Design Document

1. Purpose: We will be attempting to make a fully functional programming game. The game will be educational in that it will constantly test users on their knowledge a programming language. It will be a puzzle game as well in that the user will be traversing through a maze while answering these programming questions. Our goal is to make this app mobile as well as directed towards younger programming students. Because of this, we will make the user interface relatively simple.

1. Non-Functional Requirements:
   * Performance is going to be the most important non-functional requirement of this project. We want the application to run as efficiently as possible to allow smooth playing of the game.
   * Security won’t require too much focus which will allow us to focus more on performance. We simply want to make sure that the questions we create for the game are secure.
   * Efficiency of the memory usage of our app will also be fairly important. We want the app to be able to run effectively without taking up entirely too much memory while running in the background
2. Design Outline
   * Design Decisions
     + We are planning to use the Transaction-Processing architecture.
     + There will be a Menu UI which will allow the user to adjust settings before starting the game.
     + After moving through the Menu UI, the user will move to the Game UI which will contain the map of the maze and movement controls
     + When an obstacle is reached, the user will move to the Question UI which will contain the programming question and the answer buttons.
     + After the question is answered, the user moves back to the Game UI.
   * Design Interactions
     + We will use separate handlers to respond to any input given by the user throughout all of the UIs.
     + The user will progress from the Menu UI, to the Game UI, to the Question UI, and then alternate between the Game UI and the Question UI until the game ends.
3. Design Issues
   * Issue 1: How do we want the UI to display the game and questions?
     + Solution 1: Display the game on one interface with a space allocated for questions when an obstacle is reached.
     + Solution 2: Display the game on one interface, and when an obstacle is reached, bring the question up on a separate UI.
     + Decision: We chose Solution 2 because this will allow for a less crowded screen, and a more simple design for our interfaces.
   * Issue 2: Do we want the app to be online or offline (in regards to leaderboard and question database)?
     + Solution 1: The leaderboard and question database will be completely offline.
     + Solution 2: The leaderboard and question database will be online.
     + Solution 3: The leaderboard and question database will be offline to start, and will be set up to allow for a later online implementation.
4. Design Details
   * Main Menu Interface
     + Contains all options for creating and continuing a game, accessing settings and high scores, and exiting the game.
     + Will use handlers to link with the initialize and continue classes as well as the high score and settings classes
   * Settings
     + Will have a list of settings that will allow the user to change certain features of the game, including difficulty level and programming language of choice.
   * High Score
     + Encompasses methods to store and display high scores
     + Can be linked to from the main menu and can be accessed by the exit class when writing a new score at the end of a game.
   * Initialize
     + Will include methods necessary to configure the settings required to start a new game, as well as methods to initialize the maze and question interfaces.
     + On exit of initialization, the game will display the screen of a new game, which starts at the maze interface.
   * Continue
     + Will load previous saved data so the user can continue the game from where they left off previously.
     + Contains methods to load scores, maze state, and data about questions completed, and jump into maze interface.
   * Maze interface
     + Contains methods to display and update (repaint) the maze, as well as methods for moving through the maze, including jumping to the question interface when the user comes across an obstacle.
     + Links cyclically with the Question interface.
   * Question Interface
     + Has methods for loading questions from storage, processing user input, and reacting based on whether or not the user entered the correct answer
     + Links cyclically with the maze interface. After question process is complete, maze interface is loaded.
   * Exit
     + When game is completed, score will be submitted and high score screen will be displayed.
     + Will jump to main menu after high score list is displayed.

