HÁZI FELADAT

Programozás alapjai 2.

Kész

Tóth Mihály Balázs
OAYOF1

2024. 05. 19.

TARTALOM

1. Feladat

Készítsen filmeket nyilvántartó rendszert. Minden filmnek tároljuk a címét, lejátszási idejét és kiadási évét. A családi filmek esetében korhatár is van, a dokumentumfilmek esetében egy szöveges leírást is tárolunk. Tervezzen könnyen bővíthető objektummodellt a feladathoz! Demonstrálja a működést külön modulként fordított tesztprogrammal! A megoldáshoz ne használjon STL tárolót!

2. Feladatspecifikáció

A feladat egy filmeket tároló nyilvántartás létrehozása. Egy filmnek több attribútuma lehet (cím, játékidő, kiadás éve) a különböző típusú filmek egyedi attribútumai is eltárolhatók, például családi filmeknél a korhatár, dokumentumfilmeknél egy rövid leírás. A nyilvántartáshoz könnyen hozzá lehet adni új filmeket és új osztályok létrehozásával új filmtípusokat is.
Alapvető funkciója a nyilvántartásnak a kereshetőség, film címe alapján. Ha nem található az adott cím akkor hibát dob a rendszer. A katalógushoz hozzá lehet adni és ki lehet belőle törölni filmeket. Továbbá lehetőségünk van a teljes adatbázis kiírására, illetve beolvasására is .txt fájlból.
A filmek adatai a tetszőlegesen hosszú címet leszámítva fix méretű tárolókban vannak nyilvántartva, míg maguk a filmek dinamikusan foglalt területen, így könnyen bővíthető az új filmek hozzáadásával.
A tesztprogram egy előre elkészített adatbázist tölt be, ezen demonstrálja a keresést, a hozzáadást és a törlést.

3. Terv

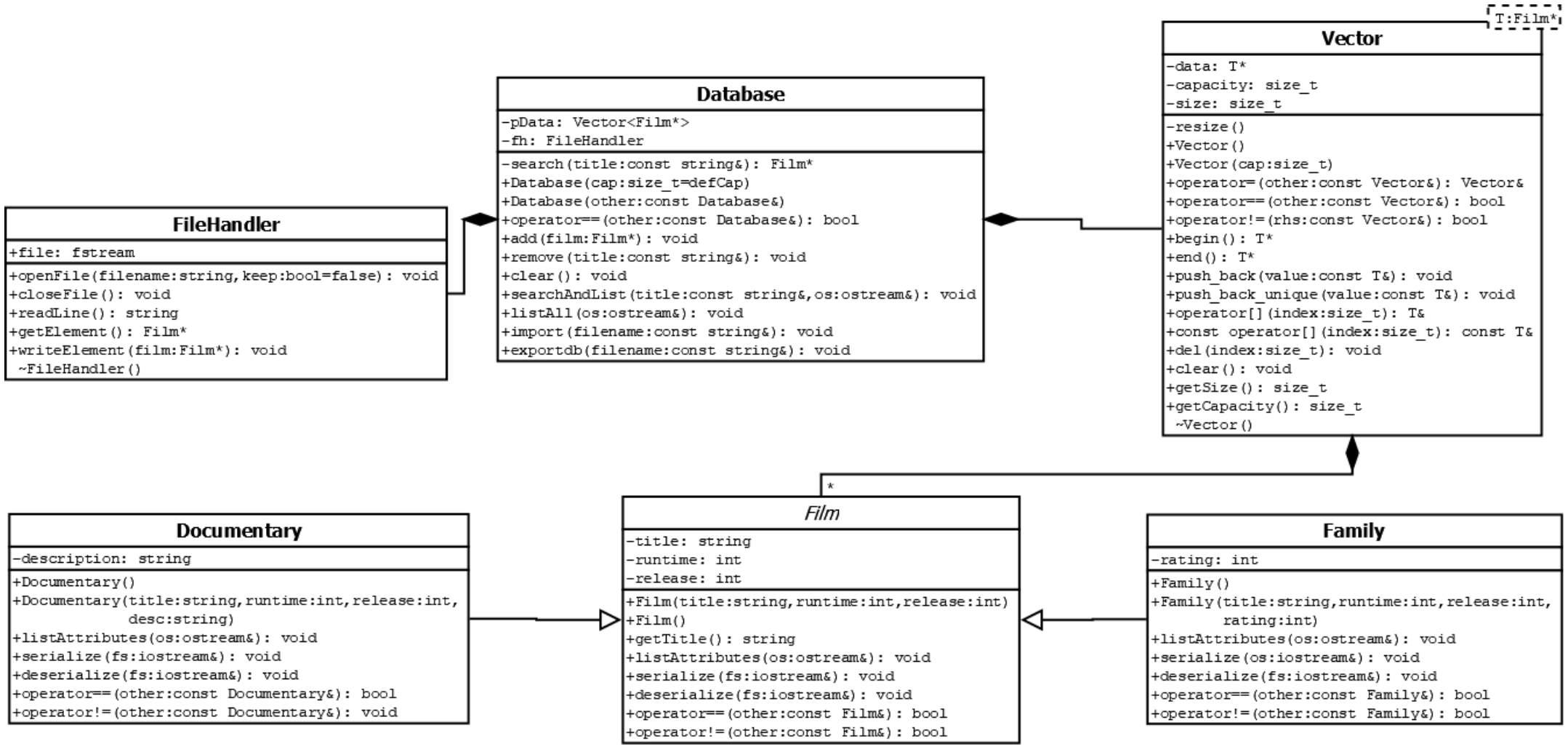
A feladat az alábbi objektumok és a tesztprogram megtervezését igényli.
Ez a terv alapján készült el a végleges megoldás, a korábbi beadásoknál látott tervektől eltér.

3.1. Objektum terv

A megvalósításhoz az UML diagramon látható osztályokat hozom létre:

- Datbase: Heterogén kollekció, egy Vector-ban tárolja a Film pointerket.
- Film: Absztrakt ősosztály, ebből példányosodik a Family és Documentary. Tárolja a címet, a játékidőt és a megjelenési évet.

- Family: Családi film alosztály, a Film-hez képest tárolja a korhatárt és felüldefiniálja a szükséges függvényeket
- Documentary: Dokumentumfilm alosztály, a Film-hez képest tárolja a leírást és felüldefiniálja a szükséges függvényeket
- FileHandler: Az importálás/exportáláshoz kezeli a fileokat
- Vector: Generikus tároló osztály


T:Film*
Vector
-data: T*
-capacity: size_t
-size: size_t
-resize()
+Vector()
+Vector(cap:size_t)
+operator=(other:const Vector&): Vector&
+operator==(other:const Vector&): bool
+operator!=(rhs:const Vector&): bool
+begin(): T*
+end(): T*
+push_back(value:const T&): void
+push_back_unique(value:const T&): void
+operator[](index:size_t): T&
+const operator[](index:size_t): const T&
+del(index:size_t): void
+clear(): void
+getSize(): size_t
+getCapacity(): size_t
~Vector()
Database
-pData: Vector<Film*>
-fh: FileHandler
-search(title:const string&): Film*
+Database(cap:size_t=defCap)
+Database(other:const Database&)
+operator==(other:const Database&): bool
+add(film:Film*): void
+remove(title:const string&): void
+clear(): void
+searchAndList(title:const string&,os:ostream&): void
+listAll(os:ostream&): void
+import(filename:const string&): void
+exportdb(filename:const string&): void
FileHandler
+file: fstream
+openFile(filename:string,keep:bool=false): void
+closeFile(): void
+readLine(): string
+getElement(): Film*
+writeElement(film:Film*): void
~FileHandler()
*
Film
-title: string
-runtime: int
-release: int
+Film(title:string,runtime:int,release:int)
+Film()
+getTitle(): string
+listAttributes(os:ostream&): void
+serialize(fs:iostream&): void
+deserialize(fs:iostream&): void
+operator==(other:const Film&): bool
+operator!=(other:const Film&): bool
Documentary
-description: string
+Documentary()
+Documentary(title:string,runtime:int,release:int,
desc:string)
+listAttributes(os:ostream&): void
+serialize(fs:iostream&): void
+deserialize(fs:iostream&): void
+operator==(other:const Documentary&): bool
+operator!=(other:const Documentary&): void
Family
-rating: int
+Family()
+Family(title:string,runtime:int,release:int,
rating:int)
+listAttributes(os:ostream&): void
+serialize(os:iostream&): void
+deserialize(fs:iostream&): void
+operator==(other:const Family&): bool
+operator!=(other:const Family&): bool

3.2. Algoritmusok

Alább a megoldáshoz szükséges algoritmusok közül csak a bonyolultabbakat említem.

3.2.1. Database :: add

Ezzel a függvénnyel lehet hozzáadni a tárolóhoz új filmet, a beillesztendő pointer használatával. Ellenőrzésre kerül, hogy van-e elég szabad hely a tárolóban (ha nincs, megnöveli) majd hozzáfűzi a tömbhöz. Ellenőrzi a címet is, duplikált adatsorok felvételének elkerülésére. A Vector push_back_unique függvényét használja.

3.2.1.1 Vector::resize

A tárolónak foglalt terület megnövelésére szolgál. A költséges memóriafoglalási és másolási műveletek számának minimalizálása érdekében minden hívásra foglal egy dupla akkora méretű tömböt és ebbe másolja át az elemeket, az eredetit felszabadítva.

3.2.2. Database :: remove

Megkeresi az adott címet és eltávolítja a tárolóból. A helyén keletkező lyuk feltöltéséhez az aktuális utolsó tömbelemet áthelyezi ide.
Ha nincs ilyen című elem, akkor *char const** kivételt dob „entry not found” értékkel.

3.2.3. Database :: searchAndList

A kapott cím alapján megkeresi majd listázza az elem attribútumait a paraméterül kapott streamre, az adott *listAttributes* függvényének segítségével
Ha nincs találat, a fent említett kivételt dobja.

3.2.4. Database :: import

A paraméterül kapott fájlból beolvassa a rekordokat és felépíti a tárolót. Egy film adatai egy sorban vannak tárolva tabulátorokkal elválasztva. A sorokat beolvassa, és az ennek megfelelően létrejövő objektumokat elhelyezi az adatbázisban.
Olvasási hiba esetén char const* kivételt dob „file not found” értékkel.
A funkció a FileHandler osztályon keresztül valósul meg.

3.2.5. Database :: exportdb

A megadott nevű fájlba exportálja az adatbázist, olyan formában, hogy az *import* függvény azt később kezelni tudja.
Megvalósításához a *listAll* függvény kimenetét irányítja át.
Megnyítási hiba esetén szintén *char const** kivételt dob „file not found” értékkel.
A funkció a FileHandler osztályon keresztül valósul meg.

3.2.6. Film::deserialize

A megadott fstreamről beolvassa az objektum adatait. A leszármazottai is használják az ősosztály deserialize függvényét.

3.2.7. Tesztprogram algoritmusai

A tesztprogram több unit testből áll, ezek a testing makróval választhatók ki.

- Az első egy Databese objektumon teszteli a keresést: hozzáad néhány filmet, majd az egyiket megkeresi és kilistázza.
- A második az exportálást ellenőrzi: felépít egy kis adatbázist, exportálja egy fájlba, majd a fájl tartalmát soronként ellenőrzi.
- A harmadik az importálást vizsgálja: felépít egy kis adatbázist és exportálja egy fájlba. Ezután létrehoz egy másolatot az első adatbázisról, törli az első adatait majd visszaolvasás után ellenőrzi, megyezik-e a tartalmuk.
- A negyedik a törlést ellenőrzi: felépít egy kis adatbázist, törli egy elemét, majd ellenőrzi, hogy a törölt elem (és csak az) hiányzik.
- Az utolsó test case a hibakezeléseket ellenőrzi. Előidézi a helyzeteket, ahol hibát dobhat a rendszer, majd ellenőrzi, hogy megfelelő hibákat dobott-e.

4. Megvalósítás

4.1. Osztályok és függvényeik

4.1.1. Database

A heterogén kollekciót kezelő osztály.

4.1.1.1 Változók

Név	Típus	Funkció
pData	Vector<Film*>	Film pointerek tárolója
fh	FileHandler	Fájlkezelő objektum

4.1.1.2 Függvények

Database::Database(size_t cap)
Adatbázis konstruktor, létrehozza az adatbázist a megadott kezdeti kapacitással.
Alapértelmezett kapacitás megadása a defCap makróval.

void Database::add(Film *film)
A függvény hozzáadja a pointerrel paraméterül kapott Film objektumot az adatbázishoz.

Film *Database::search(std::string title)
Ez a függvény végigiterál az adatbázis tárolóján, és megkeresi a paraméterként kapott címmel rendelkező filmet.
Ha megtalálja, visszaadja annak a pointer-ét, különben kivételt dob az "entry not found" üzenettel.

void Database::remove(std::string title)
Film eltávolítása az adatbázisból
A függvény eltávolítja az adatbázisból a paraméterként kapott címmel rendelkező filmet.
Ha nem találja meg a filmet akkor kivételt dob az "entry not found" üzenettel.

void Database::searchAndList(std::string title, std::ostream &os)
Film keresése és adatainak listázása
Ez a függvény megkeresi a paraméterként kapott címmel rendelkező filmet az adatbázisban, majd kiírja annak adatait a megadott ostream-re.

void Database::listAll(std::ostream &os)
Az összes film adatainak listázása
Ez a függvény kiírja az adatbázisban tárolt összes film adatait a megadott ostream-re.

void Database::import(std::string filename)
Adatbázis importálása fájlból
Ez a függvény betölti az adatbázisba a megadott fájlban tárolt filmek adatait.
A fájlban a filmek típusa után a megfelelő adatok szerepelnek, a függvény a film típusának megfelelően tölti be az értékeket.
Ha a fájl megnyitása sikertelen a „file not found” errort dobja.
A funkció a FileHandler osztályon keresztül valósul meg.

void Database::exportdb(std::string filename)
Adatbázis exportálása fájlba
Ez a függvény kiírja az adatbázisban tárolt összes film adatait a megadott fájlba.
Ha a fájl megnyitása sikertelen a „file not found” errort dobja.
A funkció a FileHandler osztályon keresztül valósul meg.

4.1.2. Vector<T>

Generikus tároló, a programban Film*- okat tárol.

4.1.2.1 Változók

Név	Típus	Funkció
data	T*	A tároló pointere
capacity	size_t	A lefoglalt méret
size	size_T	A feltöltött elemek száma

4.1.2.2 Függvények

Vector<T>::Vector()
Létrehoz egy üres 1 méretű Vector objektumot.

Vector<T>::Vector(size_t cap)
Létrehoz egy üres Vector objektumot megadott kapacitással.

Vector<T>::~Vector()
Felszabadítja a Vector által használt memóriát.

Vector<T>& Vector<T>::operator=(const Vector& other)
Másik Vector objektum értékét másolja.

bool Vector<T>::operator==(const Vector& other)
Összehasonlít két Vector objektumot egyenlőség szempontjából.

bool Vector<T>::operator!=(const Vector<T>& rhs)
Összehasonlít két Vector objektumot egyenlőtlenség szempontjából. Az előző függvényt negálja.

T* Vector<T>::begin()
Visszaadja a Vector elejére mutató iterátort.

T* Vector<T>::end()
Visszaadja a Vector végére mutató iterátort.

void Vector<T>::push_back(const T &value)
Hozzáad egy elemet a Vector végéhez.

void Vector<T>::push_back_unique(const T &value)
Hozzáad egy egyedi elemet a Vector végéhez, ha az még nem szerepel benne.

T& Vector<T>::operator[](size_t index)
Visszaadja az indexhez tartozó elemet. Van konstans változata is.

void Vector<T>::del(size_t index)
Törli az indexnél található elemet.

void Vector<T>::clear()
Törli az összes elemet a Vectorból.

size_t Vector<T>::getCapacity()
Visszaadja a Vector aktuális kapacitását.

size_t Vector<T>::getSize()
Visszaadja a Vector aktuális méretét.

void Vector<T>::resize()
Növeli a Vector kapacitását (duplázza).

4.1.3. Film

Absztrakt ősosztálya a különféle filmtípusoknak.

4.1.3.1 Változók

Név	Típus	Funkció
title	std::string	A film címe
runtime	int	Játékidő (perc)
release	int	Kiadási év

4.1.3.2 Függvények

Film::Film(std::string title, int runtime, int release)

Film osztály konstruktora
Létrehozza a Film objektumot a megadott címmel, játékidővel és kiadási évvel.

Film::Film()
Létrehoz egy Film objektumot alapértékekkel. (üres string, illetve 0)

std::string Film::getTitle()
Visszaadja a film címét.

void Film::listAttributes(std::ostream &os)
Kiírja a film attribútumait a megadott ostream-re.

void Film::serialize(std::iostream fs)
A desrialize függvény által elvárt formátumban írja ki az adatait a paraméterül kapott streamre.

void Film::desrialize(std::iostram fs)
Beolvassa az adatait a streamről.

bool Film::operator==
Összehasonlítható vele két Film objektum.

bool Film::operator!=
Összehasonlítható vele két Film objektum, az előző függvényt negálja.

Film::~Film()
Virtuális destruktor. Nincs az osztályban adattag ami igényelné, de a leszármazottakban lehetne.

4.1.4. Family

A Film osztályból származtatott családi film osztály.

4.1.4.1 Változók

Örökli a Film osztály adattagjait, illetve

Név	Típus	Funkció
rating	int	Korhatár

4.1.4.2 Függvények

Family::Family(std::string title, int runtime, int release, int rating)
Létrehozza a Family objektumot a megadott címmel, játékidővel, kiadási évvel és korhatárral. Ehhez a Film osztály konstruktorát veszi igénybe.

Family::Family()
Alapértelmezett konstruktor, amely alapértékekkel inicializálja a családi film attribútumait. (Üres string, illetve 0).

void Family::listAttributes(std::ostream &os)
Kiírja az objektum attribútumait a megadott ostream-re.

A Film osztály azonos nevű függvényének override-ja.

void Family::serialize(std::iostream& fs)
A családi film adatait a megadott fájlba írja a desrialize függvény által elvárt formátumban.

void Family::deserialize(std::iostream& fs)
A családi film adatait olvassa be a megadott fájlból.

bool Family::operator==(const Family& other)
Összehasonlít két családi filmet egyenlőség szempontjából.

bool Family::operator!=(const Family& other)
Összehasonlít két családi filmet egyenlőtlenség szempontjából.

4.1.5. Documentary

A Film osztályból származtatott dokumentumfilm osztály.

4.1.5.1 Változók

Örökli a Film osztály adattagjait, illetve

Név	Típus	Funkció
description	std::string	Rövid leírás a filmhez

4.1.5.2 Függvények

Documentary::Documentary(std:: string title, int runtime, int release, std:: string desc)
Létrehozza a Documentary objektumot a megadott címmel, játékidővel, kiadási évvel és leírással.
Ehhez a Film osztály konstruktorát használja fel.

Documentary::Documentary()
Alapértékekkel hozza létre az objektumot. (Üres string és 0)

void Documentary::listAttributes(std::ostream &os)
Kiírja az objektum attribútumait a megadott ostream-re.
A Film osztály azonos nevű függvényének override-ja.

void Documentary::serialize(std::iostream& fs) override;
A dokumentumfilm adatait fájlba írja.

void Documentary::deserialize(std::iostream& fs) override;
A dokumentumfilm adatait fájlból olvassa be.

bool Documentary::operator==(const Documentary& other) const;
Összehasonlít két dokumentumfilmet egyenlőség szempontjából.

bool Documentary::operator!=(const Documentary& other)

Összehasonlít két dokumentumfilmet egyenlőtlenség szempontjából.

4.1.6. FileHandler

A FileHandler osztály felelős a fájlkezelési műveletekért a projekten belül.

4.1.6.1 Változók

Név	Típus	Funkció
file	std::fstream	A cél file beolvasáshoz és/vagy kiíráshoz

4.1.6.2 Függvények

void FileHandler::openFile(const std::string& filename, bool keep = fasle)
Ez a metódus megnyit egy fájlt a megadott fájlnév alapján. Ha a keep hamis (alapértelmezetten az), törli a tartalmát további műveletek előtt. Ha igaz akkor megtartja. Ha a fájl nem található, kivételt dob.

std::string FileHandler::readLine()
Ez a metódus beolvas egy sort a megnyitott fájlból. Ha nincs több sor a fájlban, üres stringet ad vissza.

Film* FileHandler::getElement()
A fájlból kiolvassa a következő sort. Ebből pedig létrehozza a típusnak megfelelő Film objektumot (Family/Documentary), ezzel tér vissza.
Ha egyik típusnak sem felel meg a bemenet, „invalid input” hibát dob.

void FileHandler::writeElement(Film *film)
A paraméterül kapott Film objektumot kiírja a beolvasáshoz alkalmas formátumban.

void FileHandler::closeFile()
Ez a metódus bezárja a megnyitott fájlt, ha az nyitva van.

FileHandler::~FileHandler()
A destruktor, bezárja a fájlt, ha az nyitva van.

4.2. Tesztprogram

A program több unit testet tartalmaz az implementált osztályok és funkciók tesztelésére. A tesztesetek kiválasztásához a laborfeladatokhoz hasonlóan egy a fájl elején lévő makrót használ.
Az eredeti tervhez képest lényegi változás, hogy az egyes funkciók a hibafeltárás megkönnyítésére először egymástól izolálva vannak tesztelve, illetve nem használ csatolt adatbázist, hanem magának exportál egyet az import testhez.

A következő tesztek az adatbázis alapvető működését ellenőrzik.

- DatabaseSearchTest: Ez a teszteset az adatbázis keresési funkcióját vizsgálja. Hozzáad néhány Family és Documentary objektumot az adatbázishoz, majd keres egy adott című filmet. Ellenőrzi, hogy a keresés eredménye megfelel-e az elvárt eredménynek.

- DatabaseExportTest: Ez a teszteset az adatbázis exportálási funkcióját teszteli. Létrehoz egy adatbázist, hozzáad néhány filmet, majd exportálja az adatokat egy fájlba. Ellenőrzi, hogy az exportált fájl tartalma megfelel-e az elvárt eredménynek.

- DatabaseImportTest: Ez a teszteset az adatbázis importálási funkcióját ellenőrzi. Importál egy exportált tesztfájlt az adatbázisba, majd ellenőrzi, hogy az importált adatok megfelelően lettek-e feldolgozva és betöltve.

- DatabaseRemoveTest: Ez a teszteset az adatbázisból törlés funkcióját teszteli. Hozzáad néhány Family és Documentary objektumot az adatbázishoz, majd töröl egy adott filmet. Ellenőrzi, hogy a törlés után az adatbázis tartalma megfelel-e az elvárt eredménynek.

Az utolsó tesztek a hibakezelést ellenőrzik:

- DatabaseSearchErrorTest: Ez a teszteset a keresés hibakezelését vizsgálja. Létrehoz egy minimális adatbázist, majd megpróbál megkeresni benne egy nem létező filmet. A dobott hibát ellenőrzi, hogy megfelelő-e.

- DatabaseRemoveErrorTest: Ez a teszteset a törlés hibakezelését vizsgálja. Létrehoz egy minimális adatbázist, majd megpróbál törölni belőle egy nem létező filmet. A dobott hibát ellenőrzi, hogy megfelelő-e.

- DatabaseImportErrorTest: Ez a teszteset az importálás hibakezelését vizsgálja. Megpróbál egy nem létező fájlt beolvasni, majd a dobott hibát ellenőrzi, hogy megfelelő-e.