Documentation and Self Reflection

Chandler Smith - CS5004

## Introduction

The overall purpose of this project was to setup a relational database using a code structure that implements the core concepts of OOD. Throughout the code, you will see comments that explain what each method or piece of a method is doing. Additionally, “//TEST:” indicates where I tested the application after each method implementation. As this was a new concept in many regards, I relied heavily on external learning materials. I have cited these sources in my code but the code and overall design are my own.

## Self Reflection

## ***1. Review each of your objectives and explain if you met that objective and how you met that objective or if you did not meet that objective and why.***

**Goals Met:**

1. Setup a SQL or mySQL or Postgres database
   1. I successfully set up a PostgreSQL database that uses a JDBC driver to be editable by my Java codebase. This database was locally hosted and evidence that it works will be demonstrated in the accompanying video.
2. Create a table
   1. I created the table “athlete” to be able to store the players to the liverpool team. This table was editable via the database, IDE, and, most importantly, the GUI.
3. Allow the input of data -> adding to an existing table
   1. This table could be edited via the command line or the GUI. Methods that were used to edit this table are located in the DatabaseHelper.java file and include the ability to insert new data, return the data that is currently there, etc.
4. Execute SQL statements
   1. SQL statements were necessary for my ability to query the database. I used several to both add and return data from the PostgreSQL database.
      1. String SQL = "SELECT \* FROM athlete";
      2. "INSERT INTO athlete(first,last,age,Number)" + "VALUES( ?, ?, ?, ?)";
5. Use a GUI to search and return results
   1. While it doesn’t search, it returns and adds results. This intention shifted slightly.
6. Add the ability to create multiple tables
   1. The ability to create multiple schemas is there, and can be easily added to the schema of the database, but this was kind of a silly goal in retrospect.

**Goals Not Met:**

1. Create sortable lists
   1. I decided to have the GUI be able to add a player vs having a search interface. Adding a player was more complicated and I thought, more engaging.
2. Allow, based on a set of criteria, some to get matched with a player/rider, etc.
   1. I decided not to pursue this with the time given. It would have been a lot of busy work, adding data to the database, but theoretically could have been done with more columns.
3. Make the command line actionable.
   1. I ran methods from the command line, as indicated by me “TEST:” markers, but never enabled it to be truly actionable.

***2. If you had more time, what else would you do?***

* Better error handling, currently it just won’t let you add anyone without all their information.
* Searchable features to find a player
* Add more data tables - potentially for multiple teams
* If I really had a lot of time I could add player ability levels and, just like a battle driver, allow two people to draft teams and play them against each other.

***3. What did you learn from this project?***

*Implementation and solidified learning from coursework:*

* Implementing a design based on the MVC design pattern
* GUI implementation with action listeners
* Lists and converting lists objects to database rows
* Navigating and using existing Javadocs and Java packages
* Abstraction, separating out my GUI from my database data
* Utilizing helper methods and classes
* Multi class structure centered around player objects and helper classes

All things relational database:

* Establishing initial PostgreSQL database and connection
* The JDBC implementation which serves as a “Database Driver” (this was a huge challenge to overcome)
* SQL queries via Java - always good to gain exposure into a new language
* How to use a helper class to establish a database model instance

Specific Challenges:

* The biggest challenge I faced was writing code that engaged directly with the database. For many, many hours, I was unable to get any of my code to edit or add to my database which showed as “connected”. The solution was to implement a driver for my relational database, and have that interact with a DatabaseHelper class. The driver, JDBC, is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. Implementing this and getting the necessary packages in the build path resolved the issues I was having connecting to the database.
* Another Database challenge that proved difficult to diagnose and resolve was that each call to the database opened and closed the database connection. While this was not a problem as I individually tested the methods, it limited the GUI application’s functionality to one thing per session. Abstracting out the database connections from the methods and into the main method resolved the issue. From there, a simple refactor of the code to call an existing connection vs opening a new one enabled multiple actions to be taken to affect the database via the GUI.
* A third challenge was getting the database table to populate in the GUI. In order to actually allow the actual table data to populate, I had to create a separate data model class which calls the database each time. This enables it to be completely up to date every time it is loaded.

***4. Do you feel you would have done better or would you have preferred a structured project instead? Explain.***

This project would have been much easier if it was a specific project and, because of that, I really appreciated the freedom to pursue what was interesting to me. I don’t want easy, I want final projects to be interesting! Because I had to start from scratch and figure everything out, I was forced to explore and internalize much more than I would if my progress was structured. Half of what I read and learned I didn’t use, but I still absorbed and retained lots of new information. Furthermore, I know this process of setting up a working database quite intimately now. Keeping the project unstructured also allows students to keep working beyond the scope of the class if they desire which is very good for career prospects.

***5. What grade do you feel you deserve on this project? Explain.***

A-

I didn’t wow anyone with my GUI or my code, but I am proud of how I was able to figure out the nuances of implementing a postgres database design. Furthermore, I was able to deliberately abstract many aspects of the code and follow the MVC format successfully. I have concise, effective code with virtually no repetition. I used lists to engage with my database and produced a project that implements many of the concepts we covered throughout the course. Understanding the infrastructure was time intensive, but the new skills I acquired during this project add substantially to the existing course.

## 

## 

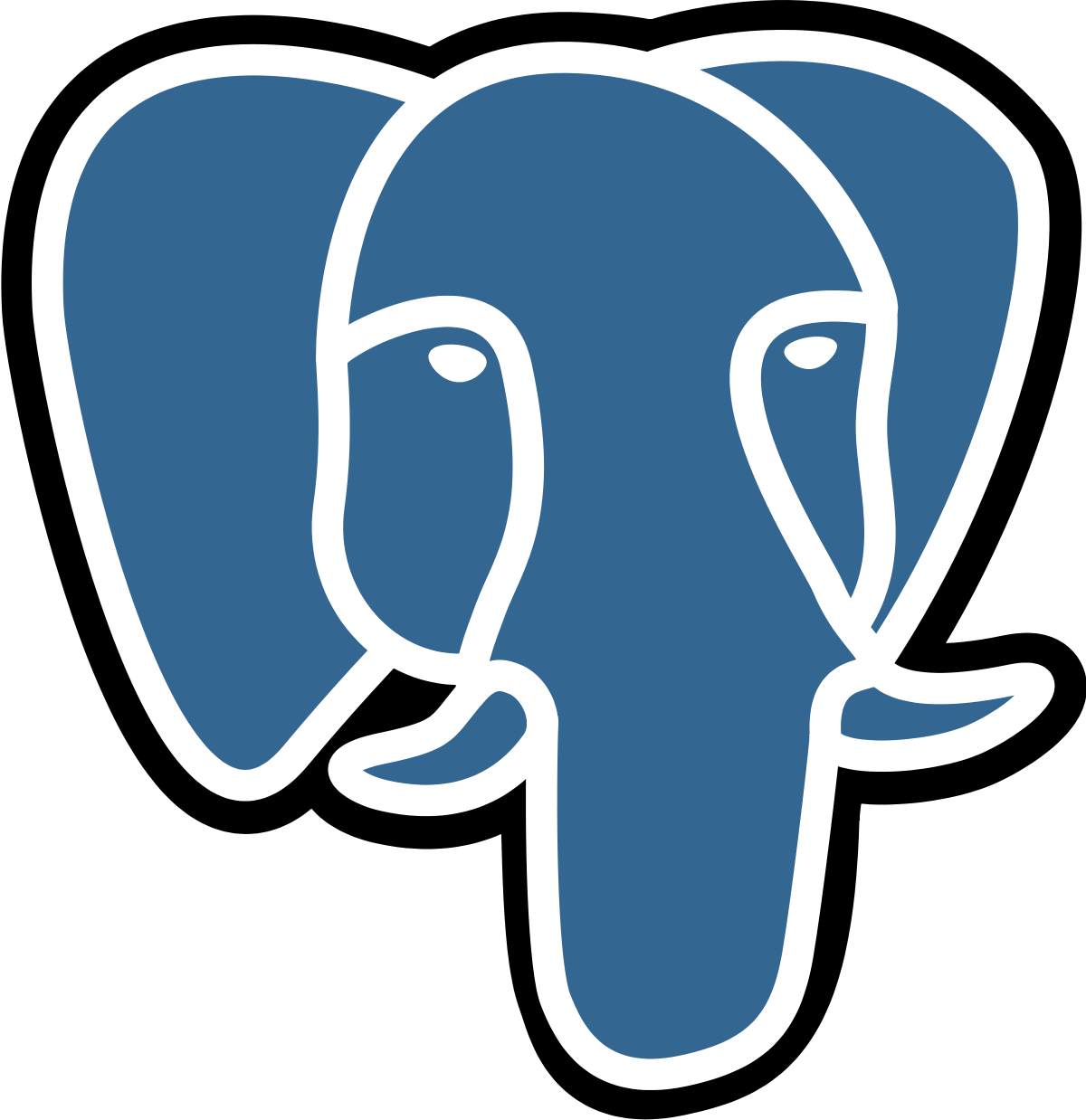
## Updated Design:

## 

## 







## 

## JDBC Database Driver

## Initial Proposal

**Description:**

The overall goal of this lab is to take some X set of data (Tour de France, Liverpool, etc) and set up a SQL database for it. To do this, I will get and input data from my source and setup a mySQL/postgreSQL database that can be sorted and manipulated. The console and the database will work together to be able to achieve this. A stretch goal will be to use a GUI to do this. Another stretch goal will be to use a file import that I can clean and use in the database vs manual entry. I am going to need my regular src code, a target file and then a test file in addition to my database.

**Objectives**:

**Primary Goals**

1. Setup a SQL or mySQL or Postgres database
   * This SQL database must be searchable
   * I must be able to add and remove data from it
2. Create a table
3. Allow the input of data -> adding to an existing table
   * I need an importer file that stores additional data onto a table that lives within a Java file
4. Execute SQL statements
5. Make the command line actionable.

**Secondary Goals**

1. Create sortable lists
2. Add the ability to create multiple tables

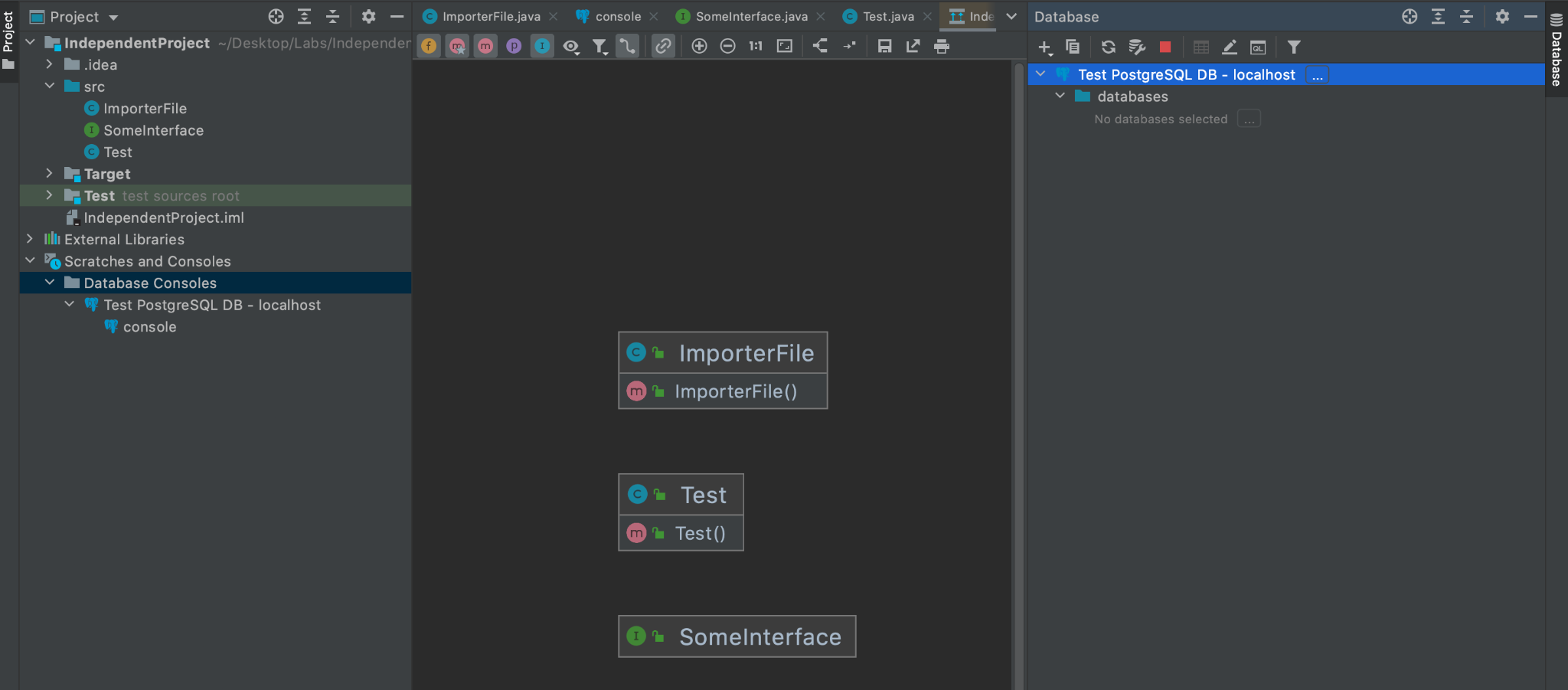
**Tertiary goals**

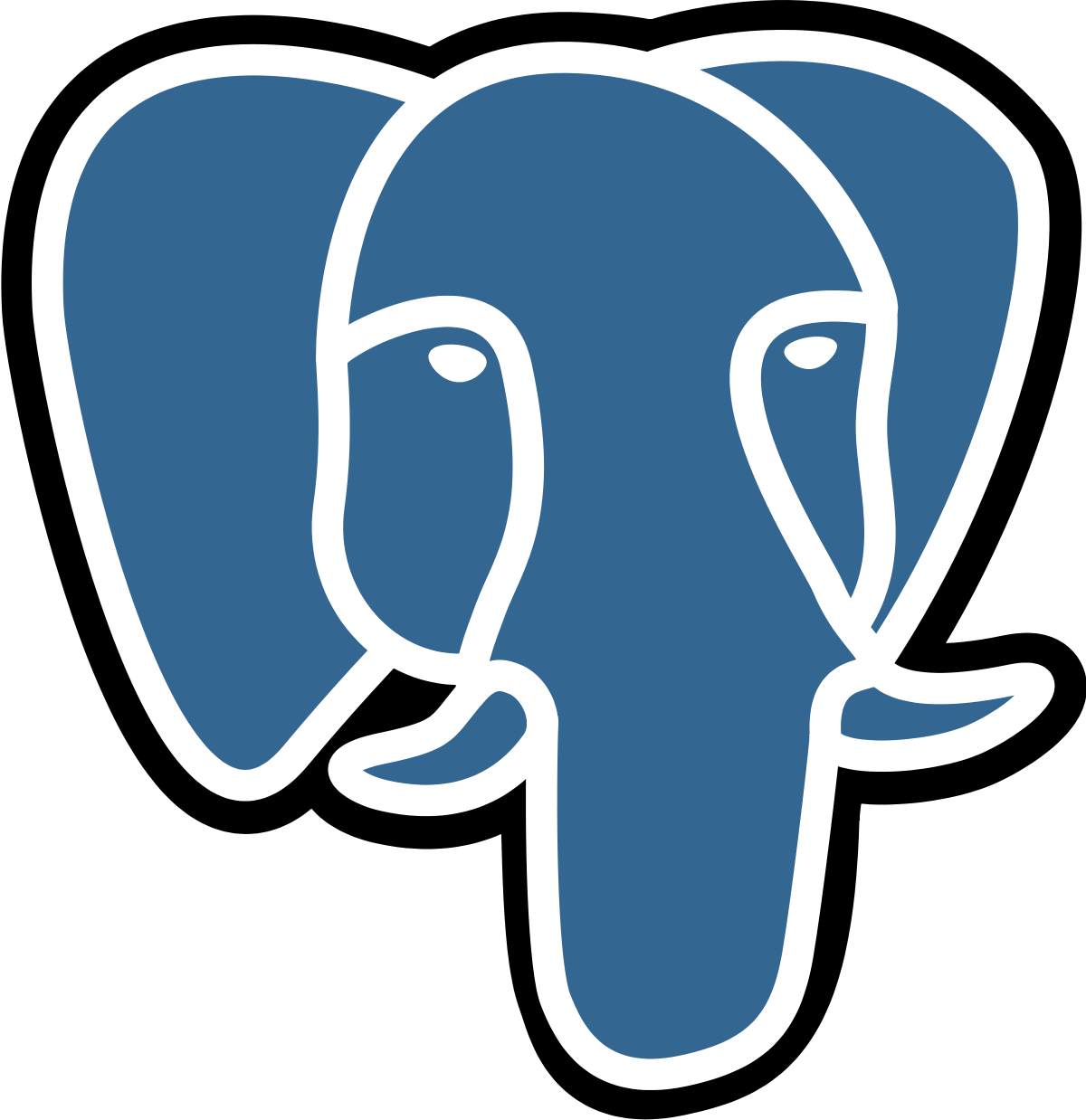
1. Use a GUI to search and return results
2. Allow, based on a set of criteria, some to get matched with a player/rider, etc.

**Test Plan Paragraph**

In order to test this, I will need to have a premade table which I can run all of the commands on. Furthermore, I will need a server that updates. To confirm this, I can print the table, create a new entry into the table, then print the table again to ensure the change. I need to complete this for all methods.

**Design Diagram:**





|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Noted from presentation:

Object modeling (Jason source)

Java persistence API

ORM object relational modeling