МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тверской государственный технический университет»

(ТвГТУ)

Кафедра “Программного обеспечения”

**Лабораторная работа**

по дисциплине “Конструирование программных систем”

Тема: « Разработка UML модели проекта “Онлайн-кинотеатр” »

Выполнил: студент группы

ПИН 17.06

Завгороднев Е. Ю

Проверил:

Мальков А.А

Тверь 2020

# Введение

Цель работы: получить навык создания на основе UML модели в системе Visual Studio 2013 и изучить особенности создания на основе UML кода на языке C# .

# Ответы на контрольные вопросы

1. Какие этапы жизненного цикла разработки программного проекта поддерживает среда Visual Studio 2013.

* проектирование
* кодирование (программирование)
* тестирование и отладка

1. Какие UML-диаграммы можно построить с использованием Visual Studio 2013.

* Диаграмма классов
* Последовательности
* Вариантов использования
* Деятельности
* Компонентов

1. Каковы возможности автоматической генерации кода в Visual Studio 2013.

* Выбор шаблона для генерации кода (класс, перечисление, интерфейс, структура)
* Выбор директории для создаваемых классов

1. Рефакторинг в Visual Studio 2013. Его возможности и ограничения.

Возможности:

* Переименование
* Создание метода для выделенного кода
* Изменение порядка аргументов функции
* Извлечение интерфейса из определенного класса
* Инкапсулирование поля

# Диаграммы

## Последовательности



## Использования



## Активности



## Классов











## Пример доработанного сгенерированого кода

package filmsproject.datalayer.jpa;

import filmsproject.interfaces.FilmAccessService;

import filmsproject.model.Film;

import filmsproject.model.Genre;

import filmsproject.model.Review;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Persistence;

import javax.persistence.TypedQuery;

import java.time.LocalDate;

import java.util.List;

@Component

public class FilmAccessJPA implements FilmAccessService {

private EntityManager manager;

@Autowired

private ReviewRepository reviewRepository;

public FilmAccessJPA() {

EntityManagerFactory factory = Persistence.createEntityManagerFactory("movies");

manager = factory.createEntityManager();

}

@Override

public Film getFilmByIdentifier(String identifier) {

return manager.find(Film.class, identifier);

}

@Override

public List<Film> getFilmsByTitle(String title) {

TypedQuery<Film> q = manager.createQuery(

"Select c from Film c Where c.title = ?1", Film.class);

q.setParameter(1,title);

return q.getResultList();

}

@Override

public List<Film> getFilmsByReleaseDate(LocalDate date) {

TypedQuery<Film> q = manager.createQuery(

"Select c from Film c Where c.releaseDate = ?1", Film.class);

q.setParameter(1,date);

return q.getResultList();

}

@Override

public List<Film> getFilmsInRange(Double fromRating, Double toRating) {

TypedQuery<Film> typedQuery = manager.createQuery("Select m from Film m WHERE m.rating BETWEEN ?1 AND ?2",Film.class);

typedQuery.setParameter(1,fromRating);

typedQuery.setParameter(2,toRating);

return typedQuery.getResultList();

}

@Override

public List<Film> getFilmsInRange(LocalDate fromYear, LocalDate toYear) {

TypedQuery<Film> typedQuery = manager.createQuery("Select m from Film m WHERE m.releaseDate BETWEEN ?1 AND ?2",Film.class);

typedQuery.setParameter(1,fromYear);

typedQuery.setParameter(2,toYear);

return typedQuery.getResultList();

}

@Override

public List<Genre> getAllGenres() {

TypedQuery<Genre> q = manager.createQuery(

"Select g from Genre g", Genre.class);

return q.getResultList();

}

@Override

public List<Review> getFilmReviews(String filmIdentifier) {

TypedQuery<Review> q = manager.createQuery(

"Select c from Review c Where c.filmIdentifier = ?1", Review.class);

q.setParameter(1,filmIdentifier);

return q.getResultList();

}

@Override

public boolean addNewReview(String filmIdentifier, Review review) {

review.setFilmIdentifier(filmIdentifier);

manager.getTransaction().begin();

manager.persist(review);

manager.getTransaction().commit();

return manager.find(Review.class,review.getReviewId()).getReviewId() != null;

}

@Override

public boolean deleteReview(Long reviewId) {

Review review = manager.find(Review.class, reviewId);

if (review.getReviewId() != null) {

manager.getTransaction().begin();

manager.remove(review);

manager.getTransaction().commit();

return true;

}

return false;

}

@Override

public boolean updateReview (Long reviewId, LocalDate date, String reviewText, double rating){

Review review = manager.find(Review.class, reviewId);

if (review.getReviewId() != null) {

manager.getTransaction().begin();

review.setCreateDate(date);

review.setReviewText(reviewText);

review.setRating(rating);

manager.getTransaction().commit();

return true;

}

return false;

}

}