This is a concise description of the system as a whole, written mainly from a developer perspective. It covers the essentials of architecture, control flow, privileges and exception handling.

The system implements an MVC architecture. The servlet class (src/GoTournaments.java) acts as the main controller, which receives an HTTP request, delegates the request to other components and finally sends an HTTP response back to the user.

The HTTP request is received in the service() method and passed on to produceResponse(), which will perform the request and return either an HTML page or a redirection URL, which are then sent to the user by service(). The produceResponse() method is expected to catch all exceptions and return an error page to report them. Therefore, any exception thrown by produceResponse() or within service() is unexpected and is to be considered a bug or an error in the environment. If possible, an exception occuring at this stage is reported to the user by the servlet. If not, it's up to the environment how to handle the error.

The produceResponse() will first check if any error occured during the initalization of the system in init(). If there was an error, an error page is returned and the user request is ignored. Otherwise, the request is passed on to serveRequest(), which will do the actual job of interpreting the request, performing it and generating the response. The serverRequest() method may throw an exception in case of a normal use error, such as an unauthorized request or an attempt to access data that has been deleted. These exceptions contain a clear error message. They are caught in produceResponse() and returned to the user as error pages. An unexpected exception from serveRequest() is also caught and returned as an error page with a stack trace.

The serveRequest() method is the part, which performs the interaction between the controller and the actual data contained by the sytem, known as "model" in MVC. The method extracts all data required from the request, and if needed, accesses the model via the TournamentManager class (explained further on). Simple requests are implemented as single method calls, more complex ones request a special transaction object from the TournamentManager, which simplifies the process. Once the data has been updated and/or extracted, the method will produce a response for the user by constructing either an HTML page or a redirection object. In MVC this component is commonly known as "view" (explained further on). If the serveRequest() finds that the user does not have the privileges required, an exception is thrown. An exception is also thrown if the request is otherwise faulty. If it cannot be interpreted at all, an error page is returned. The method is synchronized to prevent concurrent access to the model, which is not thread safe.

The model (src/model/) is the part of the system, which handles data storage, access and update. Its main class is TournamentManager, which acts as the facade to the controller. In theory, the model is entirely separate from the rest of the system. If necessary, the controller and view can be changed or even completely replaced to create a new kind of user interface. The inner workings of the model are described in an accompanying diagram.

The view, (src/view/) contains classes for all page views (src/view/page/) and redirection URLs (src/view/url). The pages are constructed using an XHTML library (src/view/xhtml). The library is not a complete one and has been designed particularly for the Go system. It should be relatively simple to extend, though, if needed, by following the javadoc documentation. The pages contain a method to extract their HTML code as a simple String, which is sent to the user in the servlet.