For my design I decided to use a Singleton design pattern for the Library class with a private constructor, and a getter to retrieve the single instance so that there could only ever be one instance of the Library at any time during operation. For the library class I also decided to use an Optional type for the natural language decision since a client could provide one but doesn’t have to. For the Song and Podcast classes I decided to use the Flyweight design pattern so that no duplications can occur. Clients can compare 2 playlists which check if either songs are the same object or if episodes have the same number title and are associated with the same Podcast object, and if the songs/episodes appear in the same order. I also made sure all classes were encapsulated with final/private variables and returned appropriate values to ensure no information leakage.