**Overview:** the client only talks to the server, who in turn will contact the microservices when necessary.

**Division of Functionality:**

* The client is only responsible for generating UI and printing server status messages.
  + The client will alter between 2 menus:
    - A main game menu featuring functions such as adding/removing a word from the database, and starting a new game/loading an existing game.
    - A game menu where the client may guess individual letters or words, and query the database for the existence of a word.
* Game generating/processing, request to login, and database querying is handled by the main game server.
* Authentication, saving/loading of user data, and checking the client heartbeat is handled by the user account microservice.
* The word microservice is only responsible for adding/removing words, or returning a word based on a specific condition (e.g., a minimum specified length, must contain a specific character, etc.).

**RMI Objects:**

* ServerFactory- Creates server objects for multiple clients. Only has method createServer().
* ServerInterfaceImpl – server object, stub instantiated in client. Main public methods fall into 2 categories: login methods, game processing methods.
  + sequence: Variable storing information regarding rmi calls that have been made to it.
  + Login methods: all of the following invoke a nested RMI involving the user account microservice object; all except SaveData expect a username string as the sole parameter.
    - CheckValidUser – login flag returned
    - ValidateUserData – UserData object returned
    - LogoutUser
    - SaveData - expected parameter is a UserData object
  + Game processing methods: all of the following expect a UserData object, and an input string
    - ProcessUserInput – modified UserData object returned
    - ProcessWordQuery – status message string returned

ProcessPuzzleGuess – modified UserData object and status message returned in ActiveGameData, a wrapper object

* + Heartbeat validation: the user ‘heartbeat’ passes through to the UserAccountServer, where it is fully managed.
* UserAccountServer – user account microservice object, stub instantiated in main server; methods included: login, logout, load, and save, which perform the corresponding functionality associated with ServerInterfaceImpl’s login methods. Also features validateHeartbeat, to record received ‘heartbeats’ from the user, and heartbeatMonitor, to periodically remove those users that are no longer sending ‘heartbeats’.
* DatabaseImp – word microservice object, stub instantiated in main server, makes sql queries on a sqlite database. Has methods
  + addWord-adds a word
  + removeWord- removes a word
  + establishDatabase- Starts the SQL
  + randomWord- gets word by character
  + randomWordLength- gets word by length
* Database – stub for database implementation
* DatabaseServer-Starts the server