The lifetime of an incoming socket

1. The *IntellectualServer* collects any incoming requests and sends them to *SocketHandler#acceptSocket.*
2. The *SocketHandler* acts on the incoming socket.
    a. All active socket filters are applied to the socket, filtering out any unwanted requests.
        i. If the filter (a predicate) returns a negative result, the socket is immediately shut down through *SocketHandler#breakSocketConnection* **[END]**
    b. The socket is sent to an available *Worker* through the *ExecutorService* instance. When available *Worker#run* is called.
3. If the socket is active it is handled by *Worker#handle(Socket).*
    a. The raw input is read into a list of strings.
    b. A request is created.
        i. The protocol type is determined.
        ii. The input is once again read, in search for requested query and incoming headers.
        iii. The headers are handled.
        iv. The query is read, and handled.
        v. Cookies are read and stored.
        vi. If authorization is provided, it is read.
    c. The request is handled by *Worker#handle.*
        i. The request handler for the request is fetched by *Router#match*, 404 if no appropriate generator is present. **[abstract]**
        ii. A session is fetched or created (then stored).
        iii. The request is validated through any given validators.
        iv. The server checks if a cached response exists for the response:
            If it does: The cached response is read through *Server#getCacheManager#getCache(RequestHandler)*
            If not: *RequestHandler#handle* is called.
                i. Middleware is allowed to act on the request.
                ii. If an alternate outcome, or redirect has been requested, it is handled.
                iii. The response is generated **[abstract]**
        v. Cache may be stored now.
        vi. All handlers are allowed to act on the response.
        vii. The headers are printed to the remote socket.
        viii. Output is compressed (if md5 compression is enabled).
        ix. The data is written to the remote socket, and the output is flushed.
4. The socket is closed **[END]**