COMP 303 Winter 2021

Assignment 4

Belle Pan 260839939

19th March, 2021

`FilterCollectionConjunction/FilterCollectionDisjunction` Classes

- These two classes implement the `WatchListFilter` interface.
- The classes both have a `List` of objects of type `WatchListFilter` denoted `aFilters`.
  - The functions, `addFilter(WatchListFilter pFilter)` and `removeFilter(WatchListFilter pFilter)` are used to add and remove filters to `aFilters`.
  - `aFilters` is a collection of filters that we want to apply to a `Watchable` object to determine whether it should be included in a `WatchList`.
  - The functions `filter(Movie pMovie), filter(TVShow pTVShow)` and `filter(Episode pEpisode)` return a `boolean` value depending on whether or not the parameter (argument passed in) fits the criteria of the filters in `aFilters`. In the case of class `FilterCollectionConjunction`, the `Watchable` must fit the criteria of ALL of the filters in `aFilters` for the function to return `true`; however, in class `FilterCollectionDisjunction`, the `Watchable` object needs to fit only the requirement of one of the filters in `aFilters` for a return value of `true`.
  - The function `getFilters()` in both classes returns a deep copy of all the filters in `aFilters` . This ensures that the filters are not tampered with unnecessarily.

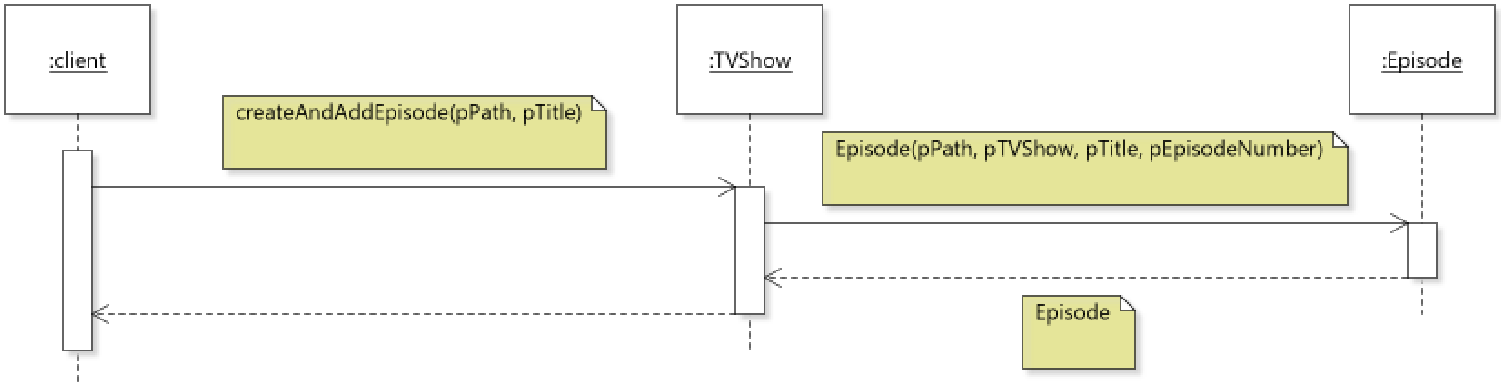
`TVShow/Episode` Classes

- These classes implement the **prototype design pattern** in which new objects are created by cloning a pre-existing prototype associated with the object. This design pattern is useful when many instances of objects that are similar need to be created as it conserves resources and allows for faster creation of objects.
- Field `aPrototype` in class `TVShow` is used to store the prototype of an Episode associated with the TVShow.
  - The methods setPrototype(Episode pPrototype) and getPrototype() allows aPrototype to be initialized and retrieved. The setter method is private, so it is not easily accessible and modifiable by the client.
- The method createAndAddEpisode(File pPath, String pTitle) is used to create Episode objects associated with the TVShow. The Episode constructor is not directly accessible to the client as it is a package-private constructor, so this method is the only way for clients to create an Episode.
  - When the first Episode associated with a TVShow is created using this method, the constructor for Episode must be called as there is no prototype for and Episode of the TVShow yet; this first Episode is then set as the prototype Episode for this TVShow and is stored in aPrototype.

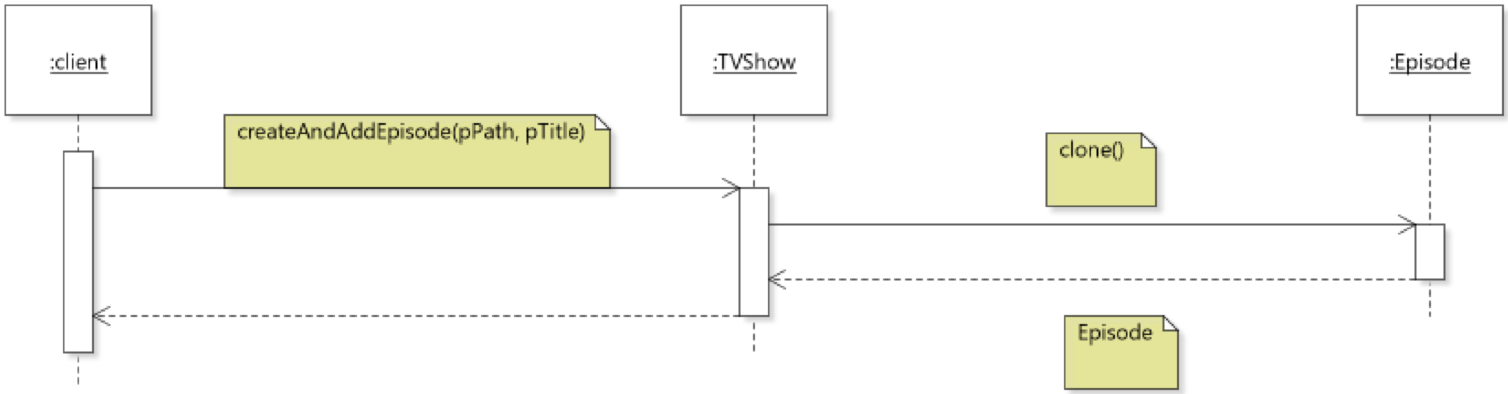
- When subsequent Episodes associated with the same TVShow are created using this method, aPrototype is simply cloned using the clone() method from class Episode, and the fields are updated using their respective setter methods defined in class Episode.

The sequence diagrams below show how a new Episode is created in relation to its TVShow:

1. Sequence diagram of the first Episode object created for a given TVShow:


:client
:TVShow
:Episode
createAndAddEpisode(pPath, pTitle)
Episode(pPath, pTVShow, pTitle, pEpisodeNumber)
Episode

2. Sequence diagram of subsequent Episode objects created:


:client
:TVShow
:Episode
createAndAddEpisode(pPath, pTitle)
clone()
Episode