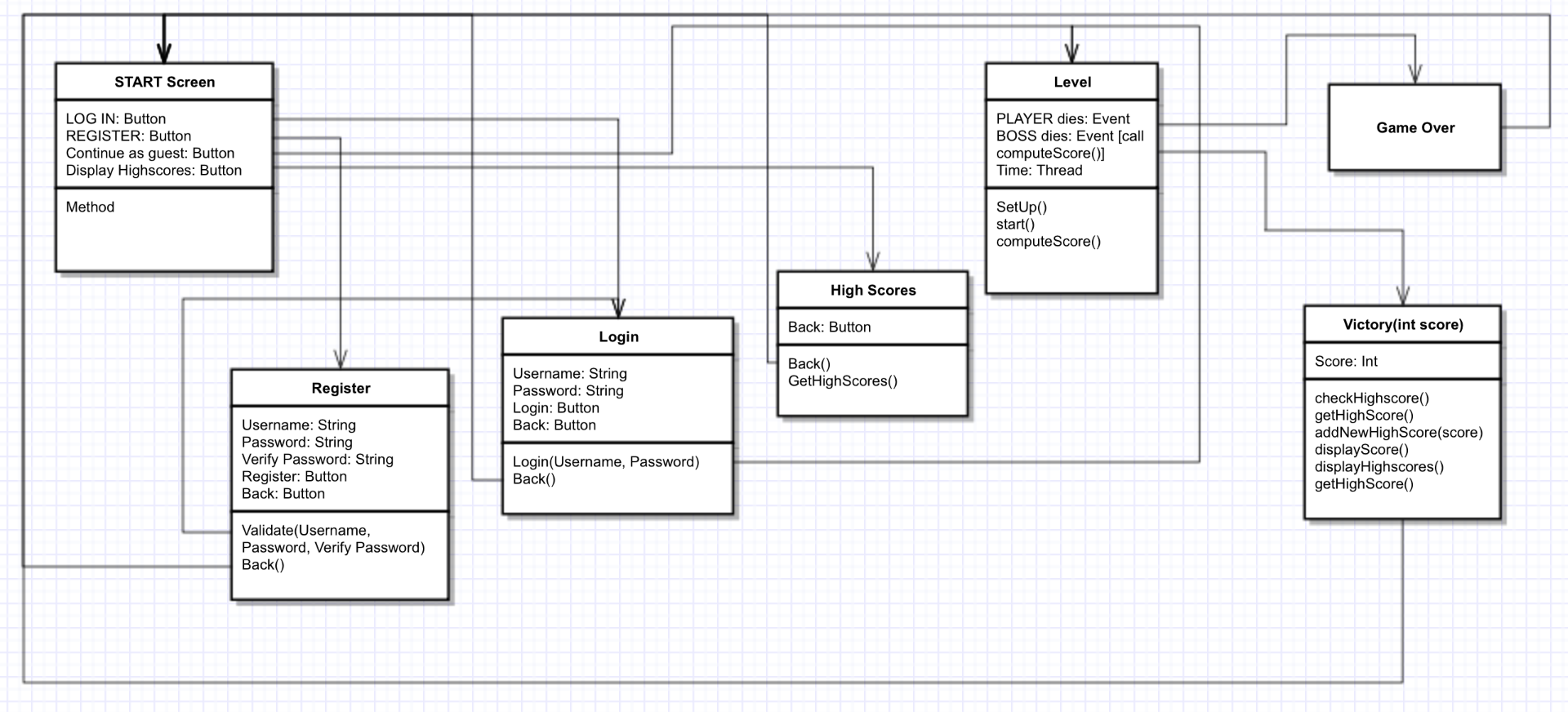
***Moonbase - Detailed Design Document***

Jun Yen Leung - junyenle@usc.edu

Ethan Barker - esbarker@usc.edu

Jake Leventhal - jglevent@usc.edu

Uri Rahimi - urahimi@usc.edu

**Section 1: Broad Program Structure**

*(some features removed.. see below for more details)*

**1a: Flow and Overview**

* **Program Flow From Application Run:**
  + **START screen →** 
    - **Has Login, Register, Continue as guest**
    - **Login screen (on Login button click)**
      * **Has Username field, Password field, Login button, and Back button.**
      * **Login button calls login() function, which will check the username and password against the server.**
      * **Further explained in data section below.**
      * **If successful, redirects to GAME PICK SCREEN.**
      * **Back button redirects to START SCREEN.**
    - **Register screen (on Register button click)**
      * **Has Username field, Password field, Register button, and Back button**
      * **Register button calls validate() to check Username and Password fields’ validity (check password not null and that username not taken).**
        + **If validated successfully, calls on the RegisteringUserThread on the server to register user.**
      * **Further explained in the data section below.\**
      * **If successful, redirects to START SCREEN**
      * **Back button redirects to START SCREEN.**
    - **Continue as Guest** 
      * **Redirects player to GAME PICK SCREEN.**
  + **GAME PICK →**
    - **If logged in, allows to pick 2P game or 1P game**
    - **Otherwise, only allows 1P game**
  + **LEVEL →**
    - **setup() will set up the game**
    - **update() will update the game state every frame**
    - **GAME OVER occurs upon reaching the spaceship at the end**
    - **More level details in section 1b below.**
  + **GAME OVER →**
    - **Displays text and an endgame screen.**

**1b: LEVEL specifics**

**Player will be able to move and jump through a 2D world. Upon reaching the end, they will get in a spaceship and fly away. This ends the game.**

**Section 2: External Technologies**

**2a: Programs / Tools**

* **The majority of actual application design will be done in Java Swing.**
* **Sketches, concept art, and game art will be drawn in Adobe Photoshop using Wacom Intuos 4.**
* **Model vectoring will be done in Inkscape / Photoshop and converted into animation sprite sheets in Inkscape.**
* **Voice lines will be recorded in Audacity.**
* **Music score will be written in MuseScore and Bosca Ceoil**

**2b: Libraries**

* **JDBC**

**2c: Programming Languages**

* **Server logic: Java**
* **Client logic: Java**

**Section 3: Data Management**

**3a: Database Schema**

* **Single collection MySQL database.**

**3b: Data Structure**

* **Sample login data collection:  
  {  
   username: “ttrojan”,  
   password: hashed password here  
  },  
  {  
   username: “bbruin”,  
   password: hashed password here  
  }**

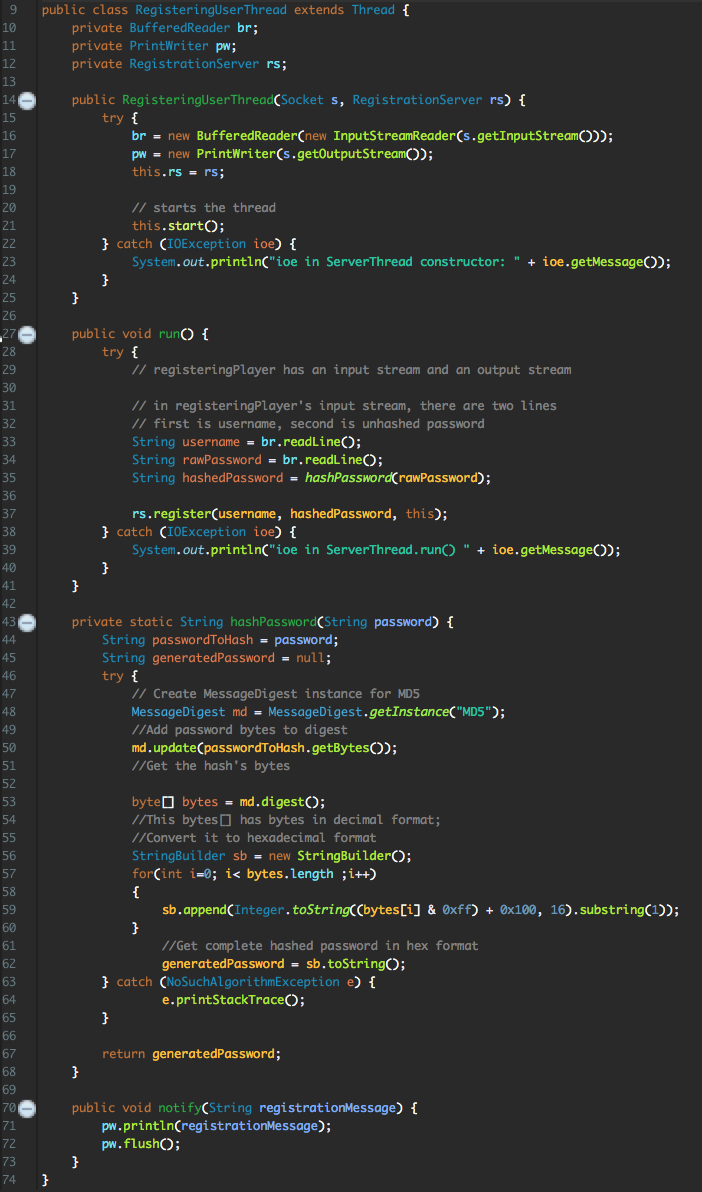
**3c: Pushing and Pulling Data**

**3ci: User Registration**

* **At the home screen (start screen), users will be prompted to either login or register which both will be making two separate calls.** 
  + **Registering: There will be three fields for the user to enter data into. First, the user will need to input a username, password, and password confirmation. Upon registering, a validate() function will be called to make sure that the user has entered valid input.**
    - **If the username that the user is registering with is already taken, an error next to the username field will be displayed.**
    - **If the password field contains text shorter than 6 characters, an error will be displayed next to the password field.**
    - **If the text in the password field is not equal to the text in the confirm password field, an error message will be displayed next to the confirm password field.**
    - **If all the fields are valid, a RegisteringUserThread object is created to represent the user that is registering. This thread contains member variables for a PrintWriter, a BufferedReader, and a reference to the registration server.**
      * **When the RegisteringUserThread is created, the username and password to register with is read from its socket’s input stream.**
      * **The password is then hashed so that I can be stored securely using the RegisteringUserThread’s hashPassword(String rawPassword) function.**
        + **This function performs bitwise operations on the password to transform it to a 32 character hashed version of the Password.**
      * **The registration server’s register(String username, String password, RegisteringUserThread registeringUser) method is then called.**
        + **This adds the new username and password to the Mongo database and then notifies the user of successful or failed registration via the registeringUser’s notify(String registrationMessage) method.**

**This method simply flushes the registrationMessage to the RegisteringUserThread’s output stream so that it can be displayed on the client’s front end.**

* + - **After the registration is successful, we will trigger a text-to-speech event notifying the player that registration was a success.**
    - **The code for what RegisteringUserThread, including hashPassword(String rawPassword) and notify(String registrationMessage) functions.**

****

* + - **After registration, redirect to the login screen.**

**3cii: User Login**

* **The user login will simply check and verify that the username and hashed password matches a corresponding existing entry in the username/password database table via the login() method.**
  + **This will make a call to the LoginServer which will make the actual database check.**
    - **If there is a match for the username and the hashed password, the user will be logged in.**
    - **Otherwise, an error message will be displayed notifying the user that the username and password combination is invalid.**
* **Once this has been accomplished we will trigger a text-to-speech event welcoming the player to the game.**

**Section 4: Graphics**

**4a: GUI Design**

**4a0: Global Specifications**

* **The application will run in fullscreen only. The GUI will be designed for 1080p (1920x1080 pixels) screens.**

**4ai: Pre-LEVEL**

* **The first screen shown to the user (START screen) should have three buttons: Log in, Register and Continue as Guest.**
* **Behind these buttons, there should be START screen art.**
* **The START screen should look like this:**

**4aii: LEVEL**

* **The playing arena on which the actual gameplay will take place (LEVEL) will be a long platform.**
* **The platform will exist on the foreground.**
* **The foreground is the only level where character hitboxes will exist.**

**4aiii: Post-LEVEL**

* **The post-LEVEL screen should simply say Game Over and show a nice post-game graphic.**

**4b: Modelling and Animation**

**4bi: Environment**

* **The LEVEL’s background (the sky and mountains in the figure in 4aii above) should have some looping animations. For instance, there might be moving clouds. The LEVEL art will be painted in Photoshop.**

**4bii: Player Character (PC)**

* **The PC will be drawn by hand, vectored in Photoshop / Inkscape and converted into sprite sheets for animation in Inkscape.**
* **PC actions will be animated in the same way.**
* **Miscellaneous looping animations (e.g. clouds, waterfalls on the mountains) will be drawn in Photoshop. These will be applied in separate layers with equal parallax factors to the layers that they correspond to so as to seamlessly integrate them into the static art.**

**Section 5: Sound**

**5a: Soundtrack**

* **All soundtracks will be written using MuseScore or Bosca Ceoil**

**5b: Sound effects**

* **All sound effects will be recorded with Audacity.**

**Section 6: Game Mechanics**

**6a: Player Character (PC)**

**6ai: Movement**

* PC can move left/right with the left/right arrow keys and jump with the up key.

**6b: Hostile Agents**

**These do not exist.**

**6c: Environment**

**6ci: PC Collision**

* Each tile will be an object with which we can detect collision with the PC.