Soccer App

Purpose:

Considering the busy life we all lead, most soccer fans don't have enough time to browse about their favorite team's statistics or a player's info. Both Manchester United's website and UEFA's website are designed to aid in such endeavors. Combining the information presented by these two servers, this app provides succinct information on an intuitive user interface (UI).

Functional Overview:

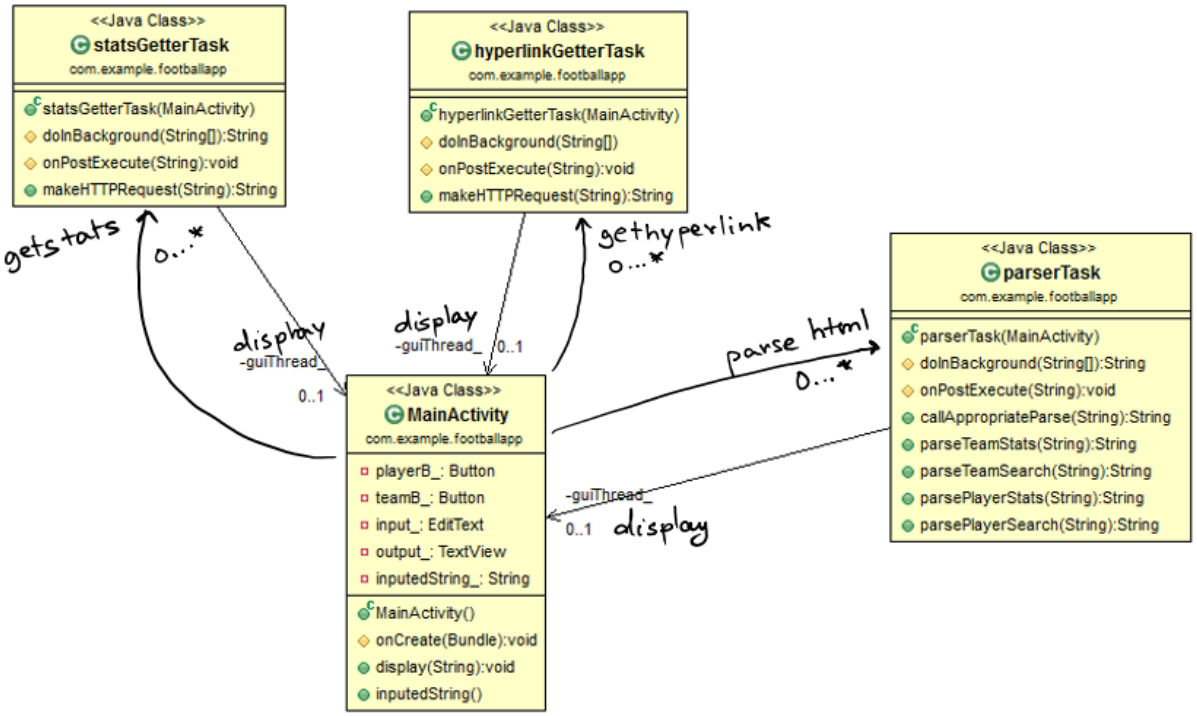
The UI presents the user with two options. The user can enter the name of a Manchester United's player or a UEFA ranked team. Upon entering either kind of query, the user has to tap the appropriate button. There are two buttons, one establishes a channel to the UEFA server while the other communicates with the Manchester United's server. Event listeners for the buttons instantiates a separate thread to fetch information from the appropriate server. The thread is a child class of the AsyncTask class provided for Android. Separate tasks fetch HTML documents that contain player and team info. After extracting appropriate information the UI is updated. If the user selects a player, the player's stats are presented. If the player doesn't exist, an appropriate message is presented. If the query resulted in multiple players a list of the players is provided. On the other hand, if the user selects a team, the rank of the team and 5 matches are provided. If the team isn't acceptable, an acceptable message is provided. As the app is fetching and parsing the information progress reports are provided to the UI.

Class Overview:

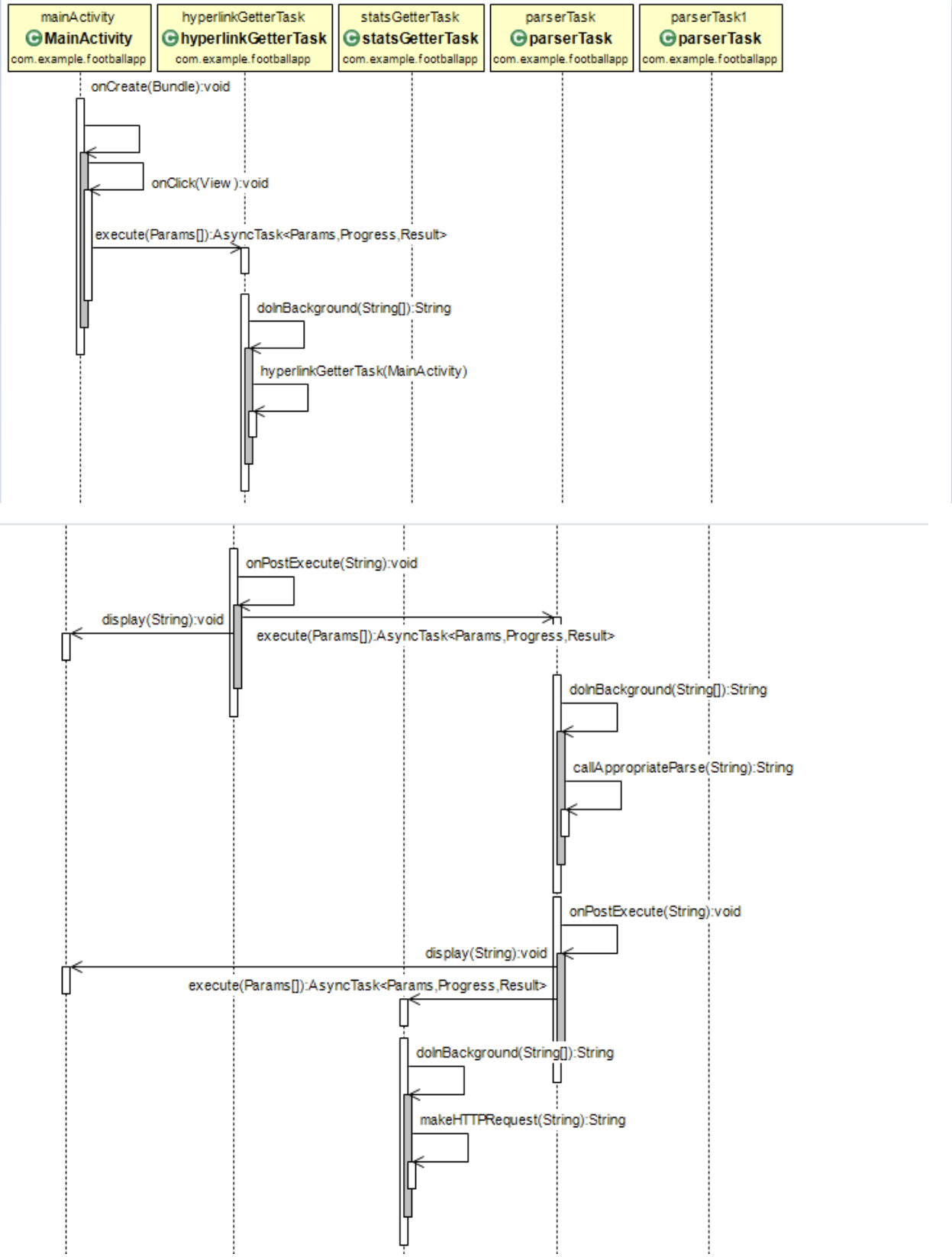
There are three extensions of the AsyncTask class, each of which extract different information. One extension is the hyperlinkGetter task, which gets the hyperlink corresponding to the inputted query. Depending on which button was clicked, this task fetches the primary HTML text from the correct server. The HTML contains the hyperlink for the team or player queried, which represents the resource that needs to be sent to the server to fetch the desired information. Once the HTML is sent to the app, a separate task is instantiated to parse and extract the hyperlink and the current task is terminated by displaying that the procedure is moving along without glitches. Another task is the parsertask, which extracts useful information from the fetched HTML. Using certain characters as the guide, this task fetches the correct hyperlink to request from the hyperlinkGettertasks's response. Once the hyperlink is extracted, the parsertask instantiates a statsGetterTask to fetch the resource. The parser task then terminates by posting an update to the UI. The other task is the statsGetterTask, which is instantiated by the parser task once the hyperlink is parsed. This task establishes connection with the correct server to send the correct resource. Once the server responds with the corresponding HTML, another instance of the parsertask is created and the HTML string is passed for information to be extracted from. For a player a table is extracted with information of the player, while for a team its rank and 5 matches are extracted.

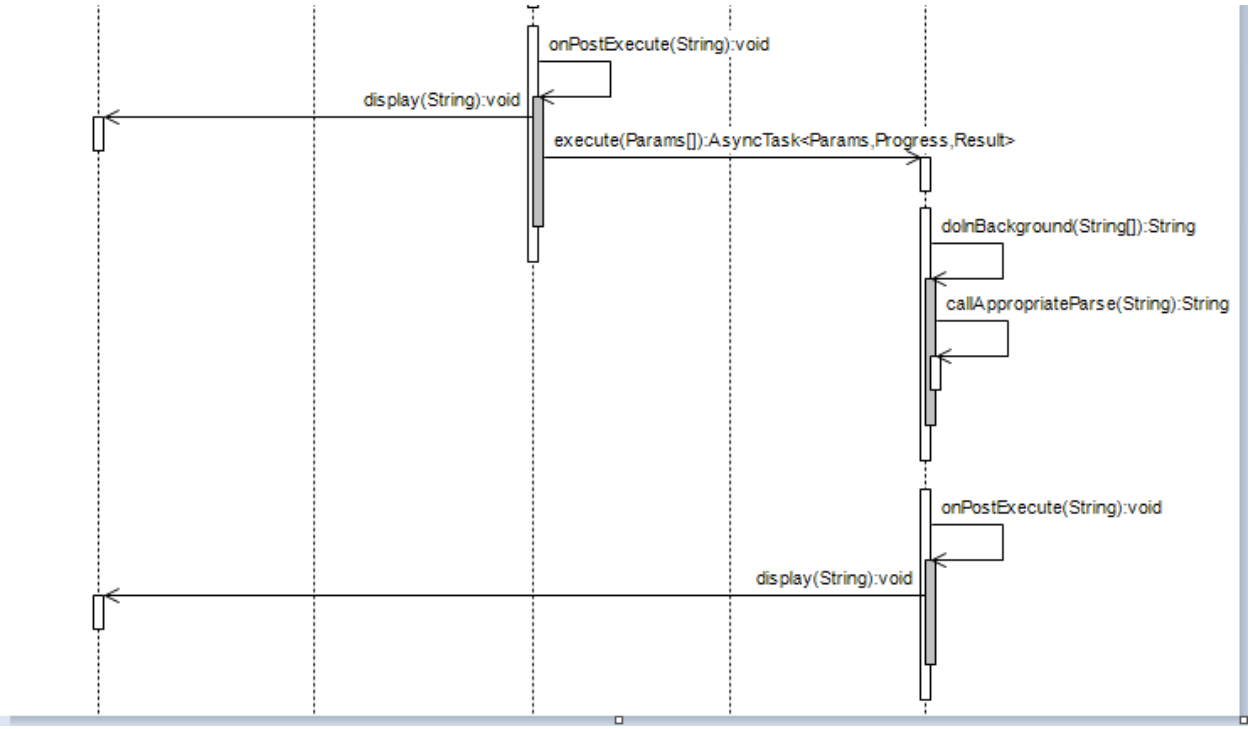
Class Diagram:

The tasks need to know about the main activity to call the display(…) method and to retrieve the inputted string. The main activity doesn't know about the parser but each for the task calls it in the UI thread.



Sequence Diagram:




onPostExecute(String):void
display(String):void
execute(Params[]):AsyncTask<Params,Progress,Result>
doInBackground(String[]):String
callAppropriateParse(String):String
onPostExecute(String):void
display(String):void