SE3PROJECTA DEVELOPER GUIDE

Java Documentation

Java documentation is available for this project, and can be found in the SE3ProjectA/doc/javadoc directory.

Design decision Rationale

Selection of seat type/quantity before arrangement selection

The decision was made to force the user to select the number and type of tickets they want before selecting their seating arrangements as this makes random allocation to simplify the random allocation of seating process. Especially for random allocation in a block, the number of seats needed to be known before the step where users select which seat they want to book.

Use of XML files as a persistent storage medium

The decision was made to use XML files to store persistent data, as unlike other data storage options such as JSON, XML does not require 3rd party libraries, as an XML library is included with Java.

As it is plain text, XML data is easily editable for configuration purposes.

Unified repository class

The unified repository class makes passing data around the project easy, in that any element of the GUI can retrieve data from the repository.

In future the repository class could be turned into an interface or abstract class, and it will

Event Orientation

The GUI is oriented around events through the Transaction and Allocation classes. This means that when a Transaction changes, parts of the GUI update.

This was implemented because having each Panel reference each other would become unmaintainable in short order. This also means things can be added to the GUI easily; e.g. a total price widget could simply subscribe to the allocationsChanged event so that price is updated whenever allocations are changed – without knowing anything about the widgets changing the allocations.

State machine

The GUI maintains a state machine which controls which elements are shown or hidden. This was done to minimize coupling between the GUI classes, however this also means we need to pass the GUI class to everything which changes the state.

Assumptions

1. It has been assumed that each theatre only ever shows one movie.
2. There is only one day of bookings considered within the application. The application could simply be extended to support infinite dates, though as the repository class loads every

Theatre Session when the GUI is loaded, this could cause problems in future (as instead of loading data for perhaps 20 theatres, you would be loading data for perhaps 20 theatres times (30 following days + every day before today)

3. Currency uses the same format as US currency (specified within Money.java)
4. It has been assumed that the end-user will not need to add theatres or movies in future. However, it is possible for this to be accomplished by changing the contents of the XML files in which the entities are stored. Note that the system assumes that these files will never change, and will not necessarily respond correctly if they are.