCommand(const char *command);
25
30      void addFilm();
31
36      void deleteFilm();
37
41      void listFilms() const;
42
47      void searchFilms() const;
48
53      void saveFilms() const;
54
59      void loadFilms();
60
61 public:
65      Application();
66
70      ~Application();
71
77      void run();
78 };
79
80 #endif // APPLICATION_H
```

documentaryFilm.cpp File Reference

Implementation of the **DocumentaryFilm** class.

```
#include "documentaryfilm.h"
#include <cstring>
```

Detailed Description

Implementation of the **DocumentaryFilm** class.

documentaryFilm.h File Reference

DocumentaryFilm class declaration.

```
#include "film.h"
```

Classes

- class **DocumentaryFilm***Class representing documentary films.*

Detailed Description

DocumentaryFilm class declaration.

documentaryFilm.h

Go to the documentation of this file.

```
1
5
6 #ifndef DOCUMENTARYFILM_H
7 #define DOCUMENTARYFILM_H
8
9 #include "film.h"
10
15 class DocumentaryFilm : public Film {
16 private:
17       char *description;
18
19 public:
23       DocumentaryFilm();
24
32       DocumentaryFilm(const char *title, int duration, int year, const char
*description);
33
38       DocumentaryFilm(const DocumentaryFilm &other);
39
43       ~DocumentaryFilm();
44
49       const char *getDescription() const;
50
55       void setDescription(const char *description);
56
61       void print() const override;
62
68       bool save(FILE *file) const override;
69
75       bool load(FILE *file) override;
76
83       bool match(const char *key, const char *value) const override;
84
89       Film *clone() const override;
90 };
91
92 #endif // DOCUMENTARYFILM_H
```

familyFilm.cpp File Reference

Implementation of the **FamilyFilm** class.

```
#include "familyfilm.h"
#include <cstring>
```

Detailed Description

Implementation of the **FamilyFilm** class.

familyFilm.h File Reference

FamilyFilm class declaration.
`#include "film.h"`

Classes

- class **FamilyFilm***Class representing family films.*

Detailed Description

FamilyFilm class declaration.

familyFilm.h

Go to the documentation of this file.

```
1
5
6 #ifndef FAMILYFILM_H
7 #define FAMILYFILM_H
8
9 #include "film.h"
10
15 class FamilyFilm : public Film {
16 private:
17      int ageLimit;
18
19 public:
23      FamilyFilm();
24
32      FamilyFilm(const char *title, int duration, int year, int ageLimit);
33
38      FamilyFilm(const FamilyFilm &other);
39
43      ~FamilyFilm();
44
49      int getAgeLimit() const;
50
55      void setAgeLimit(int ageLimit);
56
61      void print() const override;
62
68      bool save(FILE *file) const override;
69
75      bool load(FILE *file) override;
76
83      bool match(const char *key, const char *value) const override;
84
89      Film *clone() const override;
90 };
91
92 #endif // FAMILYFILM_H
```

film.cpp File Reference

Implementation of the **Film** class.

```
#include "film.h"
#include "documentaryfilm.h"
#include "familyfilm.h"
#include <cstdlib>
#include <cstring>
```

Detailed Description

Implementation of the **Film** class.

film.h File Reference

Film class declaration.

```
#include <cstdio>
```

Classes

- class **Film***Base class for all film types.*

Detailed Description

Film class declaration.

film.h

Go to the documentation of this file.

```
1
5
6 #ifndef FILM_H
7 #define FILM_H
8
9 #include <cstdio>
10
17 class Film {
18 private:
19      int id;
20      char *title;
21      int duration;
22      int year;
23      static int nextId;
24
25 public:
29      Film();
30
37      Film(const char *title, int duration, int year);
38
43      Film(const Film &other);
44
48      virtual ~Film();
49
54      int getId() const;
55
60      const char *getTitle() const;
61
66      int getDuration() const;
67
72      int getYear() const;
73
78      void setTitle(const char *title);
79
84      void setDuration(int duration);
85
90      void setYear(int year);
91
95      virtual void print() const;
96
102      virtual bool save(FILE *file) const;
103
109      virtual bool load(FILE *file);
110
117      virtual bool match(const char *key, const char *value) const;
118
123      virtual Film *clone() const;
124
130      static Film *createFromFile(FILE *file);
131 };
132
133 #endif // FILM_H
```

filmManager.cpp File Reference

Implementation of the **FilmManager** class.

```
#include "filmmanager.h"
#include "documentaryfilm.h"
#include "familyfilm.h"
#include <cstdlib>
#include <cstring>
```

Detailed Description

Implementation of the **FilmManager** class.

filmManager.h File Reference

FilmManager class declaration.
`#include "film.h"`

Classes

- class **FilmManager***Class for managing a collection of films.*

Detailed Description

FilmManager class declaration.

filmManager.h

Go to the documentation of this file.

```
1
5
6 #ifndef FILMMANAGER_H
7 #define FILMMANAGER_H
8
9 #include "film.h"
10
18 class FilmManager {
19 private:
20       Film **films;
21       int count;
22       int capacity;
23
29       void expandCapacity();
30
40       Film *createFilm(const char *type, const char *title, int duration, int year,
const char *extraData);
41
42 public:
47       FilmManager(int initialCapacity = 10);
48
52       ~FilmManager();
53
59       bool addFilm(Film *film);
60
70       bool addFilm(const char *type, const char *title, int duration, int year,
const char *extraData = nullptr);
71
77       bool deleteFilm(int id);
78
82       void listFilms() const;
83
88       void searchFilms(const char *query) const;
89
95       bool saveFilms(const char *filename) const;
96
102       bool loadFilms(const char *filename);
103
108       int getCount() const;
109 };
110
111 #endif // FILMMANAGER_H
```

main.cpp File Reference

Main entry point for the app.

```
#include "application.h"
#include <iostream>
```

Functions

- int **main** ()
  Main function.

Detailed Description

Main entry point for the app.

Function Documentation

int main ()

Main function.

Returns

Exit code

Index