Техническое задание на разработку приложения "Movie Business Management Platform"

Описание проекта

Создать приложение для управления процессами в киноиндустрии, которое будет включать функциональность для:

- Управления съемками фильмов.
- Организации премьер.
- Управления контрактами с актерами и режиссерами.
- Учет финансов (бюджет фильмов, доходы, расходы).

Приложение должно быть разработано на языке Java с использованием принципов ООП. Предполагается дальнейшее расширение на веб- и мобильные платформы.

Требования к функционалу

1. Управление съемками фильмов:

- Создание съемочных проектов с указанием названия, жанра, сроков съемок, бюджета, ответственного продюсера и статуса проекта. При создании проекта предусмотреть валидацию данных (например, проверка на корректность дат и положительное значение бюджета).
- Добавление и удаление актеров и режиссеров из проекта. Реализовать в виде методов, которые обновляют список участников проекта. Должна быть предусмотрена проверка на уникальность имени участника и тип роли (актер или режиссер).
- Управление статусами проекта ("планируется", "в производстве", "завершен") с возможностью автоматического обновления статуса в зависимости от текущей

даты и состояния проекта. Реализовать через перечисления (`enum`) для статусов.
- Генерация отчета о текущем состоянии всех проектов в текстовый файл, включая информацию о названии, жанре, текущем статусе, дате окончания съемок и списке участников. Реализовать обработку ошибок при записи в файл.

2. Организация премьер:

- Создание события премьеры с указанием фильма, даты, места, бюджета, количества билетов и приглашенных гостей. Предусмотреть автоматическую проверку доступности бюджета и дат для проведения премьеры.
- Учет проданных билетов с возможностью возврата билетов. Реализовать обработку исключений, если количество проданных билетов превышает доступное количество.
- Список приглашенных гостей (актеры, режиссеры, спонсоры).
- Генерация отчета о мероприятии с указанием проданных билетов, общей прибыли, списком гостей и отзывами участников. Реализовать возможность отправки отчета по электронной почте (с эмуляцией отправки).

3. Управление контрактами:

- Хранение данных о контрактах актеров и режиссеров (имя, роль, срок контракта, гонорар, привязка к конкретному проекту). Контракты должны быть связаны с конкретным проектом через идентификаторы.
- Отслеживание сроков действия контрактов.
- Уведомления о предстоящем окончании контрактов. Реализовать с помощью потоков (`Thread`) или планировщика (`ScheduledExecutorService`), чтобы проверка происходила автоматически раз в сутки.

4. Учет финансов:

- Управление бюджетом фильмов (распределение на съемки, гонорары, рекламу и прочее) с возможностью визуализации текущих затрат и остатка бюджета. Реализовать расчет бюджета через классы, представляющие категории расходов.
- Учет доходов (сборы с проката, спонсорские средства).
- Генерация отчетов о прибыли и убытках по каждому фильму с детализированным разбором по категориям (доходы от проката, спонсоры, расходы на съемки и рекламу). Отчет должен сохраняться в формате CSV и PDF.

Архитектура приложения

Основные модули:

1. **Core:**
   - Классы `MovieManager`, `PremiereManager`, `ContractManager`, `FinanceManager` для управления соответствующими процессами.
2. **Model:**
   - Классы данных: `Movie`, `Premiere`, `Contract`, `FinanceRecord`.
3. **Utilities:**
   - Утилиты для обработки дат, генерации отчетов.
4. **Database:**
   - Использование коллекций `HashMap` и `ArrayList` для хранения данных в памяти на начальном этапе.

Основные классы

1. `Movie`

- Поля:
  - `id` (уникальный идентификатор).
  - `title` (название).
  - `genre` (жанр).
  - `startDate`, `endDate` (сроки съемок).
  - `status` (статус проекта).
- Методы:
  - `addActor(String actorName)`
  - `removeActor(String actorName)`
  - `updateStatus(String status)`

2. `Premiere`

- Поля:
  - `id` (уникальный идентификатор).
  - `movieTitle` (название фильма).
  - `date` (дата премьеры).

    - `location` (место проведения).
    - `ticketCount` (количество билетов).
    - `guestList` (список гостей).
- Методы:
    - `sellTicket(int count)`
    - `addGuest(String guestName)`

3. `Contract`

- Поля:
    - `id` (уникальный идентификатор).
    - `personName` (имя).
    - `role` (роль: актер/режиссер).
    - `startDate`, `endDate` (срок действия).
    - `salary` (гонорар).
- Методы:
    - `isActive()`
    - `daysUntilExpiration()`

4. `FinanceRecord`

- Поля:
    - `id` (уникальный идентификатор).
    - `type` (тип записи: доход/расход).
    - `amount` (сумма).
    - `description` (описание).
    - `date` (дата).

Требования к реализации

1. Использовать принципы ООП (инкапсуляция, наследование, полиморфизм).
2. Обработку исключений для некорректных данных (например, отрицательный бюджет, неверный статус).
3. Покрытие функционала базовыми юнит-тестами (JUnit).
4. Документировать код с использованием JavaDoc.

План реализации

Этапы разработки:

1. Разработка базовых классов данных и модулей.
2. Реализация бизнес-логики.
3. Тестирование и отладка.
4. Генерация отчетов и логирование.

Критерии оценки

1. Полнота выполнения требований.
2. Корректность работы ключевых функций.
3. Качество кода (читаемость, использование ООП, обработка исключений).
4. Успешное прохождение тестов.

Дополнительно

При успешной реализации возможно расширение:

- Добавление графического интерфейса на основе JavaFX.