Logger Requirements Group 66

The functional requirements of logger within the Bust-A-Move game are grouped according to the MoSCoW model. The four groups are: Must haves, Should haves, Could haves and Would/Won't haves. These requirements cover the functionality of the logger.

Functional Requirements

Must haves

- The logger must write the contents to a file, which must be in a '.txt' format
- The logger must log when a ball is shot
- The logger must log when a ball is hit
- The logger must log when a ball hits the wall
- The logger must log when balls are popped
- The logger must log the start time of the game
- The logger must log when a level/the game is lost
- The logger must log when a level/the game is won
- The logger must provide timestamps in front of all the messages using the following format: yyyyMMddHHmmss

Should haves

- The logger should log the end time of the game
- The logger should log the cannon angle when shot
- The logger should log score changes
- The logger should log the number of balls in the grid/game
- The logger should distinguish severity levels ERROR, WARNING, INFO, DEBUG and and in the code specify the maximum severity level to be logged. This means that when WARNING is selected, ERROR and WARNING messages will be logged, and INFO and DEBUG messages will be discarded.
- The logger should create a new log file when the game starts and not use a log file of a previous run.
- The logger should log in a human-readable format
- The logger should log in the line format: "[timestamp] LEVEL: message"
- The logger should be able to also provide console logging

Could haves

- The logger could log the key-presses
- The logger could write the log output in an asynchronous thread
- The logger could log all the object creations

- The logger could log the location of a ball when it is added to the grid
- The logger could also provide logging to a html file

Would/Won’t haves

- The logger won't log the current animation state
- The logger won’t log the ball coordinates when in flight
- The logger won’t log the real-time direction of the cannon

Non-functional Requirements

The logger will be implemented using the “Chain of Responsibility” design pattern which has an abstract logger class which will cycle through the needed log implementations such as log to file and log to console classes that extent the abstract logger. This will create an extendable log system.