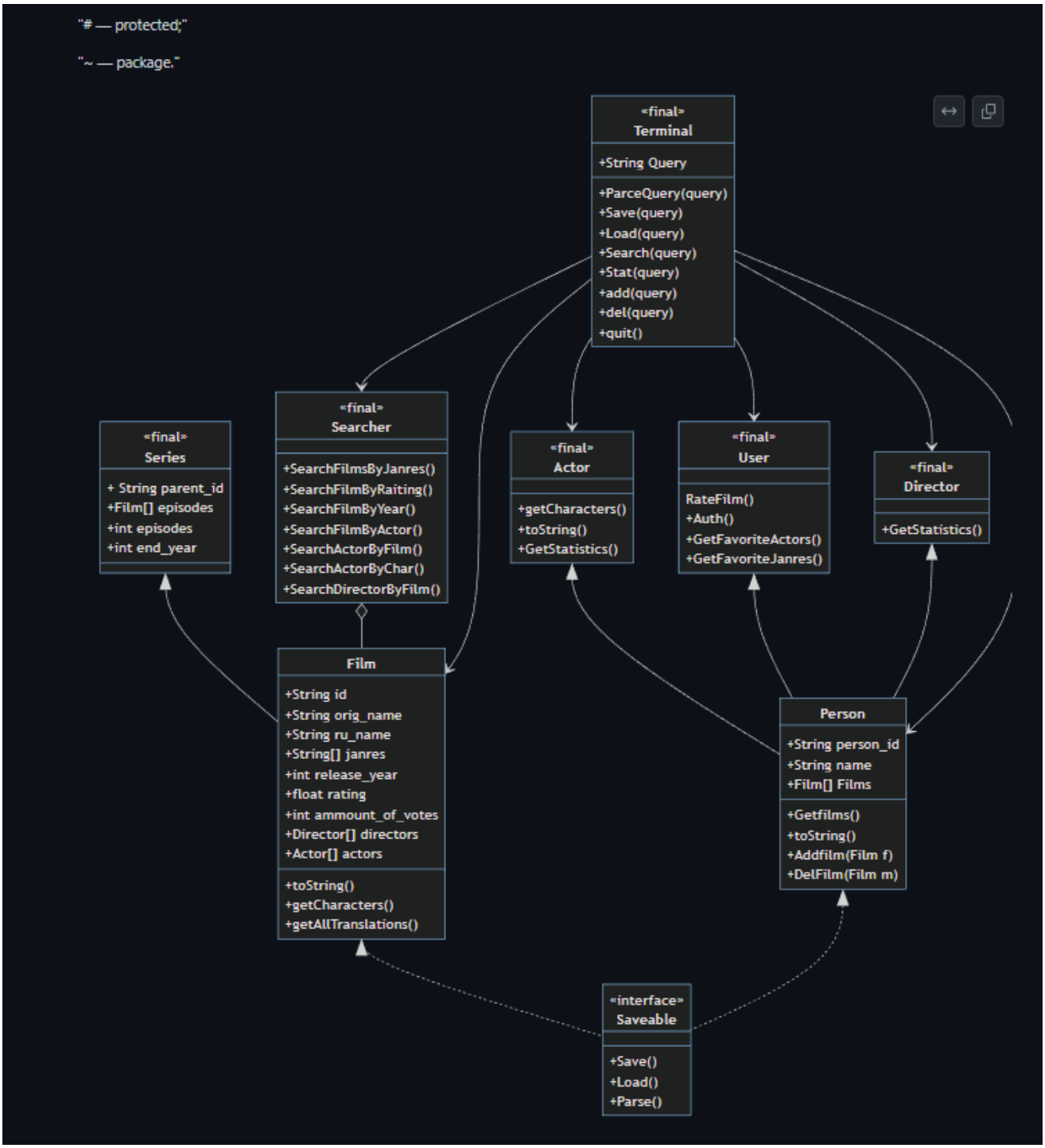

Аннотация

Класс **Terminal** является управляющим классом всего проекта, его методы направлены на вызов методов остальных классов, работу с внутренними данными и реализацию работы с пользователем посредством обработки консольно вводимых команд.

- Метод ***ParceQuery*** занимается первоначальной обработкой вводимых пользователем данных, отделяя запрашиваемые действия (**команды**) от **аргументов** и **ключей** к ним, а затем передает управление, вызывая требуемый метод. Также метод используется внутри любой другой функции класса **Terminal**, т. к. возвращает переданные аргументы списком вида:

```
public String[] ParceQuery(String input)
["Ключ", "Значение",...]
```

- Метод ***Save(String query)*** интерфейса **Saveable** содержит три перегруженных варианта. Класс **Terminal** вызывает нужный, основывясь на запросе, переданным в строке *query*.
  Операция вызывается в случае, если строка *query* содержит в себе команду **Save**.
  Первый вариант метода не принимает аргументы:

```
public boolean Save() // Первый вариант перегрузки
```

В этом случае программа сохраняет все возможные данные классов, реализующих интерфейс **Saveable** в заранее определенные пути, из ~~конфигурационного файла?~~
В противном случае, метод Save принимает в себя классы, реализующие интерфейс **Saveable**, и путь к файлу. Если аргумент пути не указан, то используется стандартный путь из ~~Файла конфигурации?~~

```
public boolean Save(Saveable data, Path path) // Перегрузка с указанием пути
public boolean Save(Saveable data) // Перегрузка с сохранением по стандартному
пути
```

Метод возвращает **True**, если сохранение прошло успешно.

- Метод **Load()** интерфейса **Saveable** реализован тремя перегрузками с аналогичным методу **Save()** применением, но с обратным эффектом. Вызывается в случае, если *query* содержит команду **Load**.

  ```
  public boolean Load()
  public boolean Load(Saveable data, Path path)
  public boolean Load(Saveable data)
  ```

- Метод **Search()** в качестве единственного **обязательного** параметра принимает строку-запрос для поиска.

  ```
  private void Search(String SearchFor)
  ```

  Он создает экземпляр класса **Searcher**, который реализует дальнейшую логику поиска по объектам и возвращение результата в основной код (или выводу его на экран).

Строкой поиска считается строка, которая идет после команды **Search** в строке *query*.

- Метод ***Stat()*** выводит статистику запрашиваемого человека, метод принимает в себя объект класса **Person**, для которого в дальнейшем вызывается метод **GetStatistics()** конкретного класса, дочернего классу **Person**, исключая класс **User**.

Работа метода основана на том, что все методы класса **Searcher** в качестве результата возвращают списки:

```
Person[] или Film[]
```

Прототип метода **Stat**():

```
private void Stat(Person p)
```

Вызов осуществляется при помощи команды **stat**, аргументом принимается индекс фильма/человека.

- Применение методов **add()** и **del()** является очевидным, они запускают мастер создания нового объекта любого типа или процесс удаления существующего соответственно:

```
private void del()

private Series add()
private Film add()
private Actor add()
private User add()
private Director add()
```

Вызов осуществляется при помощи команд **del** и **add** без аргументов.

- Метод **quit()** используется для штатного завершения работы программы:

```
private void quit()
```

Вызов: **q**

Класс **Searcher** является поисковиком, который поволяет искать фильмы или людей по нескольким параметрам:
для фильмов:

1. жанр
2. рейтинг
3. год
4. актеры

для людей:

1. фильм
2. персонаж

Таким образом, прототипы функций класса **Searcher**:

```
public Film[] SearchFilmsByJanres(String query)
public Film[]SearchFilmByRaiting(String query)
public Film[]SearchFilmByYear(String query)
public Film[]SearchFilmByActor(String query)
public Person[]SearchActorByFilm(String query)
public Person[]SearchActorByChar(String query)
public Person[]SearchDirectorByFilm(String query)
```

- Для класса **Actor** специально определен метод **getCharacters()**, который возвращает
  список персонажей актера.

```
public String[] getCharacters()
public String[] getCharacters(Film film)
```

- Для класса **Actor** перегружен метод **toString()** с целью удобного вывода информации для пользователя в терминал, при помощи стандратных методов Java.

- Для класса **User** был определен метод **RateFilm()**, который предназначен для оценивания фильма пользователем по факту просмотра.

```
public void RateFilm(Film film)
```

- Метод **Auth()** используется для авторизации по логину и паролю, вводимому с терминала.

```
public boolean Auth()
```

- Методы **GetFavoriteActors()** и **GetFavoriteJanres()** нужны для определения любимых актеров и жанров пользователя, основываясь на списке просмотренных им фильмов.

```
public Actor[] GetFavoriteActors()
public String[] GetFavoriteJanres()
```

Пара классов **Film** и **Series** содержит множество разнообразных полей:

Для **Series**:

Поле	Значение

Поле	Значение
String parent_id	Id объекта класса фильм, который ассоциируется с сериалом
Film[] episodes	Список эпизодов сериала
int episodes	Кол-во эпизодов
int end_year	Год конца, -1, если нет

Для **Film**:

Поле	Значение
String id	Id объекта
String orig_name	Оригинальное название
String ru_name	Русское название, если есть
String[] janres	Массив жанров
int release_year	Год выхода
float rating	Рейтинг фильма
int ammount_of_votes	Кол-во голосов
Director[] directors	Массив режиссеров
Actor[] actors	Массив актеров

Класс **Series** наследует **Film**, благодаря этому методы, определенные для **Film**, актуальны и для него.

- Для **Film** определен метод **toString()**, который нужен для корректного вывода информации в терминал.

- Метод **getCharacters()** нужен для получения списка персонажей, учавствующих в фильме.

```
public String[] getCharacters()
```

- Так как в базах данных один и тот же фильм может быть представлен в нескольких вариантах, был создан метод **getAllTranslations()**, который возвращает все переводы фильма.

```
public String[] getAllTranslations()
```

Класс **Person** является родителем для **User**, **Actor**,**Director**, а значит, все поля и методы у этих классов общие.

Поле	Значение
String person_id	Id объекта
String name	Имя
Film[] Films	Список фильмов

Для класса **Person** определены методы:

- **Getfilms()**, который используется для получения списка фильмов у каждого актера, режиссера или пользователя.

```
public Film[] Getfilms()
```

- Применение метода **toString()** очевидно.

- Методы **Addfilm()** и **Delfilm()** нужны для добавления фильма в массив *Films* любого экземпляра **Person** или его детей.

```
public void Addfilm(Film f)
public void DelFilm(Film m)
```

- Метод **Parse()** интерфейса **Saveable** используется для того, чтобы разобрать файлы баз данных и создать экземпляры классов.

```
public void Parse(Path FilmsFile, Path ActorsFile)
public void Parse()
```

Второй вариант функции использует пути к файлам из ~~Файла конфигурации?~~