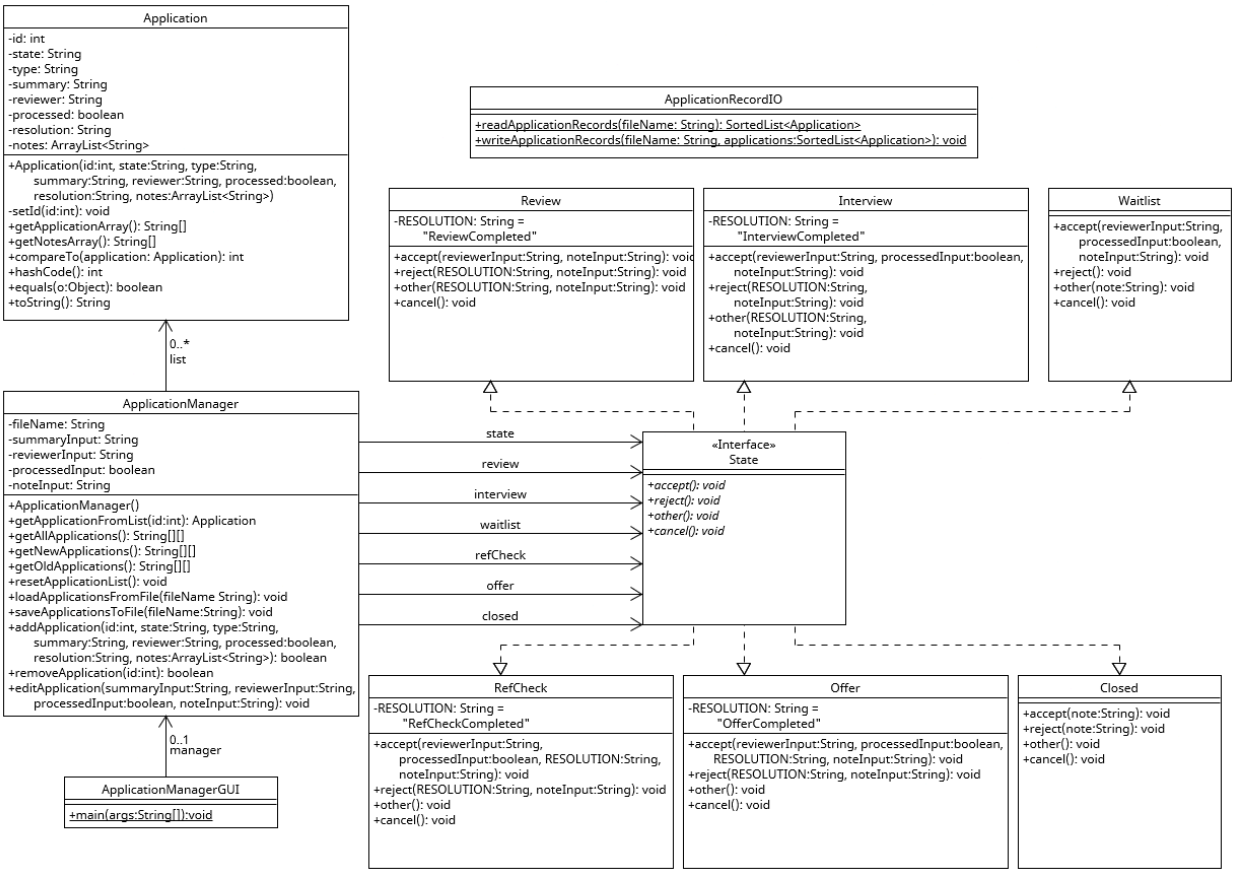
ApplicationManager: Design Proposal

Author: Christine Weld
Date: 10.3.24

UML Class Diagram



Design Rationale

The proposed design for the ApplicationManager system consists of five subpackages: application, manager, state, io, and ui.

The application package contains the Application class which represents a single Application. It has a constructor that assigns the fieldsid, state, type, summary, reviewer, processed, resolution, and notes. These fields are described in the requirements and have getter and setter methods. The notes field is an ArrayList<String>, for the functionality of adding as many note elements incrementally as indicated during the application review process.

The setId() method is private, as once the Application is constructed with its id, it should not be changed thereafter. This class includes methods getApplicationArray() and getNotesArray() whose data are String arrays to display the Application's fields and notes array as needed in the GUI. There is a compareTo() method to facilitate sorting by id number, as well as standard hashCode, equals, and toString methods.

The manager package contains the ApplicationManager class that utilizes a composition relationship with the Application class. It instantiates a list of Applications, is the main context class for the state interface, and interacts with the ui package and GUI to receive and process input from the user. The getApplicationFromList method returns a single Application as needed for reference to other methods. The list can be modified with the addApplication() and removeApplication() methods, and reassigned with an empty list with the resetApplicationList() method.

The class has field fileName, and methods loadApplicationsFromFile and saveApplicationsToFile, to read and write data to file with calls to the io class. It has methods getAllApplications(), getNewApplications(), and getOldApplications() for displaying a list of Applications depending on type to the GUI. Its fields summaryInput, reviewerInput, processedInput, and noteInput hold the input data from the user via the GUI for each editApplication() method call. This method call allows for changes to the Application's state based on user input.

The state package contains the State interface and the concrete classes Review, Interview, Waitlist, RefCheck, Offer, and Closed.. These classes utilize an inheritance relationship with the State interface, and implement the abstract method signatures accept(), reject(), other(), and cancel().

These four methods represent the different options of user input available in each state. For example, the Review class implements the accept() method and reassigns the Application to the Interview state. The other() class could implement a Standby input, or other interaction as needed. The Review, Interview, RefCheck, and Offer classes would have their own String constant RESOLUTION which would update the Application's resolution field when moving from one state to another. Additionally, every time the Application changes state, a note element would be appended to Application's note ArrayList.

The ui package contains the ApplicationManagerGUI class which is detailed by another team, but includes the main method and starts the program. This design utilizes the Model-View-Controller (MVC) pattern, with the View-Controller as the GUI class, and the model including all other classes in the program.