High Level Description

Package game

- **EndGame** – This class will receive the wins of player one and player two as well as the number of games played. From this the class will make a call to DatabaseCalls so the information can be added to the database. It will also allow for a call to FrameDriver to initialize the EndScreen panel.

- **EndPanel** – This class is an extension of Jpanel. It will output all the data regarding the game statistics including player wins, who won, and the number of games played. It will allow the users to replay the game (which sends them to the SetupPanel already populated with the previous game settings), return to the Main Menu, or see the Head to Head history.

- **GameSetup** – This class serves as a storage of the desired game attributes. A player will select the attributes they would like on the SetupPanel, and the attributes will be saved in this class. This class will then be sent to locations such as GameMaster and MapPanel to ensure that these attributes are set.

- **Map** – This class is the Map object. It allows the game to easily set obstacles and bondaries of the map and retreive that data later. The map itself is set as a double array of Tile objects.

- **MapChooserButton** – Used for the MapChooser class, this MapButton class is an extension of Jbutton. It provides special instructions for styling the button which paints the map on the button. When the button is pressed the map drawn on the button will be set as the map for the game.

- **MapChooserPanel** – Allows the user to choose the map via clicking the MapButtons.

- **MapHandler** – Handles all the Map objects. This class saves all the Map orientations desired. It allows users to easily select between the Maps.

- **SetupPanel** – Allows the user to set their Cycle color, the Map, and the game difficulty.

- **WinCondition** – An enum that provides a typesafe way of storing EndGame logic. Options are PONEWIN, PTWOWIN, or TIE.

Package gameplay

- **Cycle** – Stores the current cycle heading, position, number, and alive status. It also contains a method that will increment the cycles position according to the cycle heading.

- **ExplosionPanel** – A panel which will display the explosion graphic.

- **GameDriver** – This class is responsible for controlling all the game logic. It instantiates the Cycle objects, PlayerControl, the MapPanel, the ExplosionPanel, and GameEnd. It will check the status of the game at each iteration to make sure neither of the cycles have crashed and provide graphical updates for the MapPanel to perform on a timer. When either of the cycles have crashed, the timer will stop, and a new timer will start which will instantiate and display the ExplosionPanel and provide updates to this panel. When that timer has stopped, a call to GameMaster will be made to end the game.

- **GameMaster** – Resonsible for much of the higher level game logic. Mostly responsible for starting the GameDriver, which starts the game, and starting the EndGame, which ends the game. It also stores the Timer for the game which regulates the speed of the game updates.

- **GamePanel** – This class is the Jpanel which displays the current state of the game. It will listen for keystrokes which will make updates on the cycle headings. It also has two buttons which will change the size of the map, as well as a string which will display the current score of the game.

- **PlayerControl** – This class will receive keystrokes from the MapPanel and map them to specific cycle heading updates.

- **Tile** – This is the enum which stores the possible states of the map location. It can either be TILE, WALL, PONE, or PTWO. PONE and PTWO correspond to PONE trails and PTWO trails respectively.

Package main

- **Connect** – This class is used to connect to the MySQL database which is hosted on a LAMP server on a Ubuntu 12.04 LTE server on the amazon web services. It contains a static method called 'Connect()' which returns a connection, and can be called from any class in order to get a connection

- **DatabaseCalls** – This class contains all the main database calls like Create User, Login and Push Statistics. The constructor initializes its own connection, which is used to create statements which are further used to run queries or updates on the MySQL tables. We have a main database which is called authentication and it has 3 tables: login, allStats and playerHistory which contain the login information, all statistics, and history of a player against all other players respectively.

- **Frame** – The Jframe for the game. This is instantiated one, there is only one per application launch.

- **FrameDriver** – Adds and removes the panels to and from the Frame.

- **Main** – Instantiates the Frame, FrameDriver, and Player objects

Package menuscreen

- **PlayerOnePanel** – The main menu panel corresponding to PlayerOne actions. This is an extension of PlayerPanel.

- **PlayerTwoPanel** – The main menu panel corresponding to PlayerTwo actions. This is an extension of PlayerPanel.

- **PlayerPanel** – A panel which allows users to login, logout, create a new user, see statistics, see head to head, or play the game.

- **PlayerStatus** – An AbstractAction listener that will update the Ready / Not Ready labels according to actions from key bindings on the Player Panels. Will only update the label when the player is logged in.

- **PlayerStatusListener** – A class that creates the ReadyAction key bindings and will listen to the labels for when they are both "Ready". At this point the game will launch.

- **WelcomePanel** – Simply stores and displays the Title labels for the main menu.

Package statistics

- **AllStatsPanel** – A class that creates a JTable, populates it with all statistics and adds it to a JScrollPane which is then added to the JPanel. The constructor does not take any input and simply makes the components and layout when called.

- **HeadToHeadPanel** – A class that creates a JTable, populates it with the head to head score of two players and adds it to a JScrollPane which is then added to a JPanel. Its constructor takes input as 2 usernames and initiates a query to the MySQL table using its connection to the database.

- **PlayerHistoryPanel** - A class that creates a JTable, populates it with the game history of the logged in player and adds it to a JScrollPane which is then added to a JPanel. Its constructor takes input as the username of the player and initiates a query to the MySQL table using its connection to the database.

- **TopTenPanel** – A class that creates a JTable, populates it with all statistics of the top ten players of all time and adds it to a JScrollPane which is then added to the JPanel. The constructor does not take any input and simply makes the components and layout when called.

Package styleelements

- **SyledButton** – A class that extends JButton that provides custom styling for the JButtons throughout our program.
- **StyledLabel** – A class that extends JLabel that provides custom styling for the JLabels throughout our program.
- **StyledPanel** – A class that extends JPanel that provides custom styling for the JPanels throughout our program.

Package user

- **CreateUserPanel** – This class is an extension of JPanel. This class creates a JPanel for creating a user account and integrates it into the main frame of the game. The class imports the class DatabaseCalls and uses its method createUser to run an update on the database tables. The class first checks if the entered password meets the minimum standards (1 uppercase, 1 lowercase, 1 number, 1 non-alphanumeric character, and should be at least 8 characters long) and then asks the user to re-enter the password to confirm it. If all conditions are met, the account is created in the database.

- **LoginPanel** - This class is an extension of JPanel. This class creates a JPanel for player login and integrates it into the main frame of the game. The class imports the class DatabaseCalls and uses its method login to run a query on the database tables.

The class first checks if the entered credentials match the ones that already exist in the database and then checks if the user is not already logged into the other panel. If all conditions are met, the user is logged in and can move ahead with the game.

- **NullPlayer** – An extension of Player which will differentiate between a logged in player and a non-loggend in Player.

- **Player** – Holds the userName and player number of the player. This is used mostly to provide GUI updates and regulate action of the program.