# Milestone 4 Documents

May 13th, 2021

Team B

Alan Bruner, Derek Grayless, David Gruninger, Aden Khan

# Table of Contents

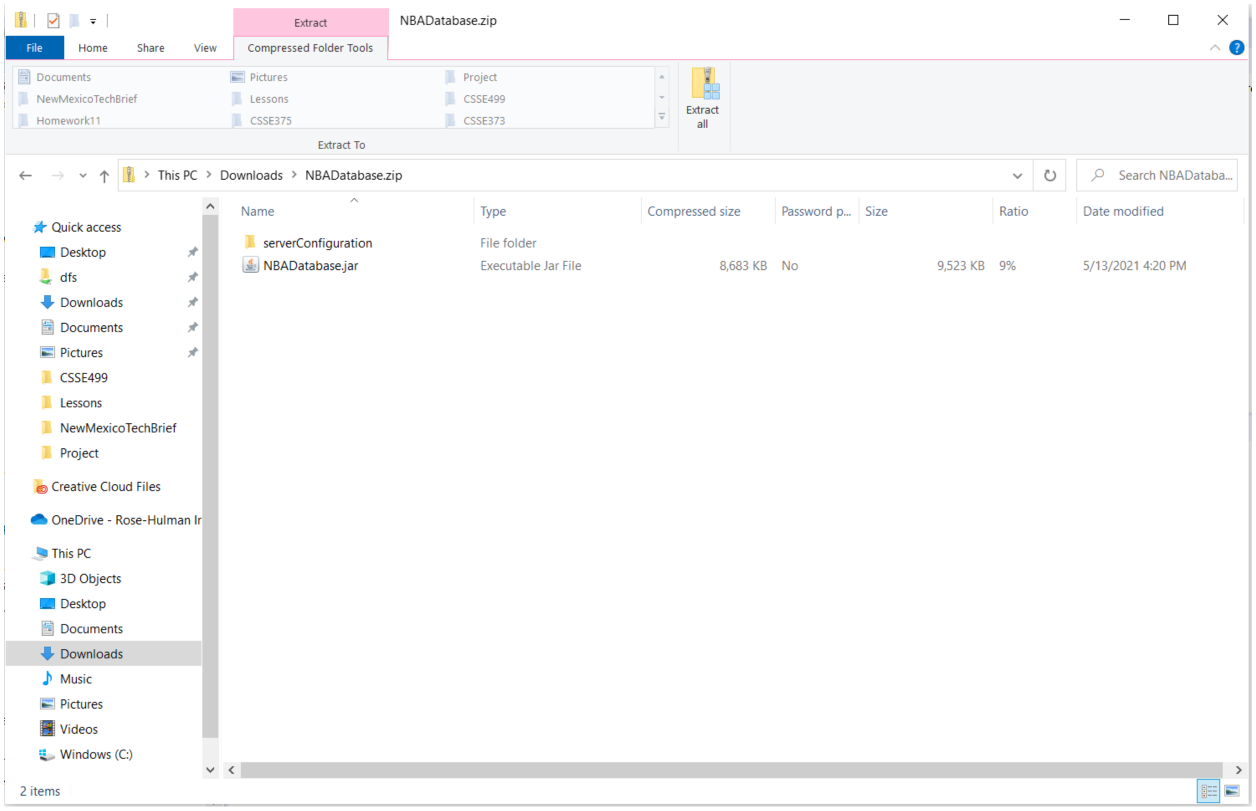
## Introduction

This system was initially developed by David Gruninger and Sam Goben between October and November 2019 for the final project for CSSE333 - Intro to Database Systems. The primary clients of the system are individuals who wish to compare player and team data for the National Basketball Association (NBA). The system allows the end users to enter the team/player of interest into a front-end UI, and the system will proceed to query an external database where the data for the players and teams are stored. The UI then displays that data back to the user. The system was written without any initial UML or design considerations, as the primary focus of the original project was the database queries and making sure the system was able to properly retrieve data. However, the UI grew over time and ended up becoming roughly 1000 lines of inefficient and poorly designed code. In an attempt to improve the system to make it more usable, testable, and maintainable, our team applied the principles we learned in CSSE375 to refactor and test the system. As a result, the system is now much easier to maintain, runs significantly faster, and has incorporated more features. The following document goes over how to use the system, the design and requirements for the system, an installation, configuration & maintenance guide, as well as an overview of our testing suite. The user guide will cover how to perform all of the common tasks within the system, as well as several common errors with the system. The installation, configuration & maintenance guide covers how to install & configure the necessary software as well as how to maintain the database being used. The testing suite will describe our 3 types of tests (Unit, Characterization, Integration), and what each category is actually testing.

# User Guide

## Downloading and Running the Project

Navigate to Moodle and download the zip titled “BasketballStatistics.zip” and extract the files somewhere on your local system. The contents of this zip show look like the image shown below:



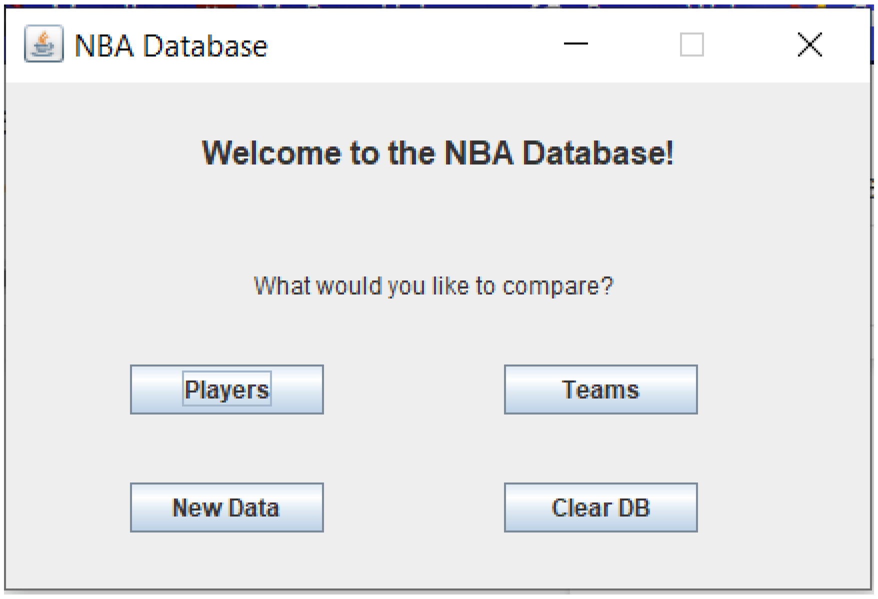
| Name | Type | Compressed size | Password p... | Size | Ratio | Date modified |
|---|---|---|---|---|---|---|
| serverConfiguration | File folder | | | | | |
| NBADatabase.jar | Executable Jar File | 8,683 KB | No | 9,523 KB | 9% | 5/13/2021 4:20 PM |

The NBADatabase.jar will be used to run the application. Please note that in order for the NBADatabase.jar file to run correctly, it must be at the same level as the serverConfiguration folder.

To run the program, double click the NBADatabase.jar file.

## Main Screen

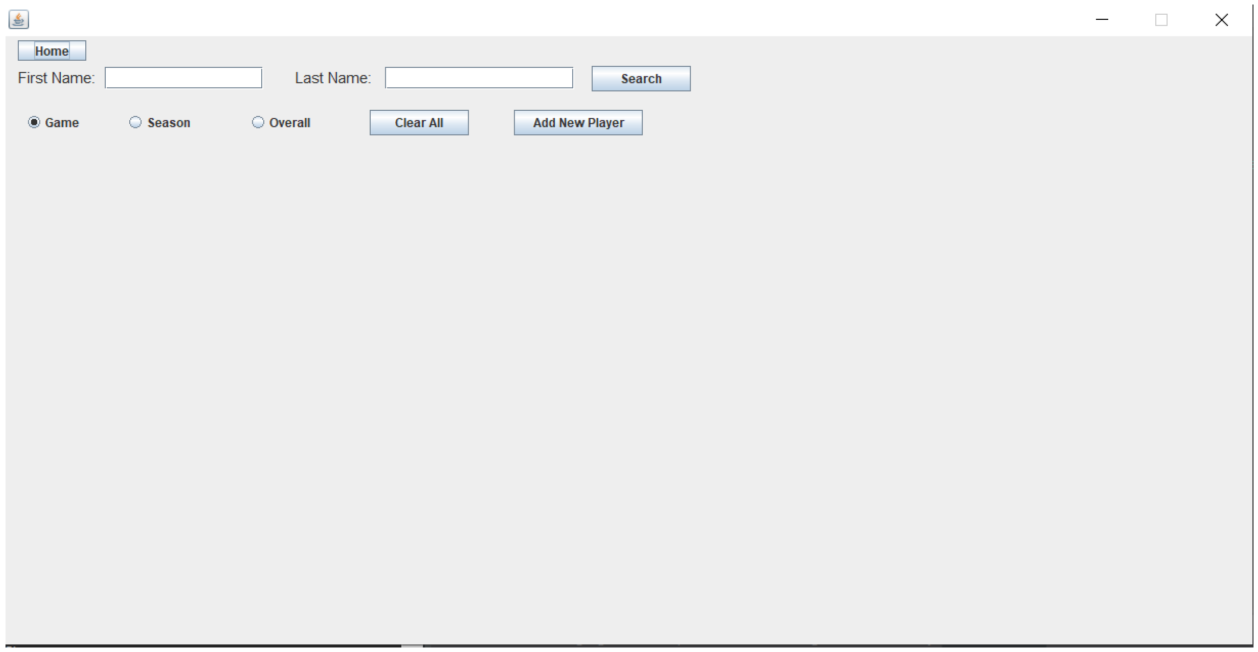
- When you start the program, it will open to this screen



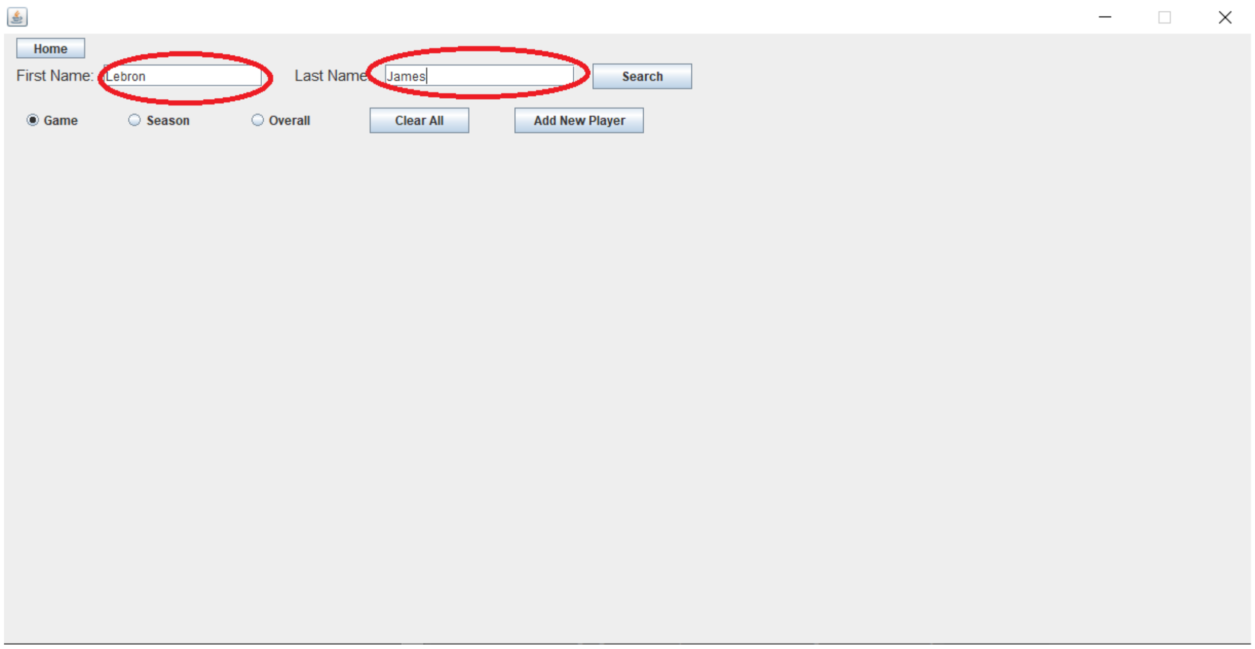
- Pressing the *Players* button will open the **Players Screen**
- Pressing the *Teams* button will open the **Teams Screen**
- Pressing the *New Data* button will open a file explorer where you can select a file to add to the database
  - Notes on how to add a file to database are shown later in this guide
- Pressing the *Clear DB* button will clear the database you are working on

## Players Screen

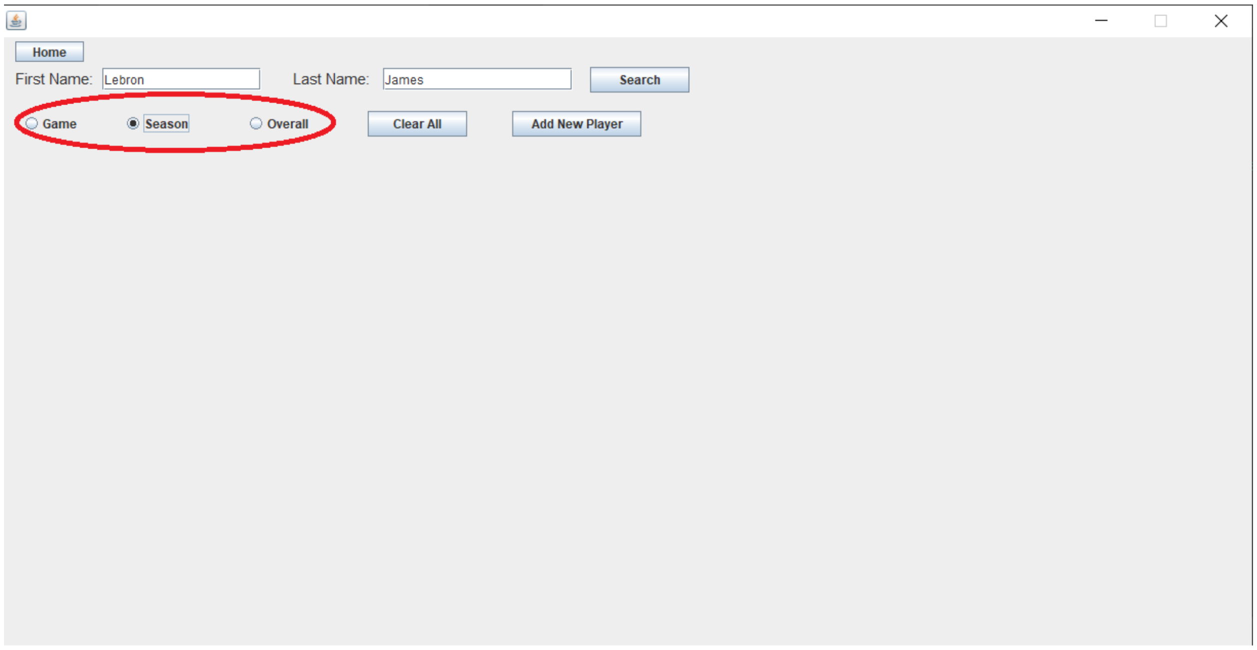
- When you go to the **Players Screen**, the program will display this screen



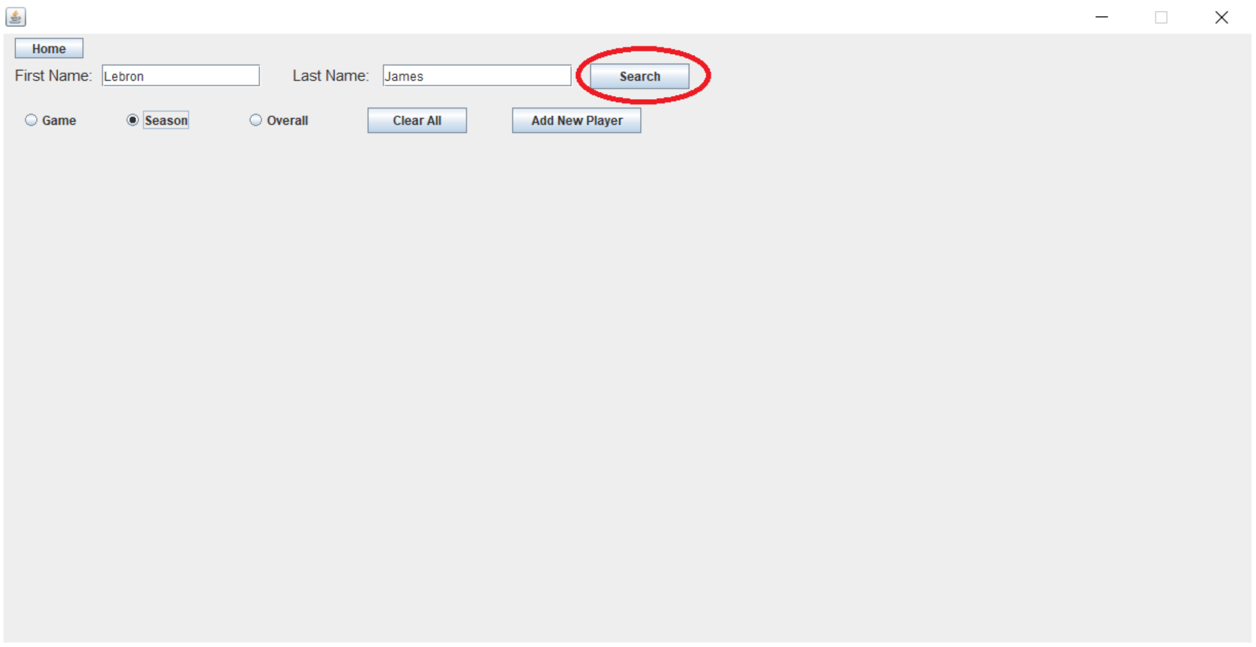
- To search for a player's information, enter their **first name** and **last name** into the two search bars (example data - First Name: Lebron, Last Name: James)



- Then, select whether you want to show data for a specific **game**, one **season**, or **overall** info for the player using the three *radial buttons*



- Once you have entered a player’s name and chosen which item you want to search for, press the *Search* button to display results


Home

First Name: Lebron

Last Name: James

Search

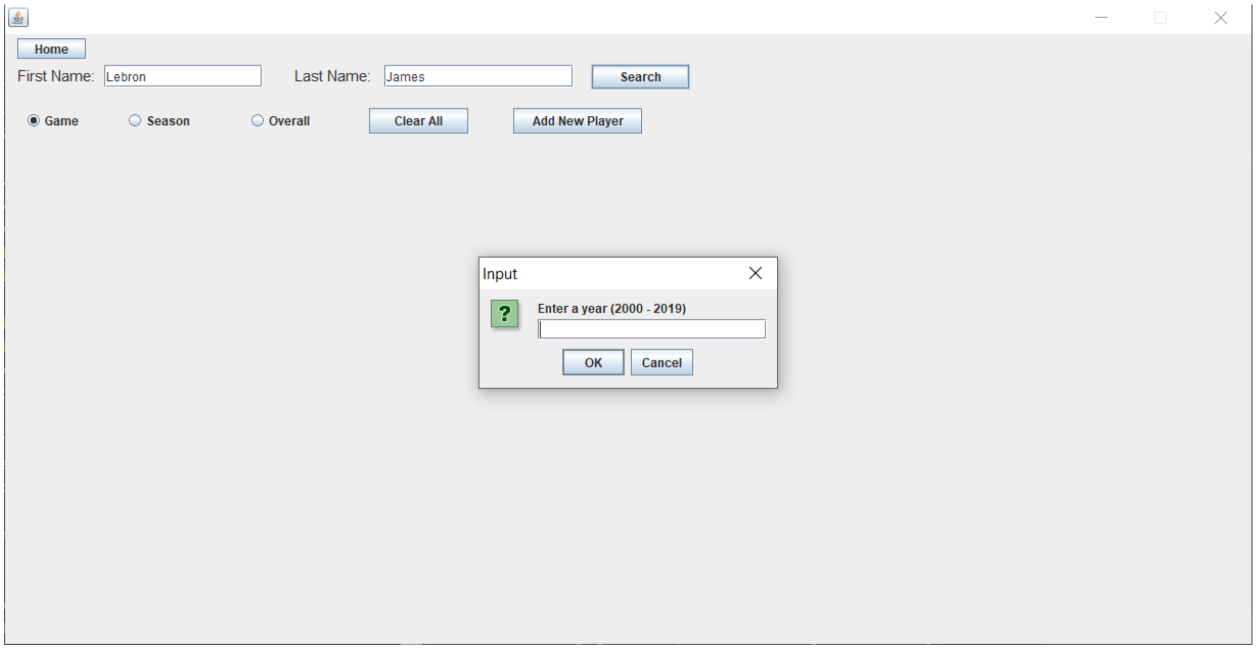
Game [✓ Season] Overall

Clear All

Add New Player

## Searching for Game

- If you selected the **Game** option, the program will display a search bar to enter the year in which the game took place. (Example year: 2003)


Home

First Name: Lebron

Last Name: James

Search

[✓ Game] Season Overall

Clear All

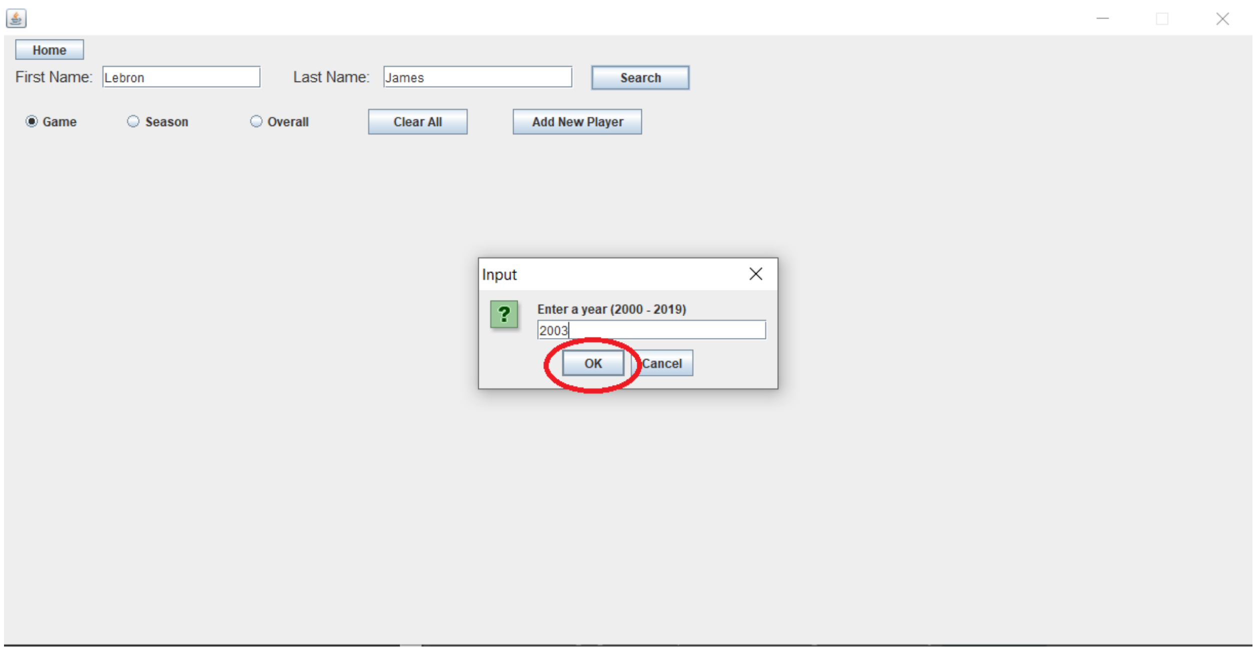
Add New Player

Input

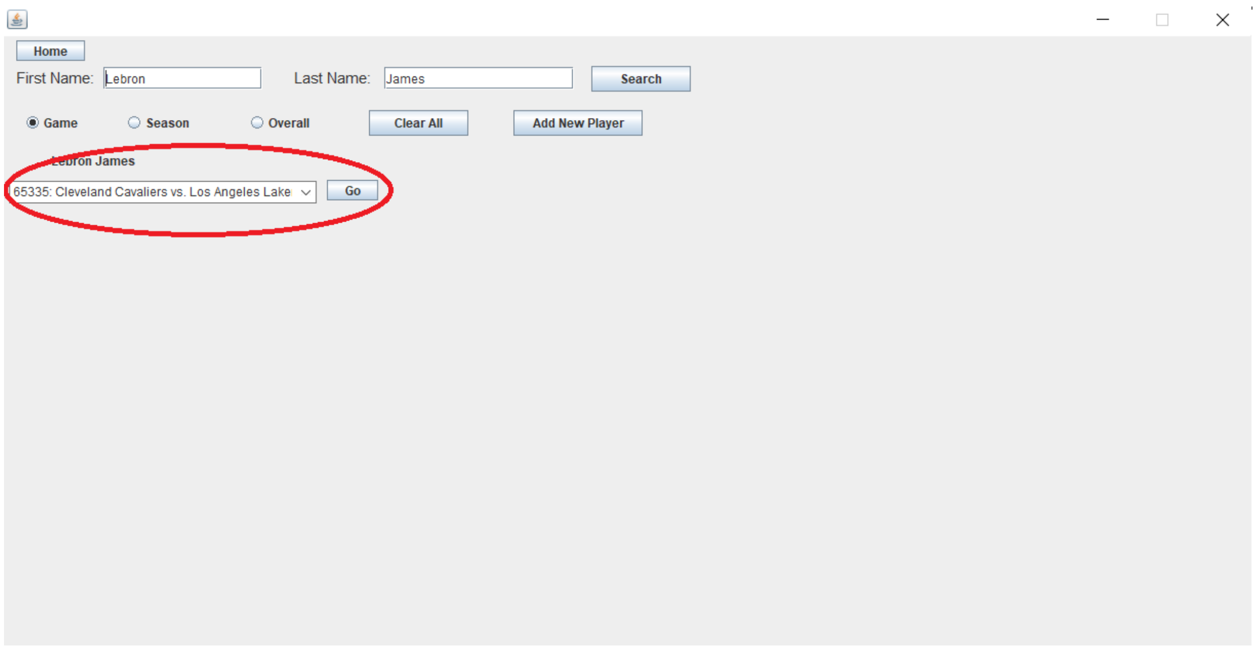
Enter a year (2000 - 2019)

OK Cancel

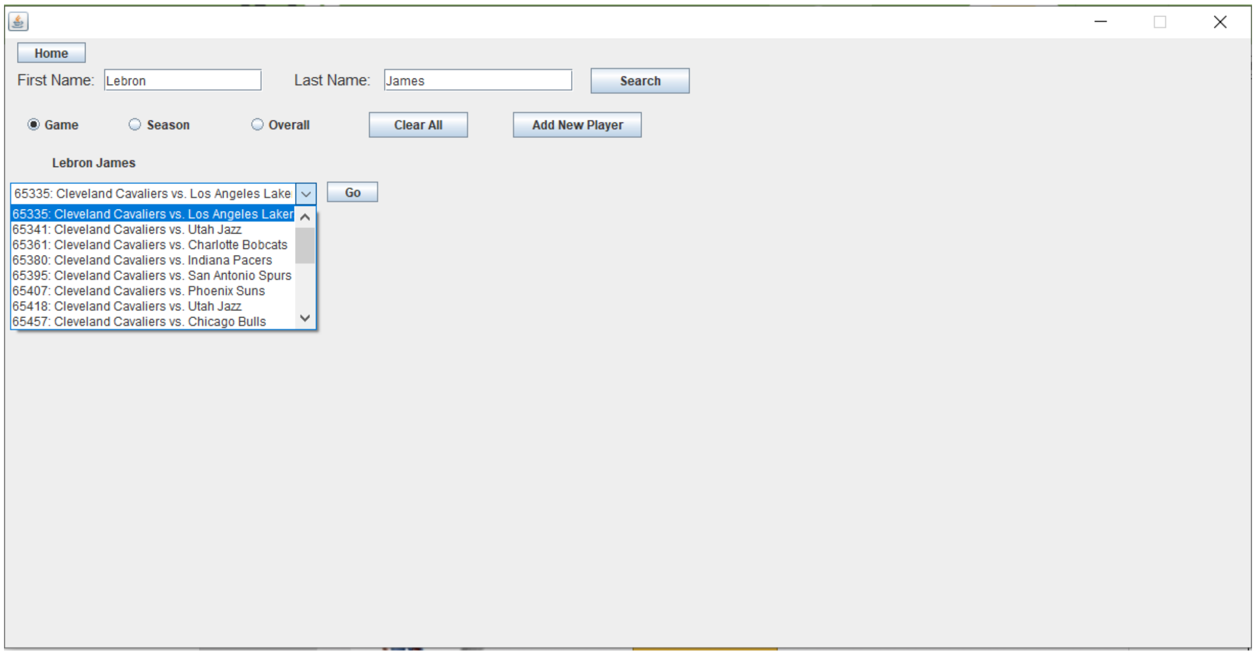
- Once you have entered the year, pres **OK**



- Then, the program will display a dropdown menu to select a game from that year



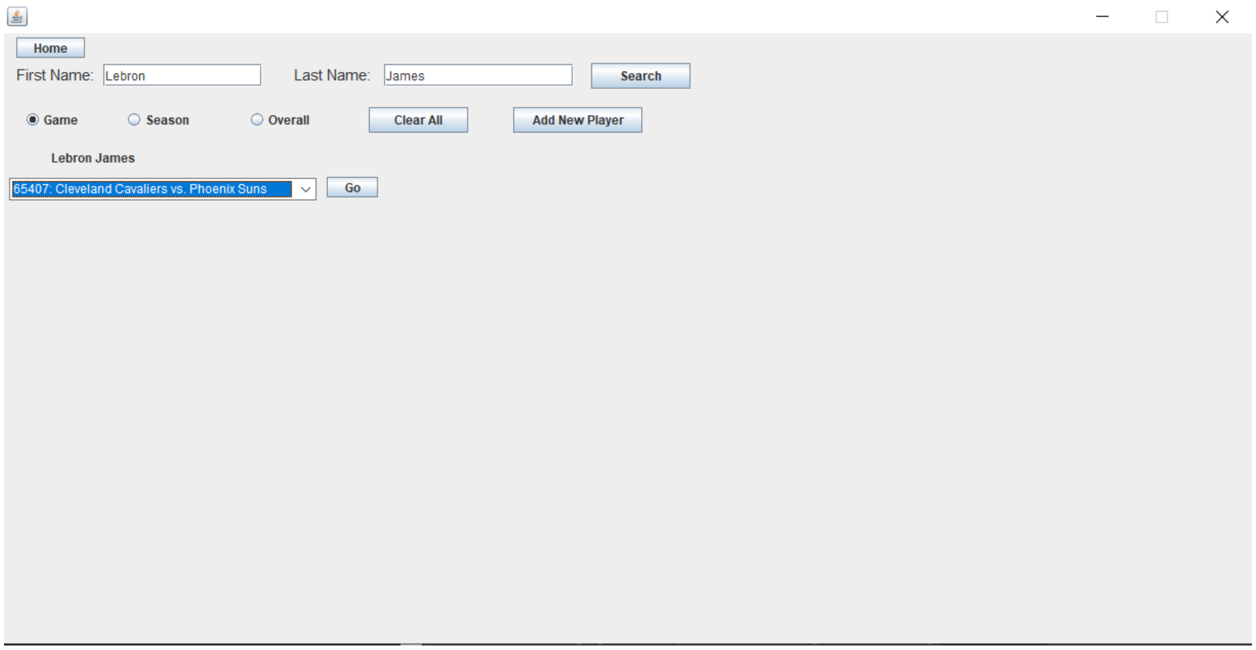
- Click on the menu to display a list of games



65335: Cleveland Cavaliers vs. Los Angeles Lake

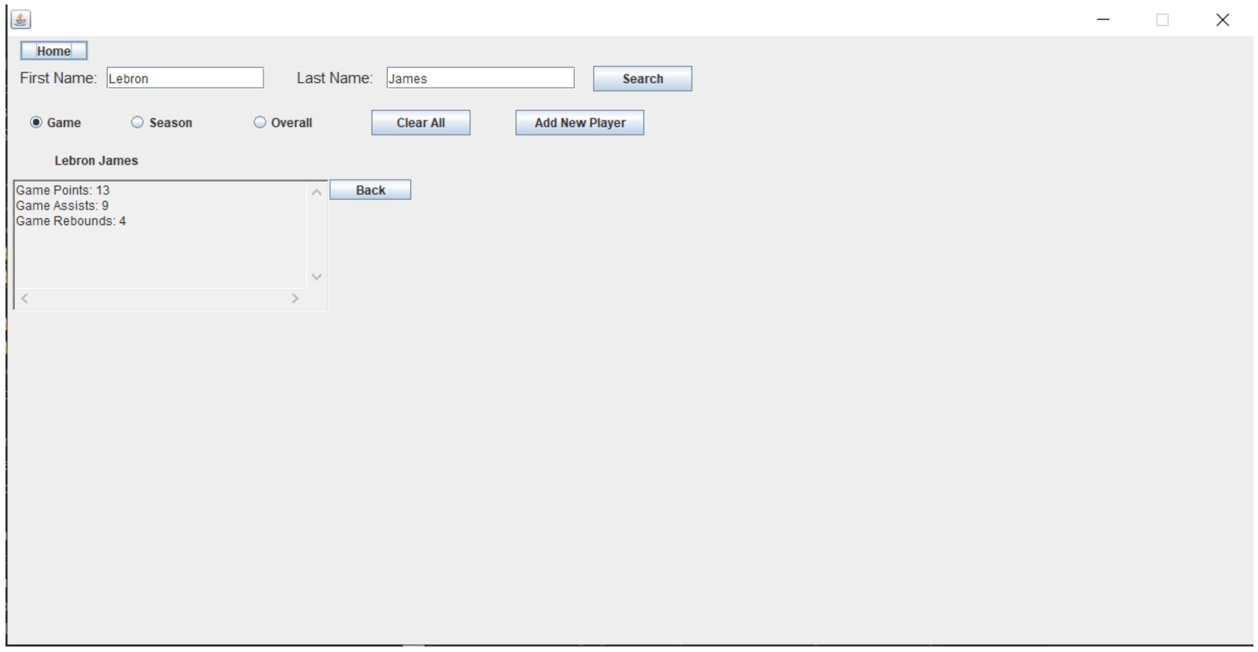
65335: Cleveland Cavaliers vs. Los Angeles Laker
65341: Cleveland Cavaliers vs. Utah Jazz
65361: Cleveland Cavaliers vs. Charlotte Bobcats
65380: Cleveland Cavaliers vs. Indiana Pacers
65395: Cleveland Cavaliers vs. San Antonio Spurs
65407: Cleveland Cavaliers vs. Phoenix Suns
65418: Cleveland Cavaliers vs. Utah Jazz
65457: Cleveland Cavaliers vs. Chicago Bulls

- Once you have selected a game, press the *Go* button



65407: Cleveland Cavaliers vs. Phoenix Suns

- The program will then display information about that player's performance in that specific game

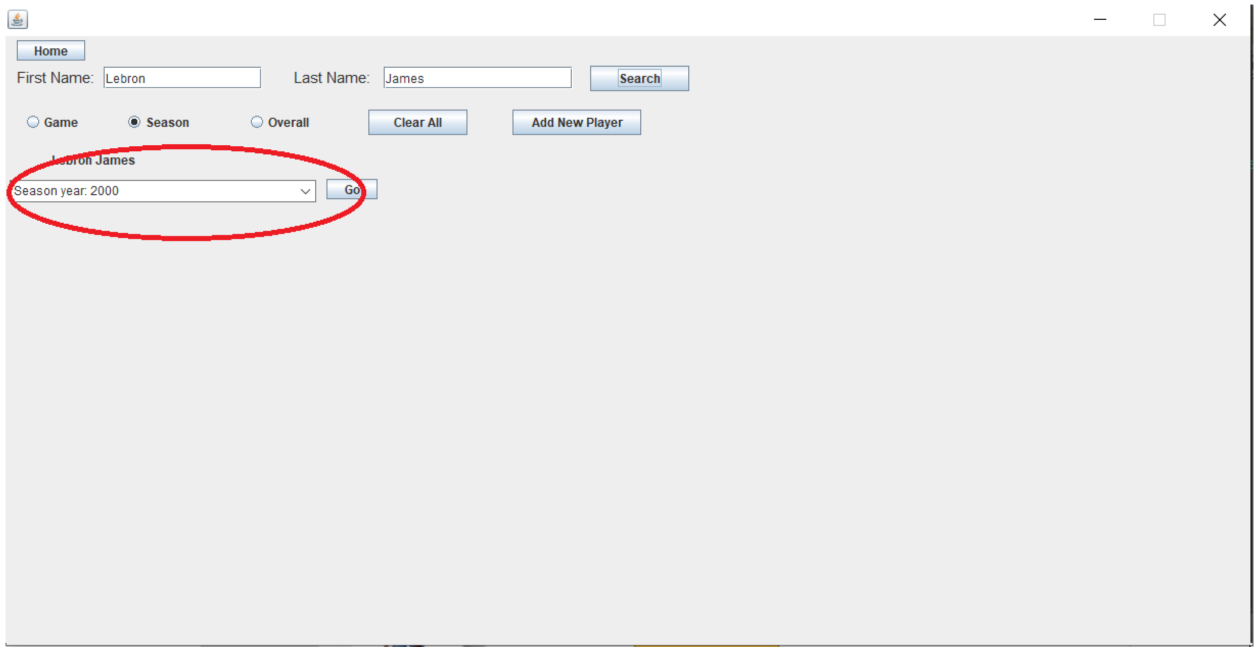


Lebron James

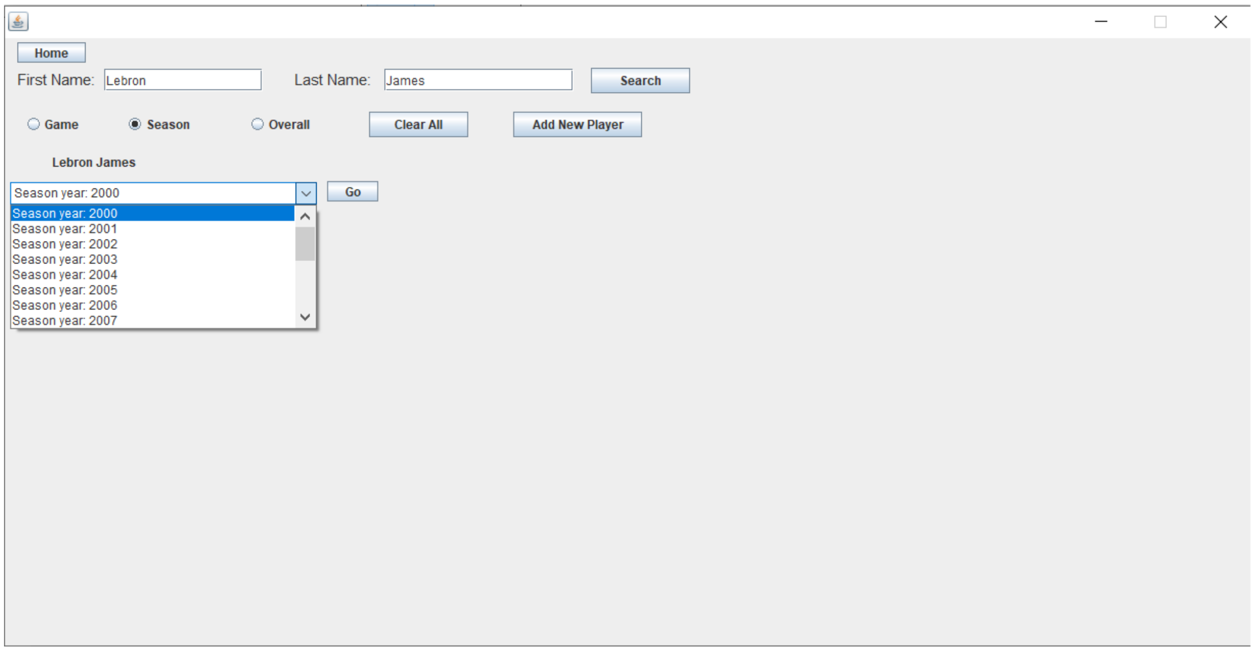
```
Game Points: 13
Game Assists: 9
Game Rebounds: 4
```

## Searching for Season

- If you selected the **Season** option, the program will display a dropdown menu to select a season year



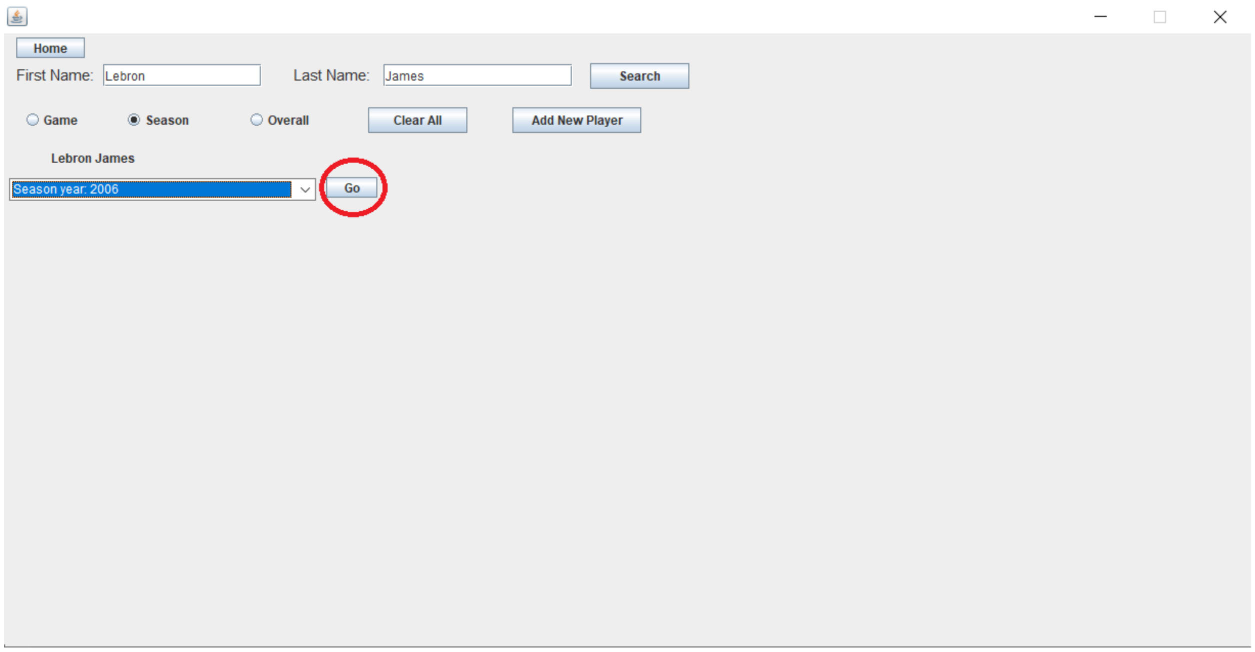
- Click on the menu to open a list of years



Lebron James

Season year: 2000
Season year: 2000
Season year: 2001
Season year: 2002
Season year: 2003
Season year: 2004
Season year: 2005
Season year: 2006
Season year: 2007

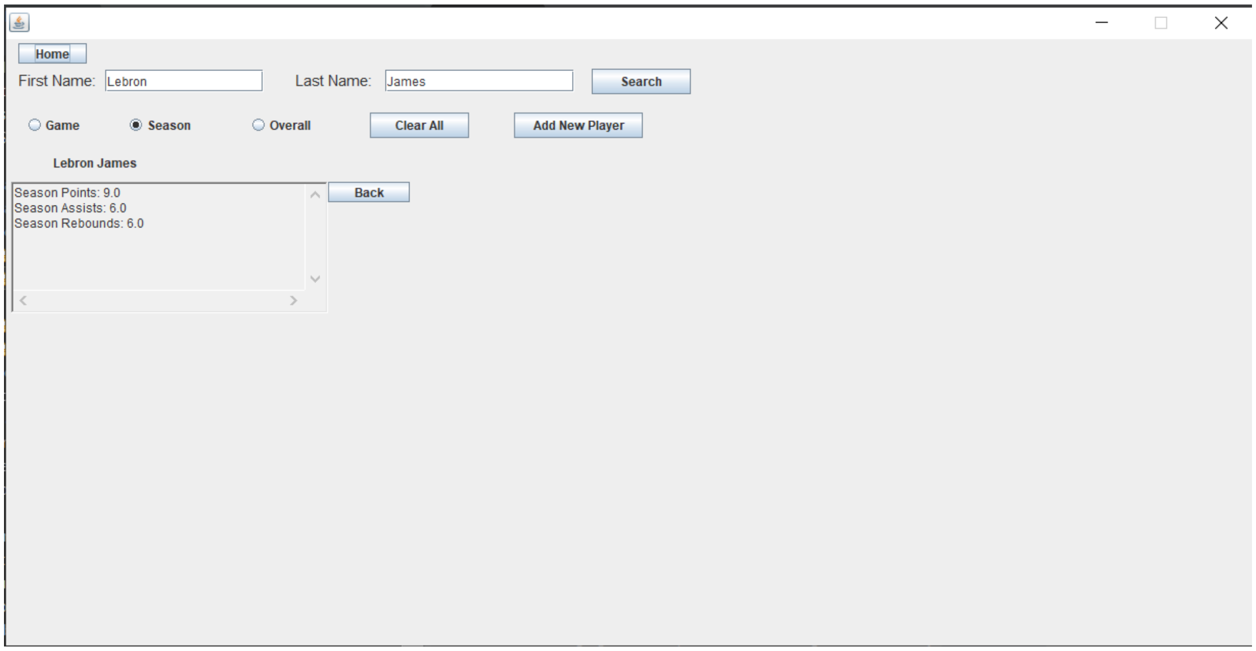
- Once you have selected a year, press the *Go* button



Lebron James

Season year: 2006

- The program will then display information about the player's average performance in that specific season



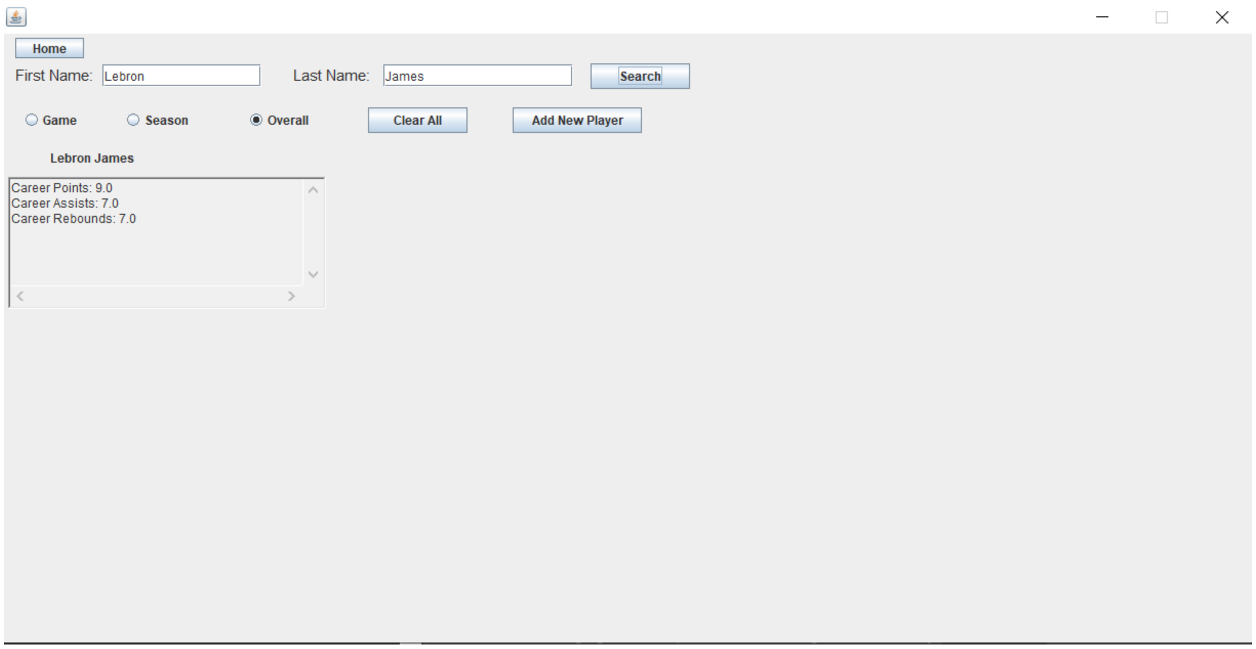
First Name: Lebron  Last Name: James
[✓ Season]

Lebron James

```
Season Points: 9.0
Season Assists: 6.0
Season Rebounds: 6.0
```

## Searching for Overall

- If you selected the **Overall** option, the program will display information about the player's average performance across their entire career in the NBA



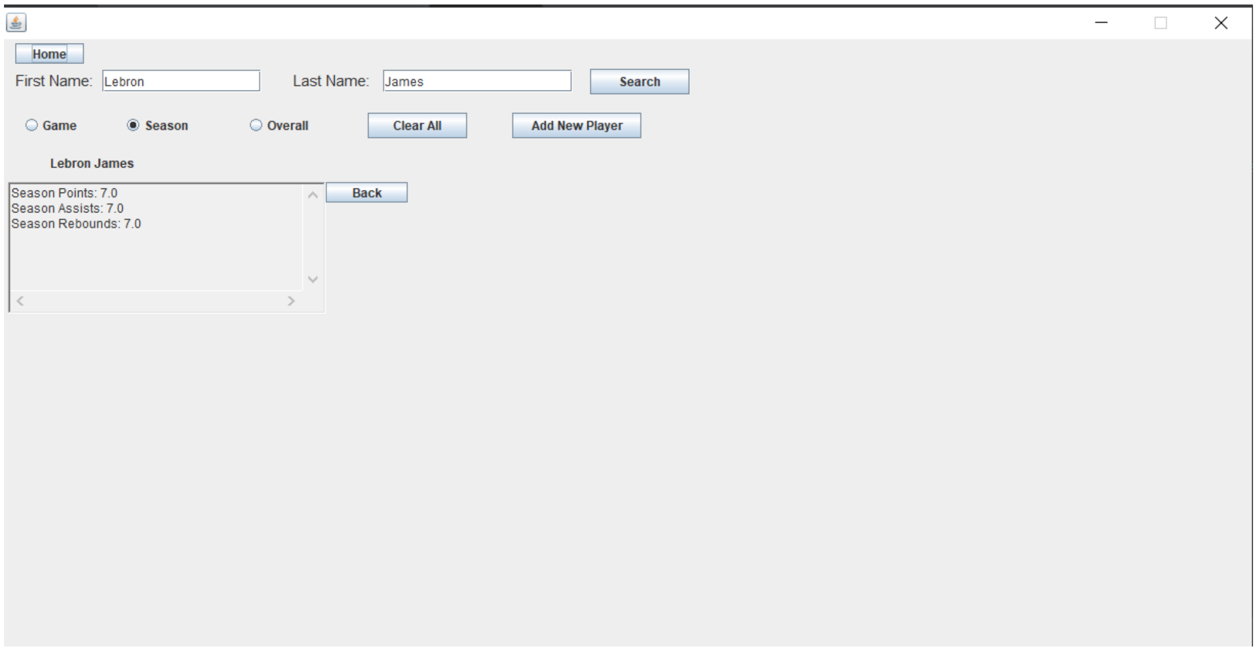
First Name: Lebron  Last Name: James
[✓ Overall]

Lebron James

```
Career Points: 9.0
Career Assists: 7.0
Career Rebounds: 7.0
```

## Display Multiple Search Items

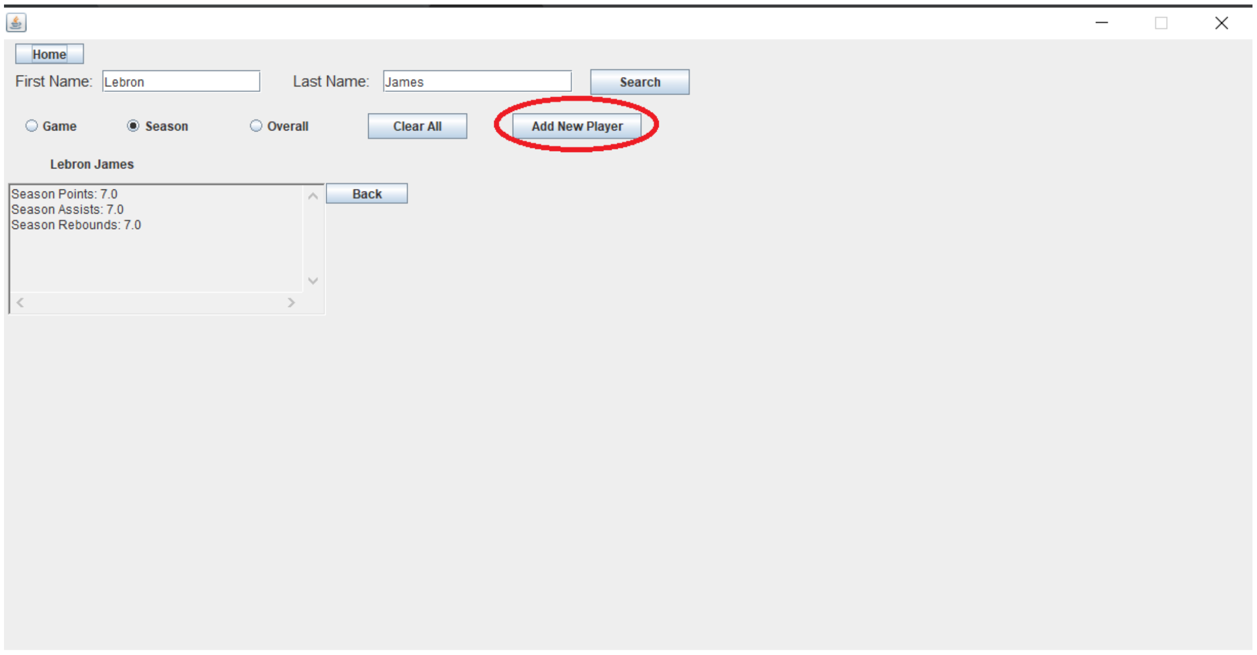
- If you want to display multiple search items at once, search for the first player's information as normal



Lebron James

```
Season Points: 7.0
Season Assists: 7.0
Season Rebounds: 7.0
```

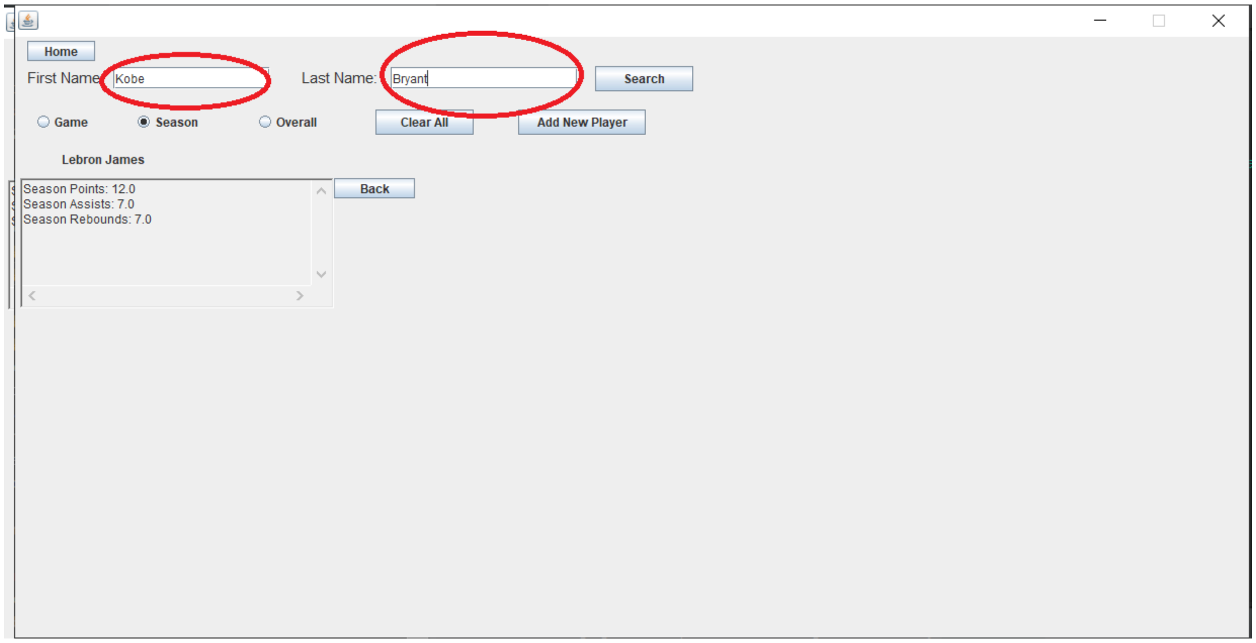
- Then, press the *Add new Player* button



Lebron James

```
Season Points: 7.0
Season Assists: 7.0
Season Rebounds: 7.0
```

- Then, enter the search information for the second player (or search for the same player again to compare their performance with a different game or season)

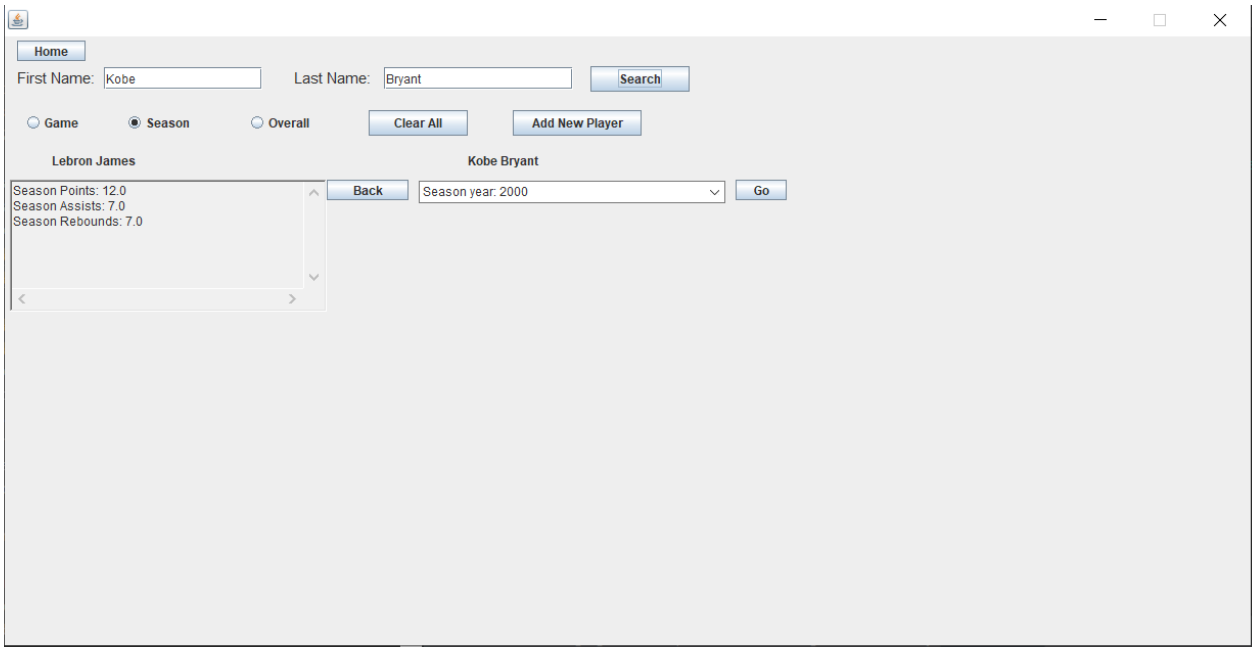


First Name: Kobe
Last Name: Bryant
[✓ Season]

Lebron James

```
Season Points: 12.0
Season Assists: 7.0
Season Rebounds: 7.0
```

- When you press Search, the second player will appear next to the first, and you can continue searching as normal



First Name: Kobe
Last Name: Bryant
[✓ Season]

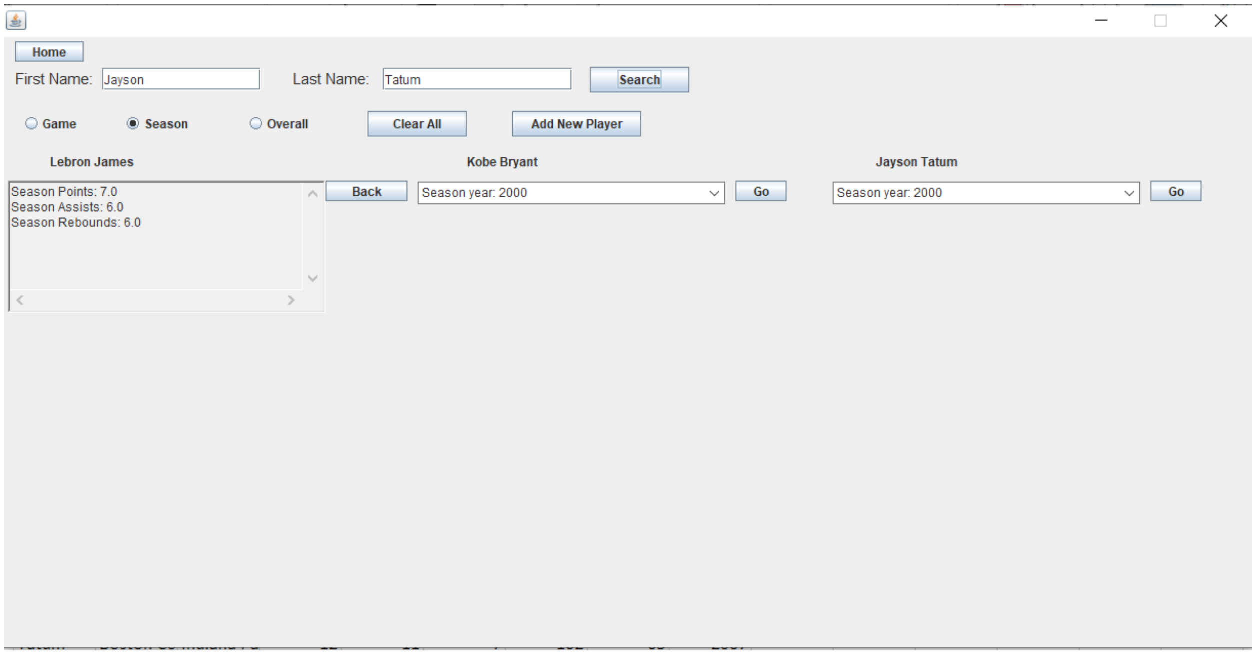
Lebron James

```
Season Points: 12.0
Season Assists: 7.0
Season Rebounds: 7.0
```

Kobe Bryant

Season year: 2000

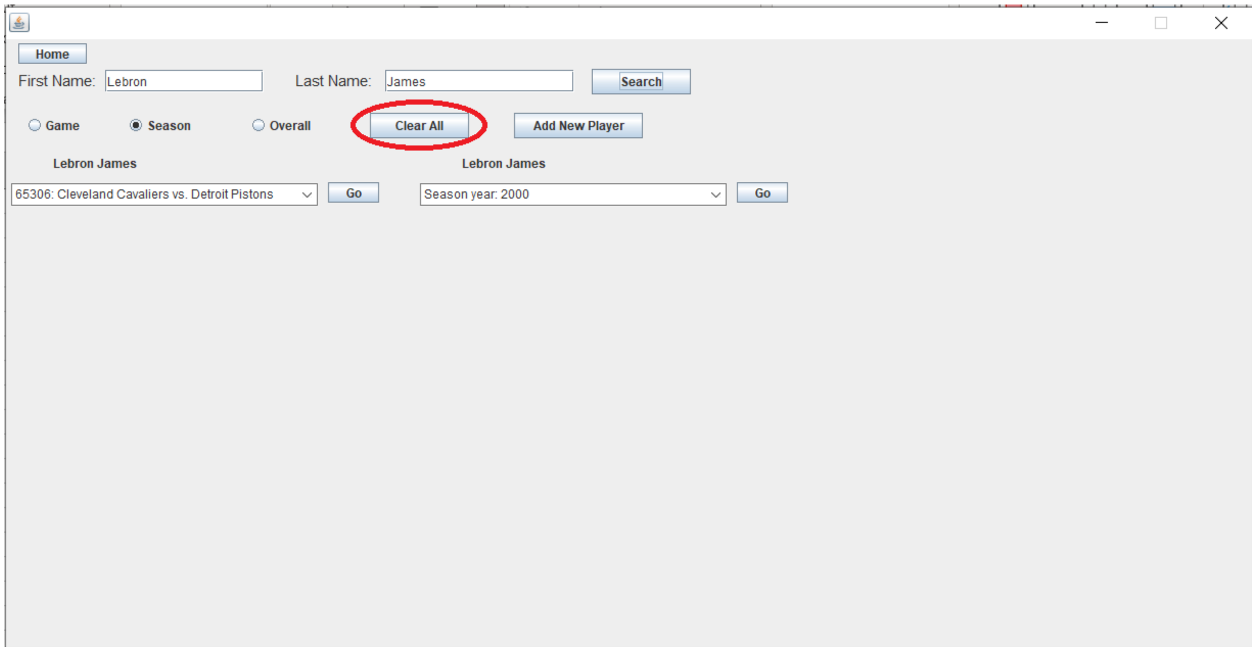
- You can search for up to three players at a time. Example data is shown below



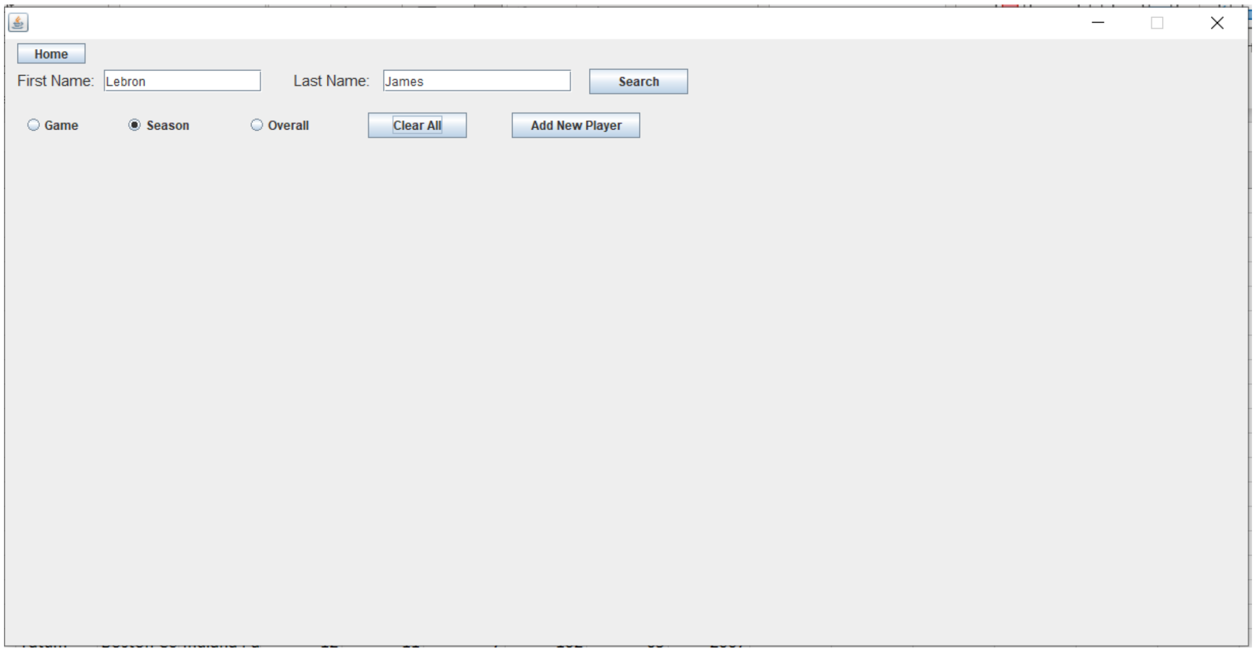
```
Season Points: 7.0
Season Assists: 6.0
Season Rebounds: 6.0
```

## Clear Data

- If you have multiple players displayed and you would like to remove them from the display, press the *Clear All* button



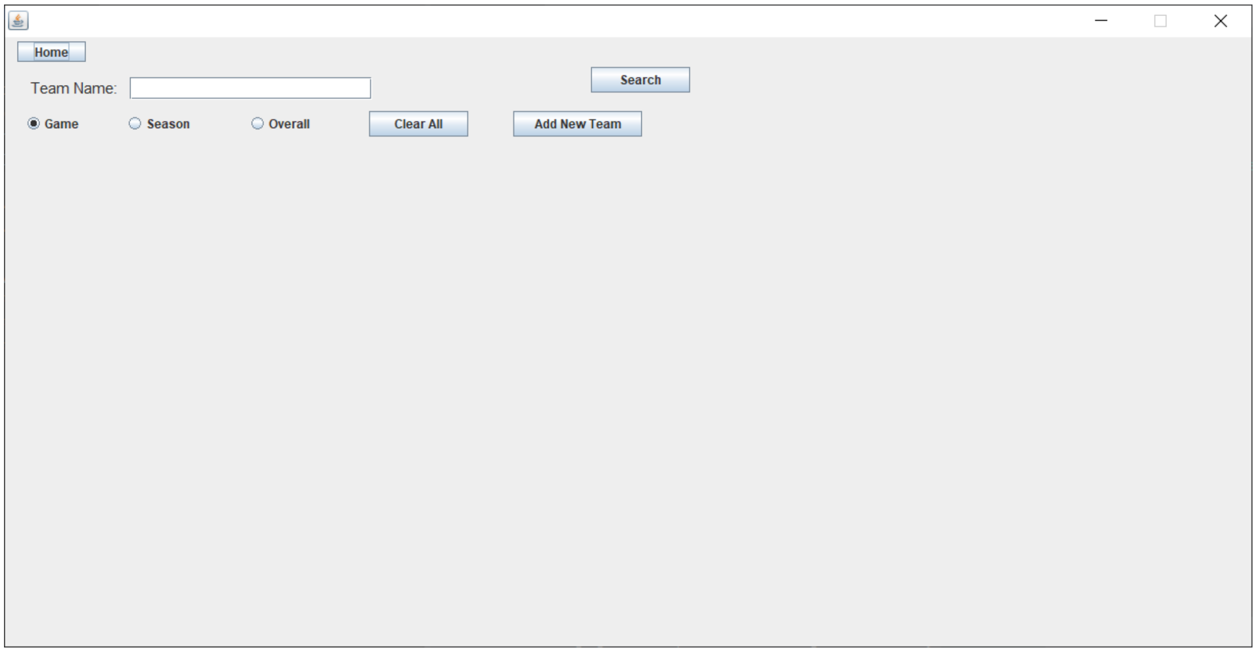
- The program will then return to a blank **Player Screen**



First Name: Lebron
Last Name: James
( ) Game (●) Season ( ) Overall

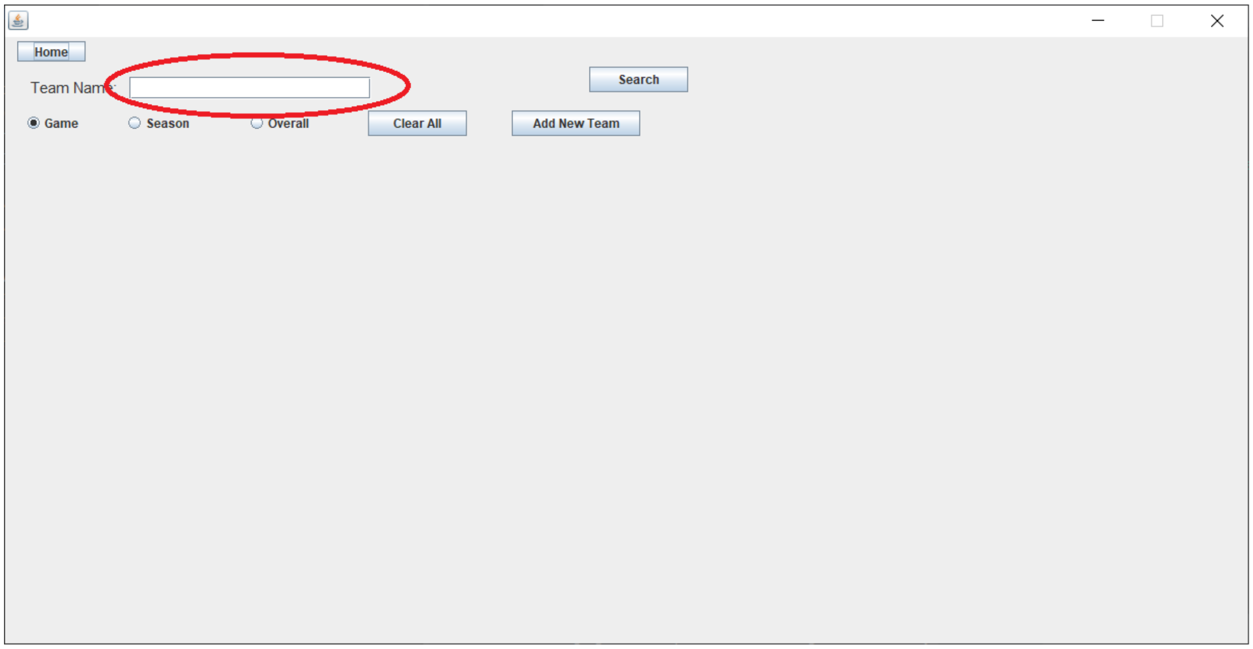
## Team Screen

- When you open the **Team Screen**, the program will display this window



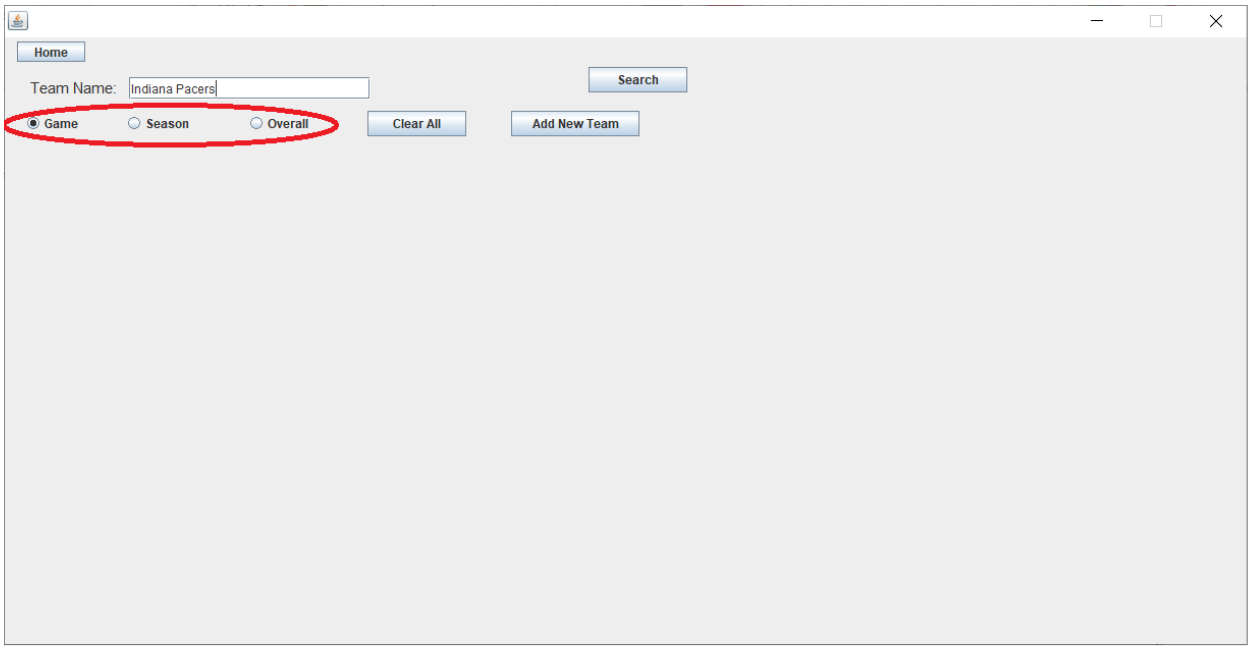
(●) Game ( ) Season ( ) Overall

- To Search for a team, enter the **Team Name** into this text box



- Note: When entering a team name, enter the full name (e.g "Chicago Bulls" instead of "Bulls")

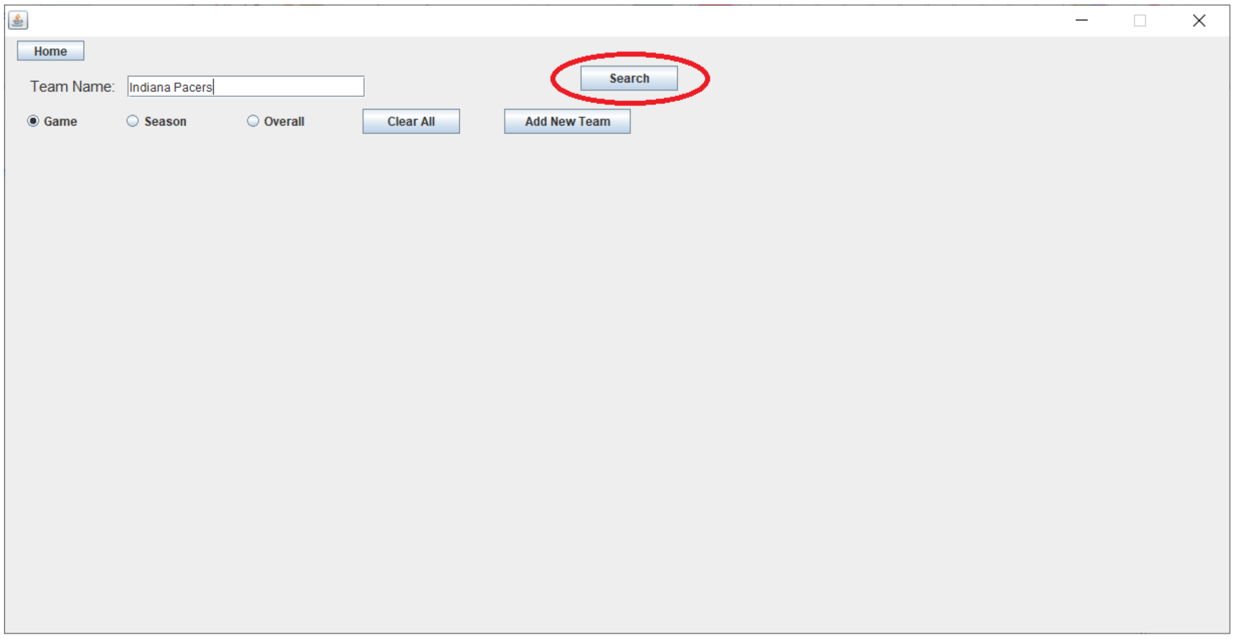
- Then, select whether you want to search for **Game**, **Season**, or **Overall** info using the three *Radial Buttons*



Team Name: Indiana Pacers

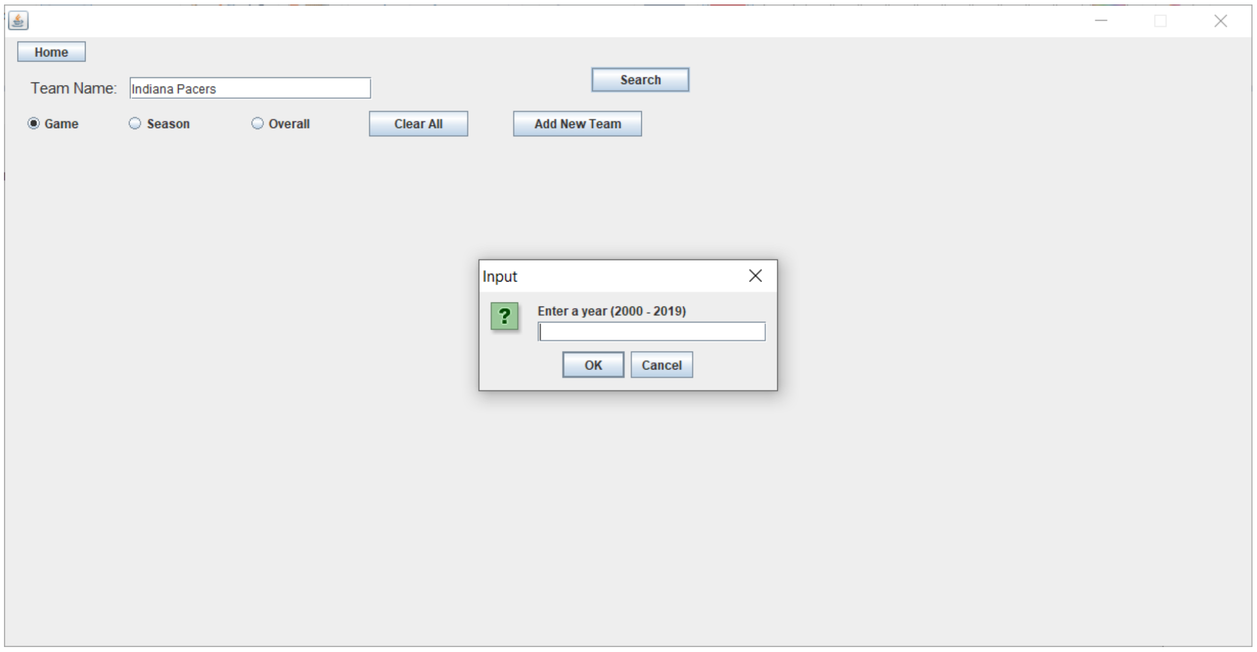
[✓ Game] Season Overall

- Once you have entered the team's name and selected the appropriate *Radial Button*, press the **Search** button

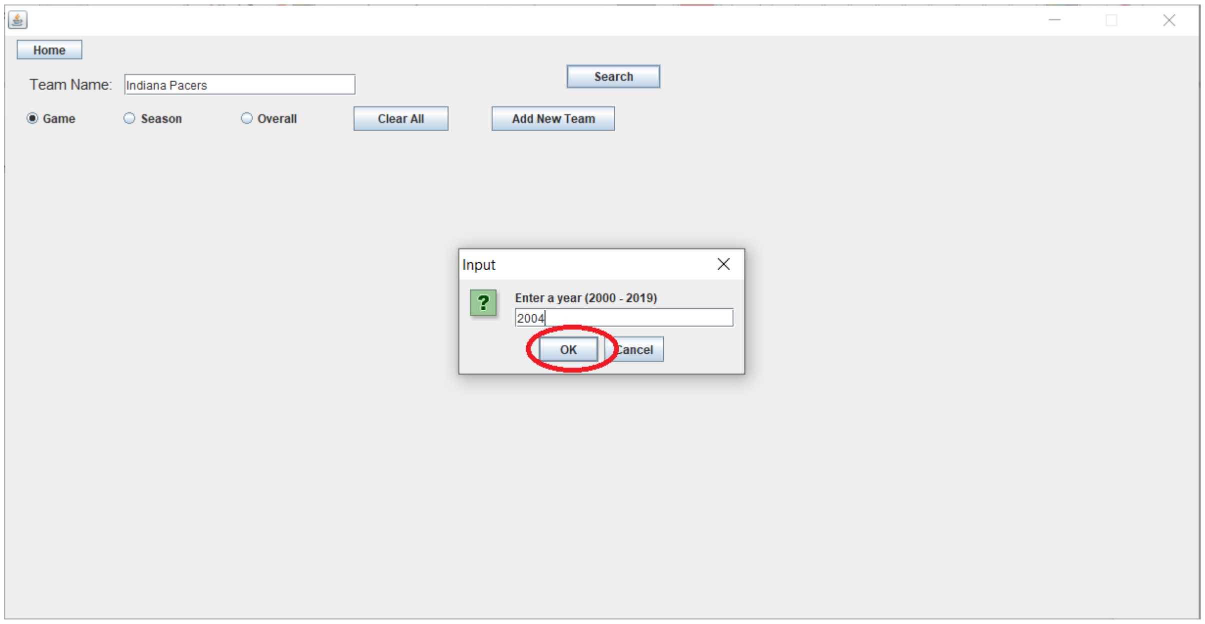


## Searching for Game

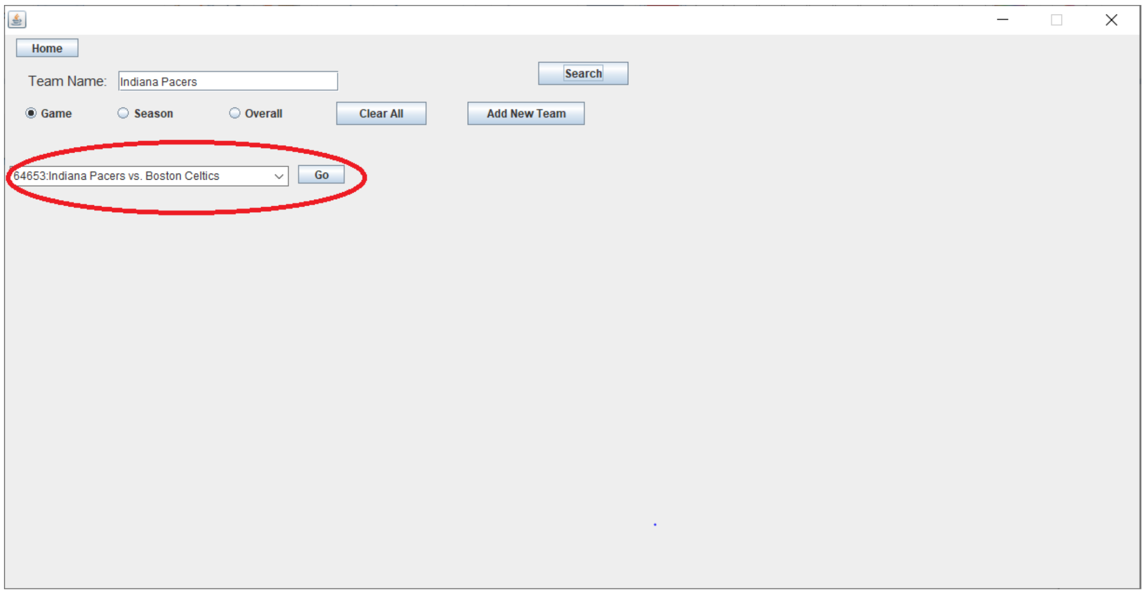
- If you selected the **Game** option, the program will display a search bar to enter the year in which the game took place (Example Year: 2004)



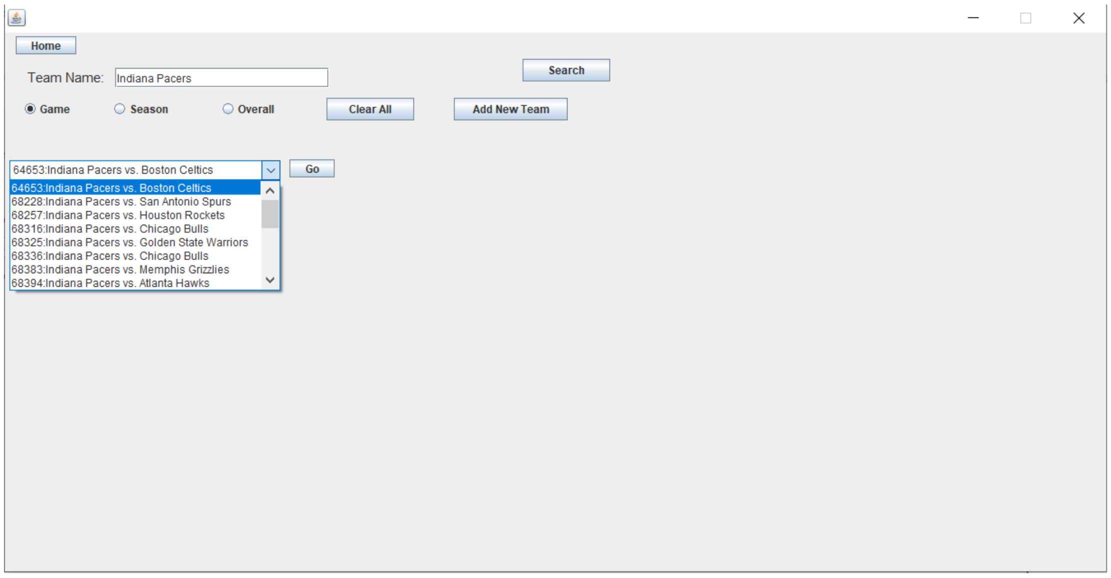
- Once you have entered the year, pres **OK**



- Then, the program will display a dropdown menu to select a game from that year


Home
Team Name: Indiana Pacers
Search
Game
Season
Overall
Clear All
Add New Team
64653:Indiana Pacers vs. Boston Celtics
Go

- Click on the menu to display a list of games



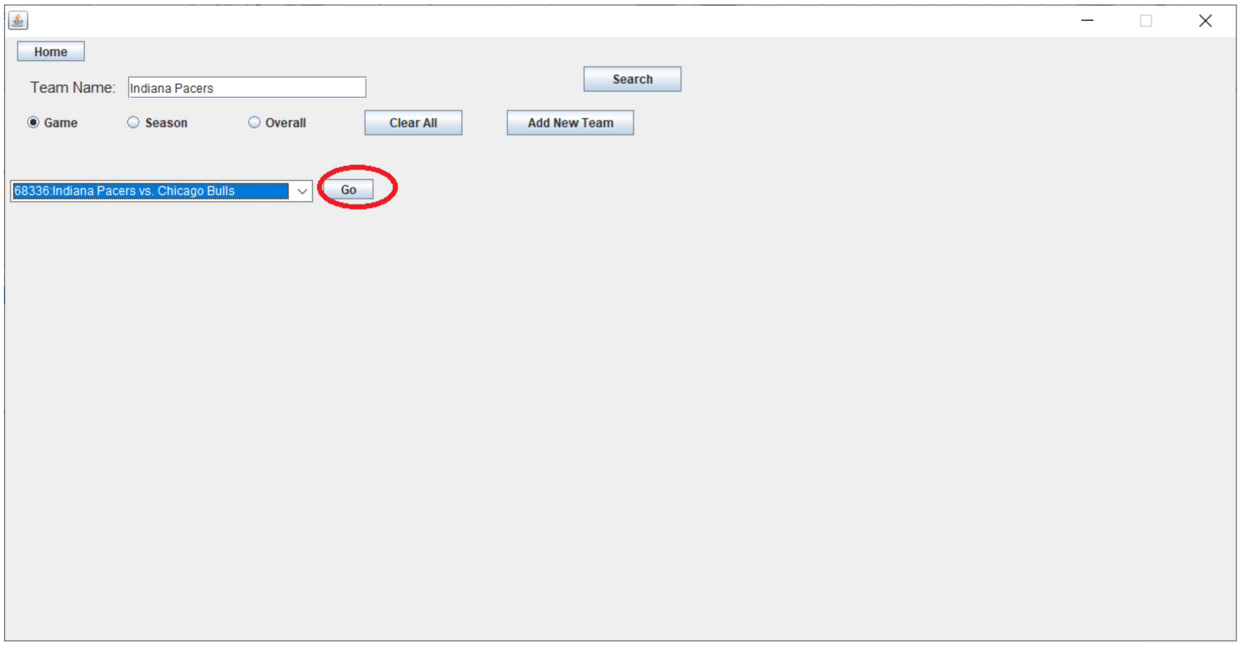
Team Name: Indiana Pacers

(● Game) Season Overall

64653:Indiana Pacers vs. Boston Celtics

```
64653:Indiana Pacers vs. Boston Celtics
68228:Indiana Pacers vs. San Antonio Spurs
68257:Indiana Pacers vs. Houston Rockets
68316:Indiana Pacers vs. Chicago Bulls
68325:Indiana Pacers vs. Golden State Warriors
68336:Indiana Pacers vs. Chicago Bulls
68383:Indiana Pacers vs. Memphis Grizzlies
68394:Indiana Pacers vs. Atlanta Hawks
```

- Once you have selected a game, press the *Go* button

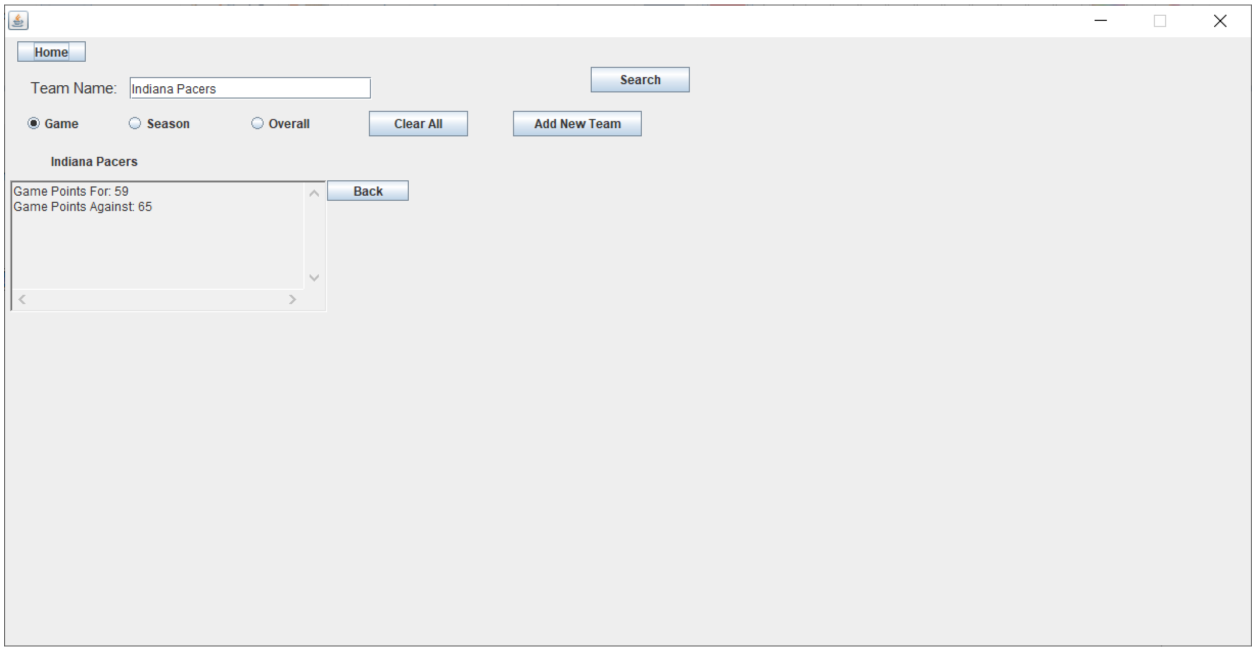


Team Name: Indiana Pacers

(● Game) Season Overall

68336:Indiana Pacers vs. Chicago Bulls

- The program will then display information about that team’s performance in that game



Team Name: Indiana Pacers

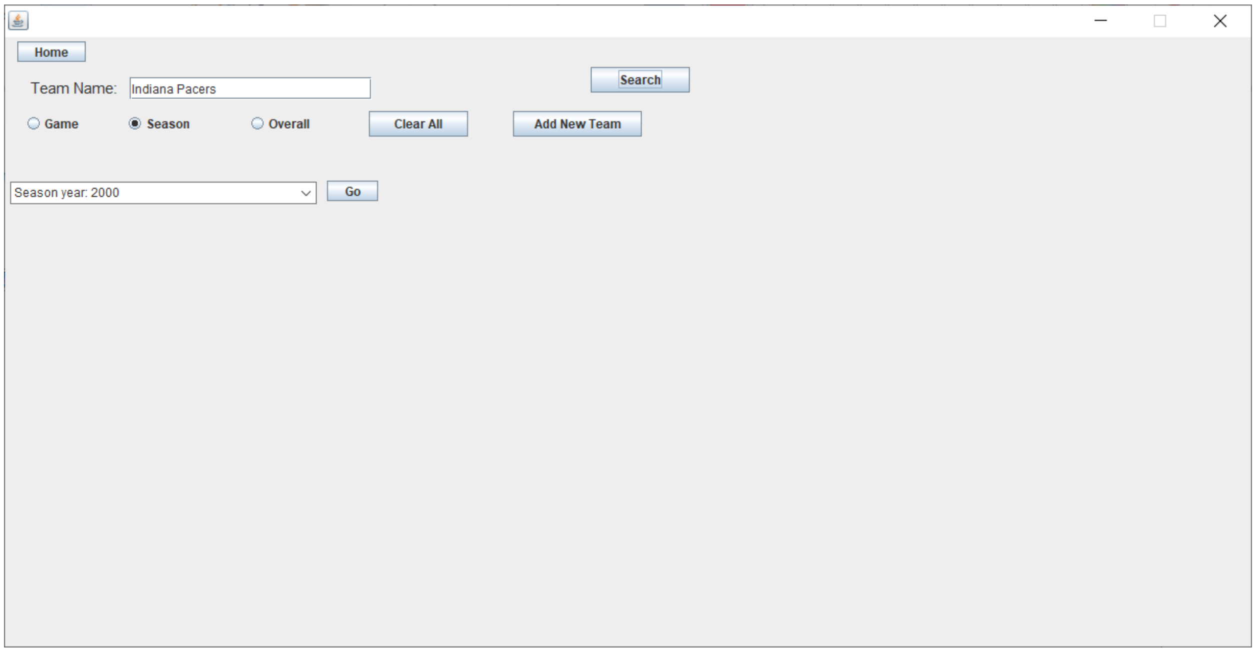
[✓ Game] Season Overall

Indiana Pacers

```
Game Points For: 59
Game Points Against: 65
```

## Searching for Season

- If you selected the **Season** option, the program will display a dropdown menu to select a season year

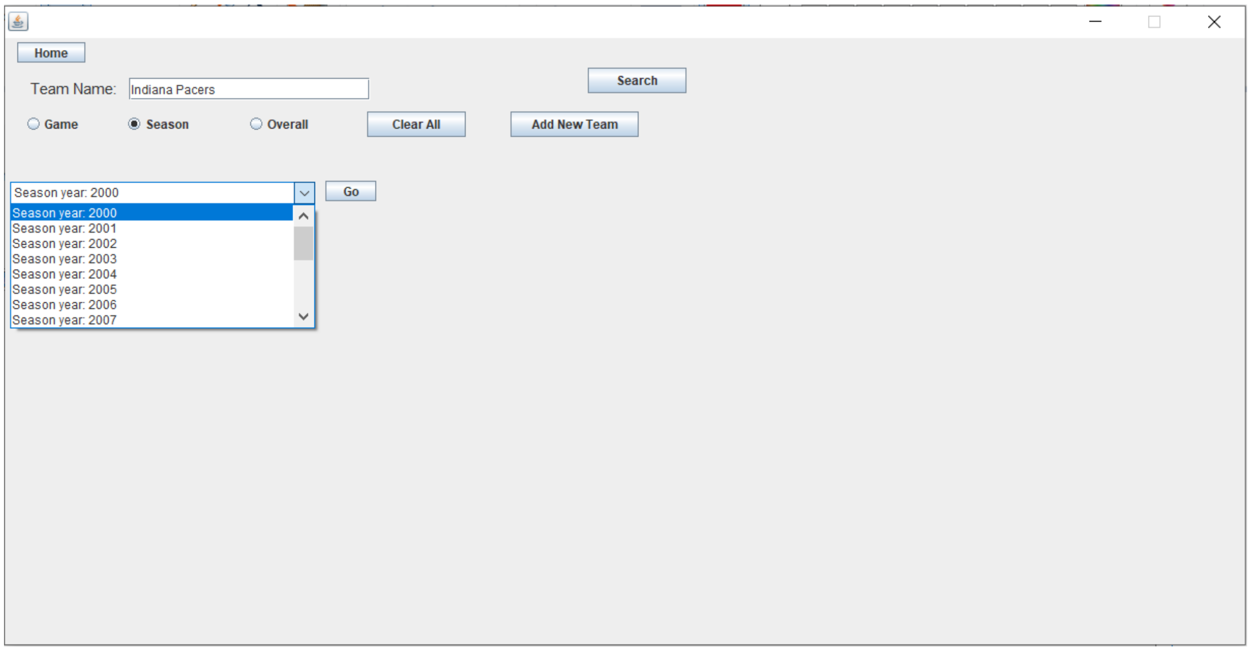


Team Name: Indiana Pacers

Game [✓ Season] Overall

Season year: 2000

- Click on the menu to open a list of years

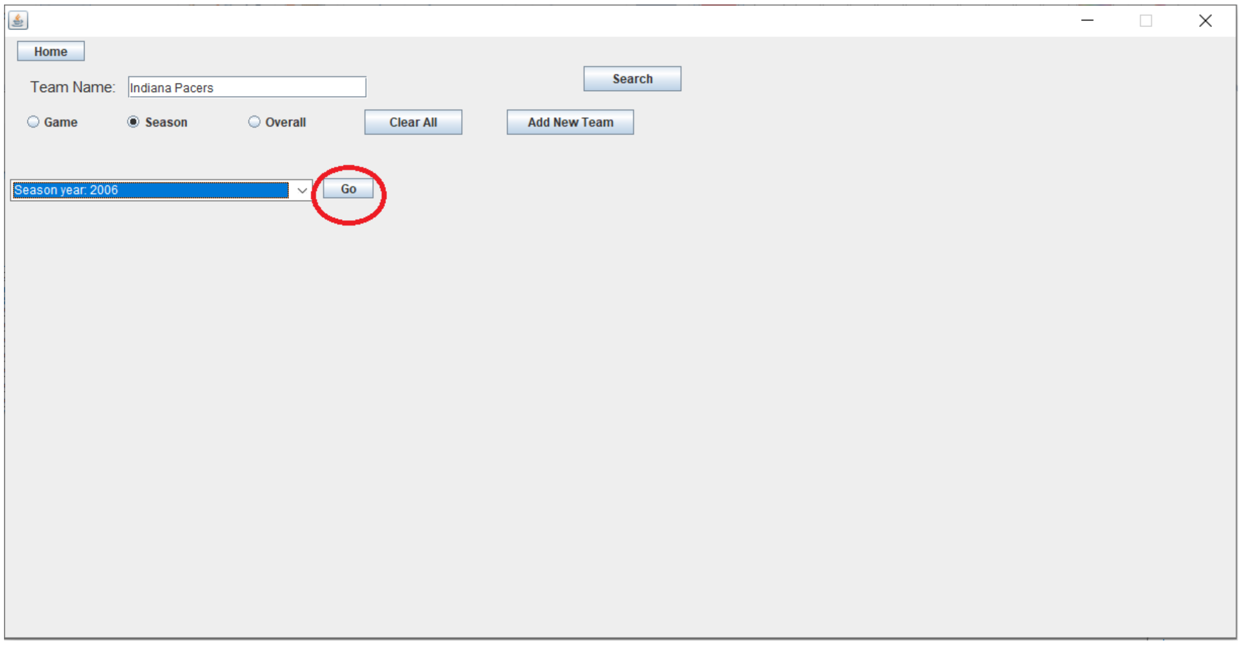


Team Name: Indiana Pacers

Season year: 2000

Season year: 2000
Season year: 2001
Season year: 2002
Season year: 2003
Season year: 2004
Season year: 2005
Season year: 2006
Season year: 2007

- Once you have selected a year, press the *Go* button



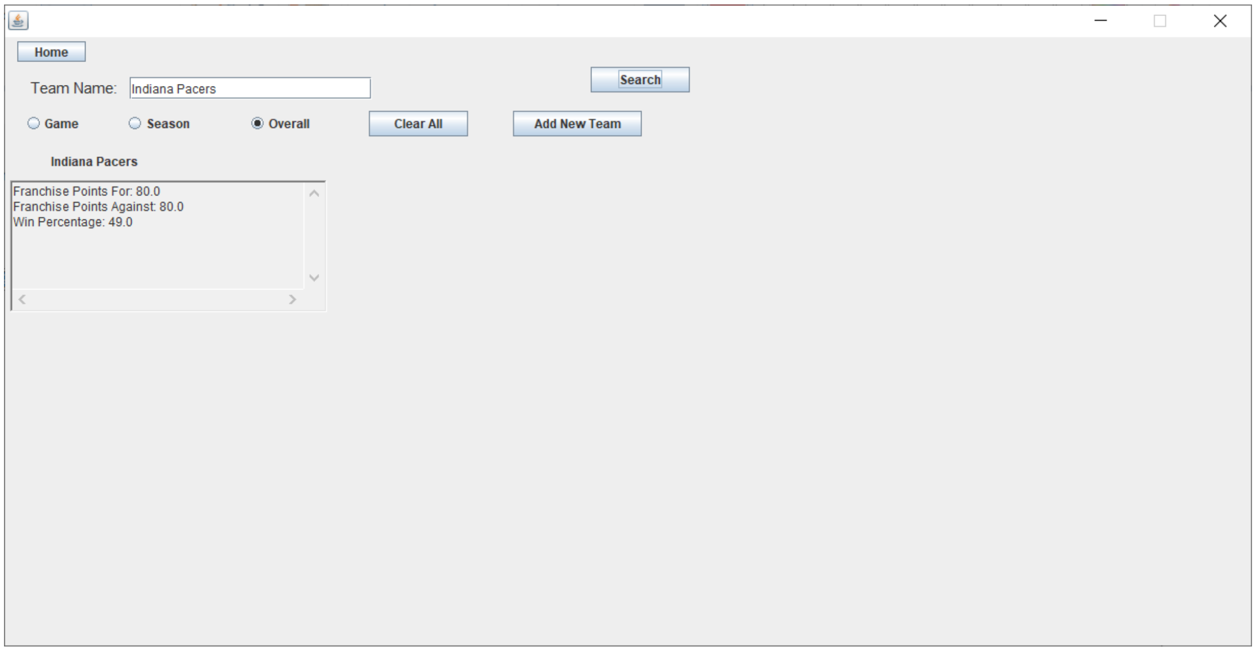
Team Name: Indiana Pacers

Season year: 2006

- The program will then display information about the team's average performance in that season

## Searching for Overall

- If you selected the **Overall** option, the program will display information about the team's average performance across all seasons



Team Name: Indiana Pacers

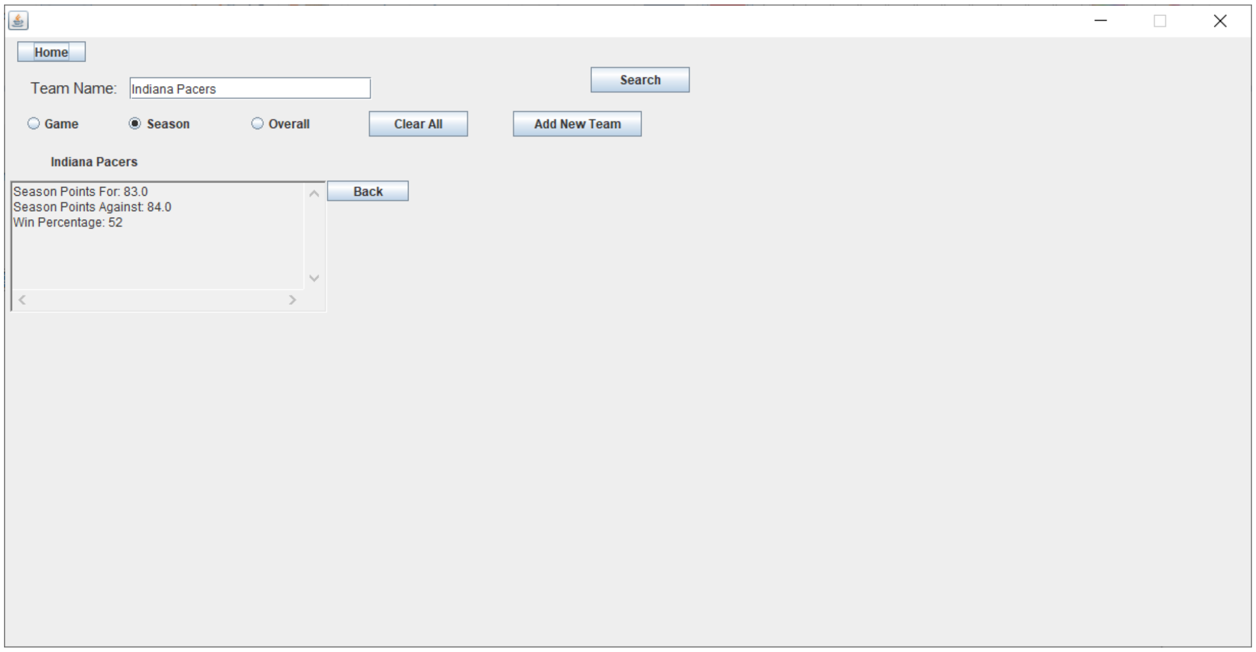
Game [✓ Overall] Season

Indiana Pacers

```
Franchise Points For: 80.0
Franchise Points Against: 80.0
Win Percentage: 49.0
```

## Display Multiple Search Items

- If you want to display multiple search items at once, search for the first team's information as normal



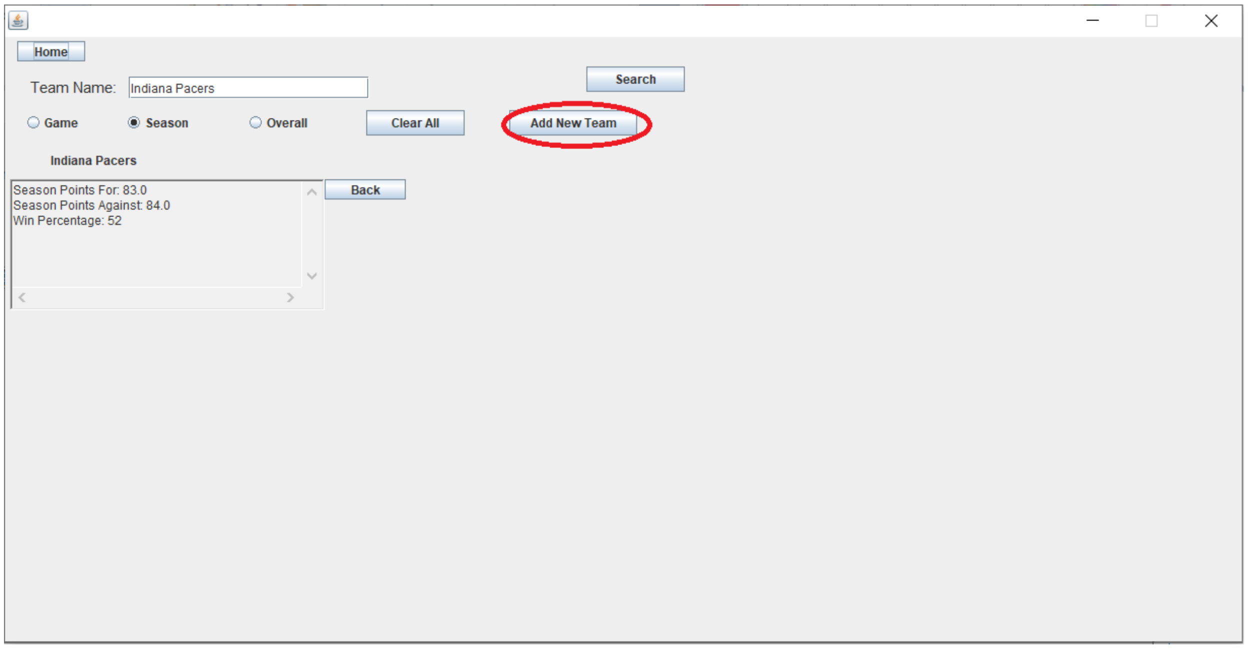
Team Name: Indiana Pacers

Game [✓ Season] Overall

Indiana Pacers

```
Season Points For: 83.0
Season Points Against: 84.0
Win Percentage: 52
```

- Then, press the *Add new Team* button



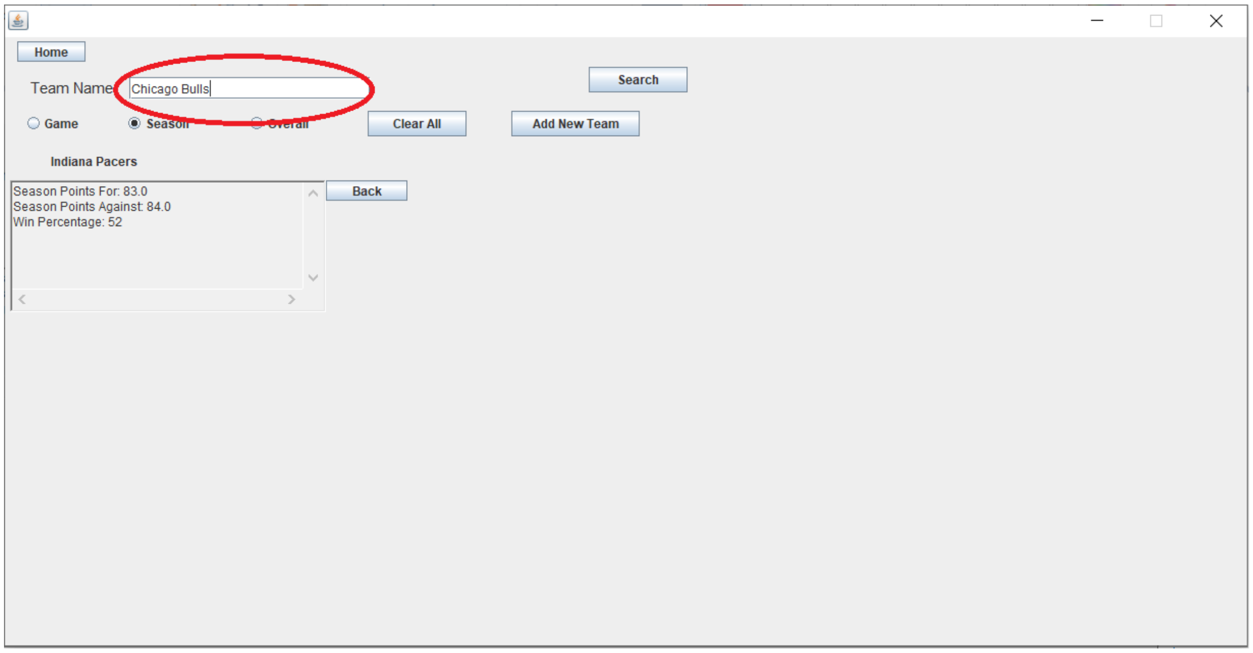
Team Name: Indiana Pacers

(●) Season

Indiana Pacers

```
Season Points For: 83.0
Season Points Against: 84.0
Win Percentage: 52
```

- Then, enter the search information for the second team (or search for the same team again to compare their performance with a different game or season)



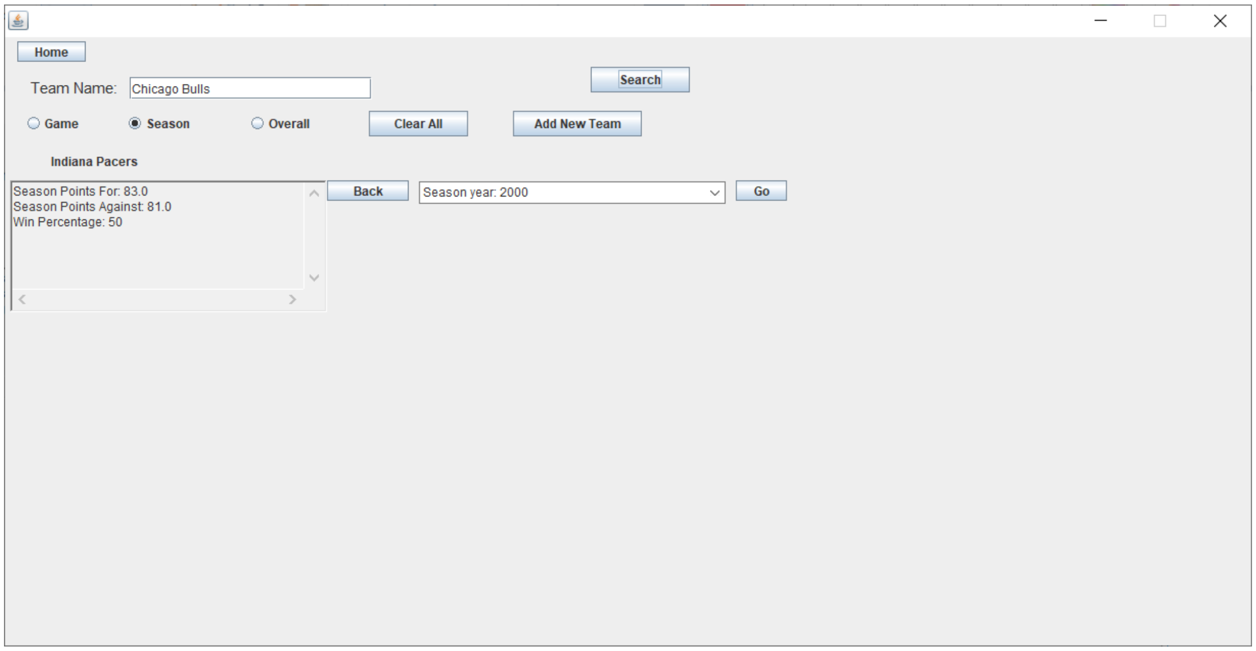
Team Name: Chicago Bulls

(●) Season

Indiana Pacers

```
Season Points For: 83.0
Season Points Against: 84.0
Win Percentage: 52
```

- When you press Search, the second team will appear next to the first, and you can continue searching as normal



Team Name: Chicago Bulls

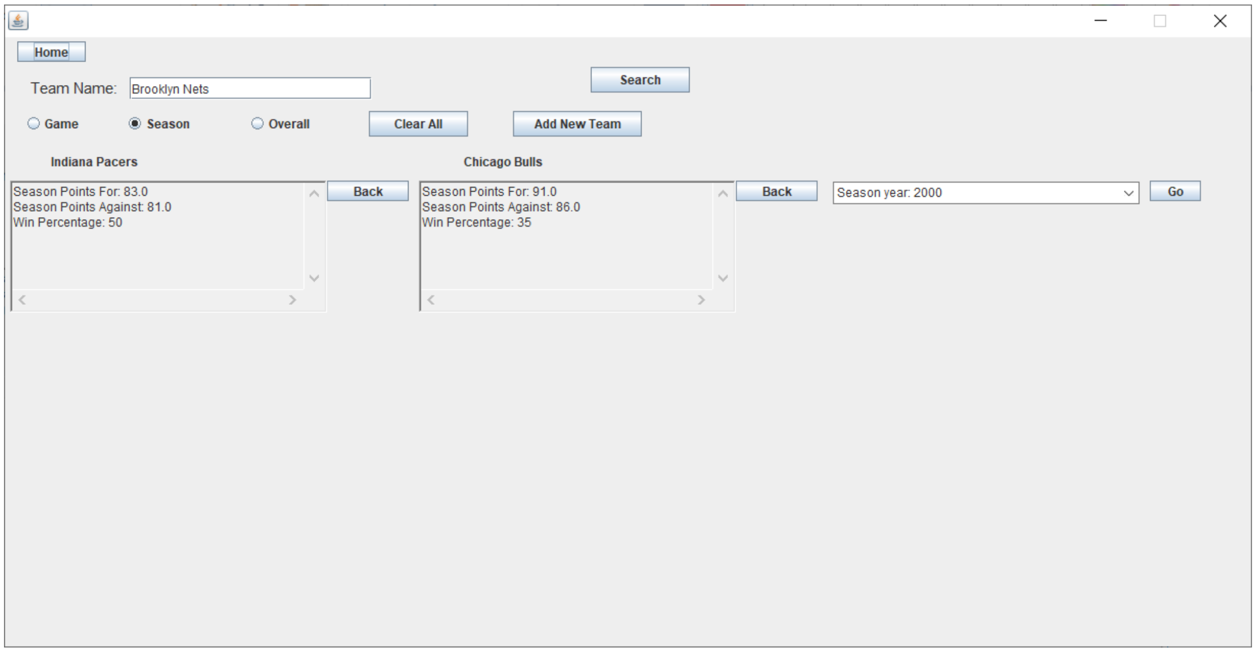
Game [✓ Season] Overall

Indiana Pacers

Season Points For: 83.0
Season Points Against: 81.0
Win Percentage: 50

Season year: 2000

- You can search for up to three teams at a time



Team Name: Brooklyn Nets

Game [✓ Season] Overall

Indiana Pacers

Season Points For: 83.0
Season Points Against: 81.0
Win Percentage: 50

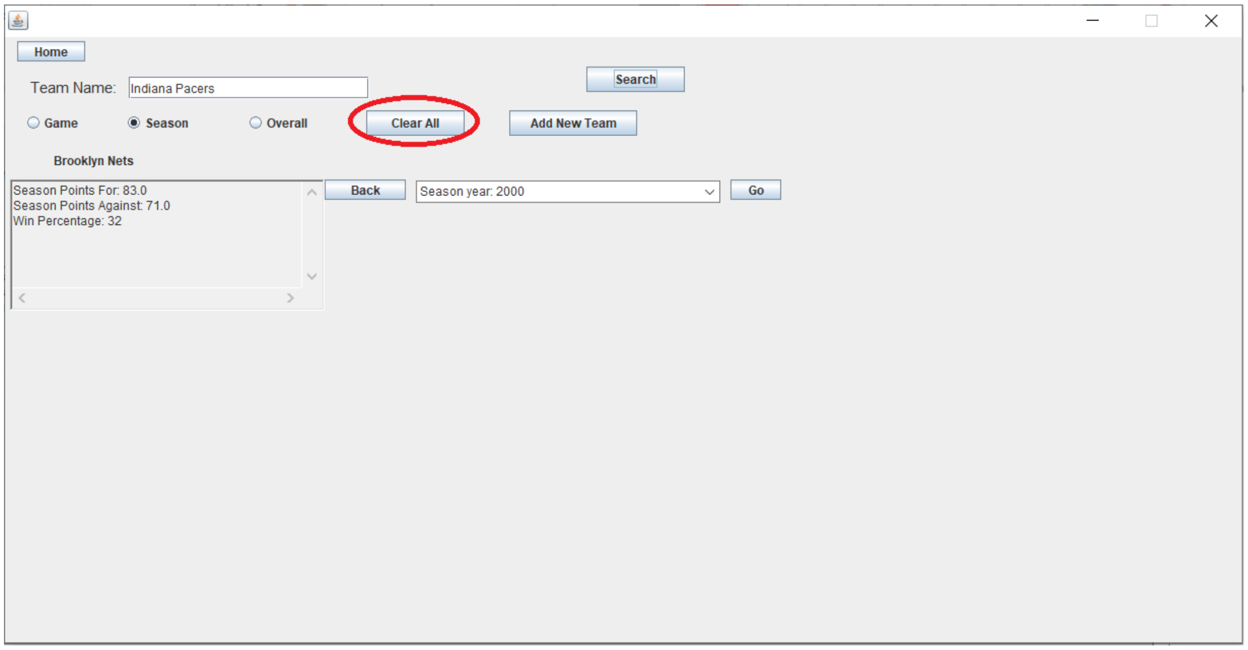
Chicago Bulls

Season Points For: 91.0
Season Points Against: 86.0
Win Percentage: 35

Season year: 2000

## Clear Data

- If you have multiple teams displayed and you would like to remove them from the display, press the *Clear All* button



Indiana Pacers

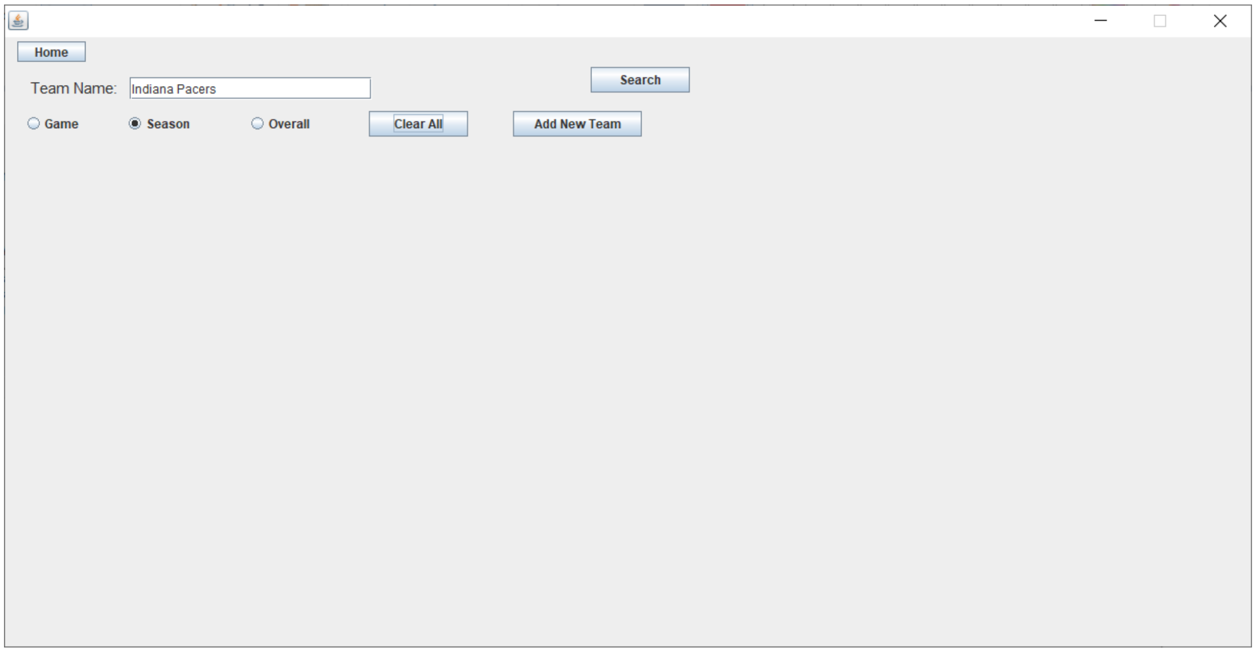
Game [✓ Season] Overall

Brooklyn Nets

Season Points For: 83.0
Season Points Against: 71.0
Win Percentage: 32

Season year: 2000

- The program will then return to a blank **Team Screen**

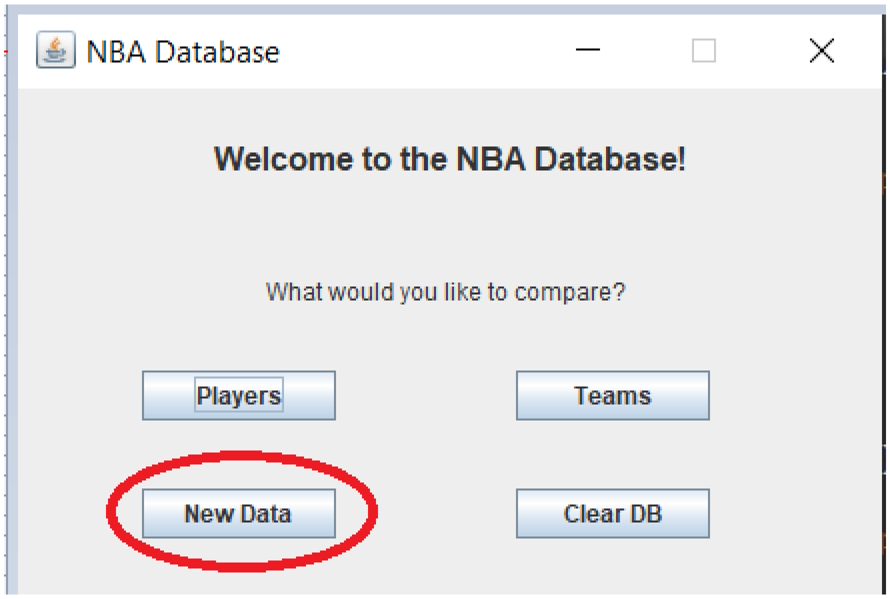


Indiana Pacers

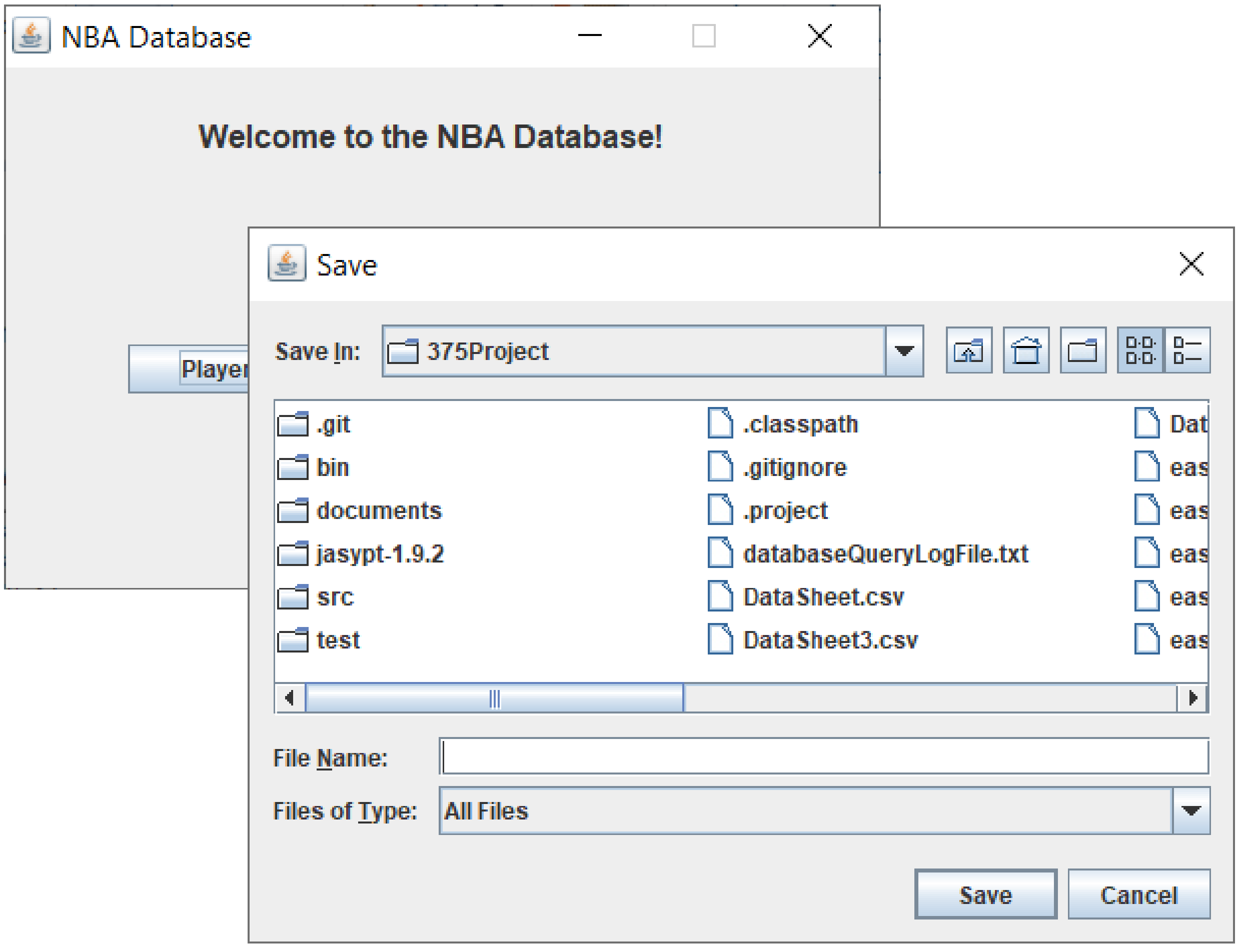
Game [✓ Season] Overall

## Adding New Data

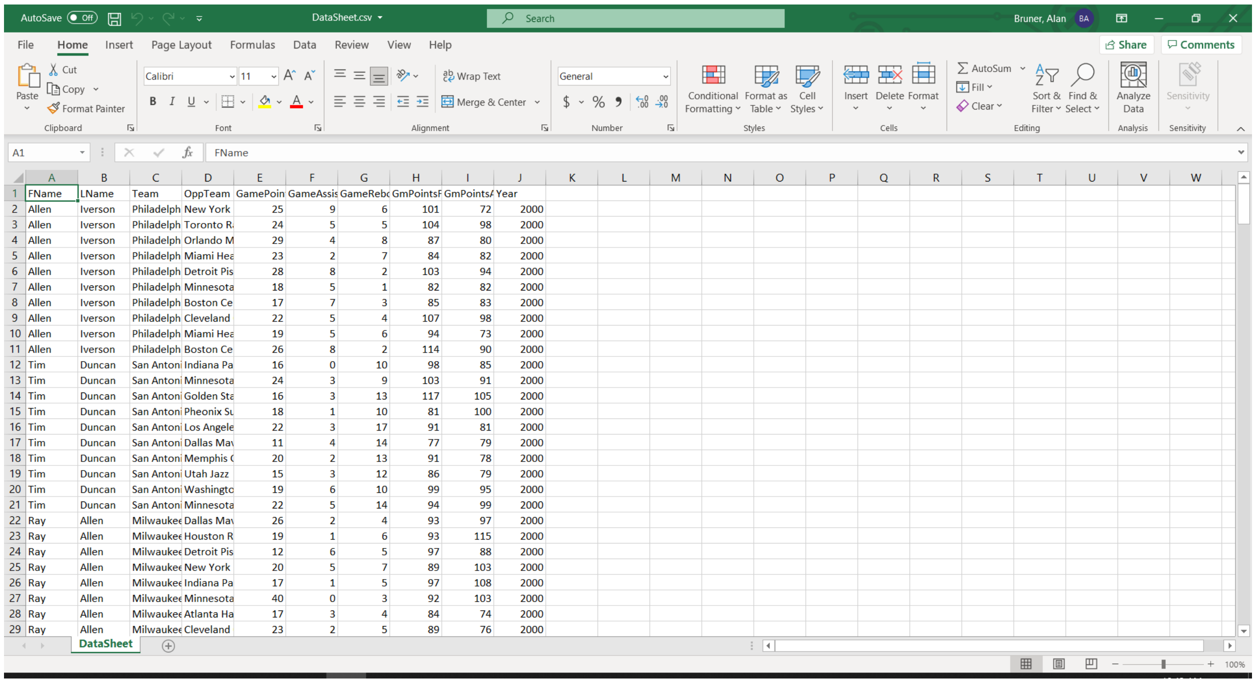
- To Add new data, navigate to the **Main Screen**, and select **New Data**



- The program will then display a window to select a file to add to the database



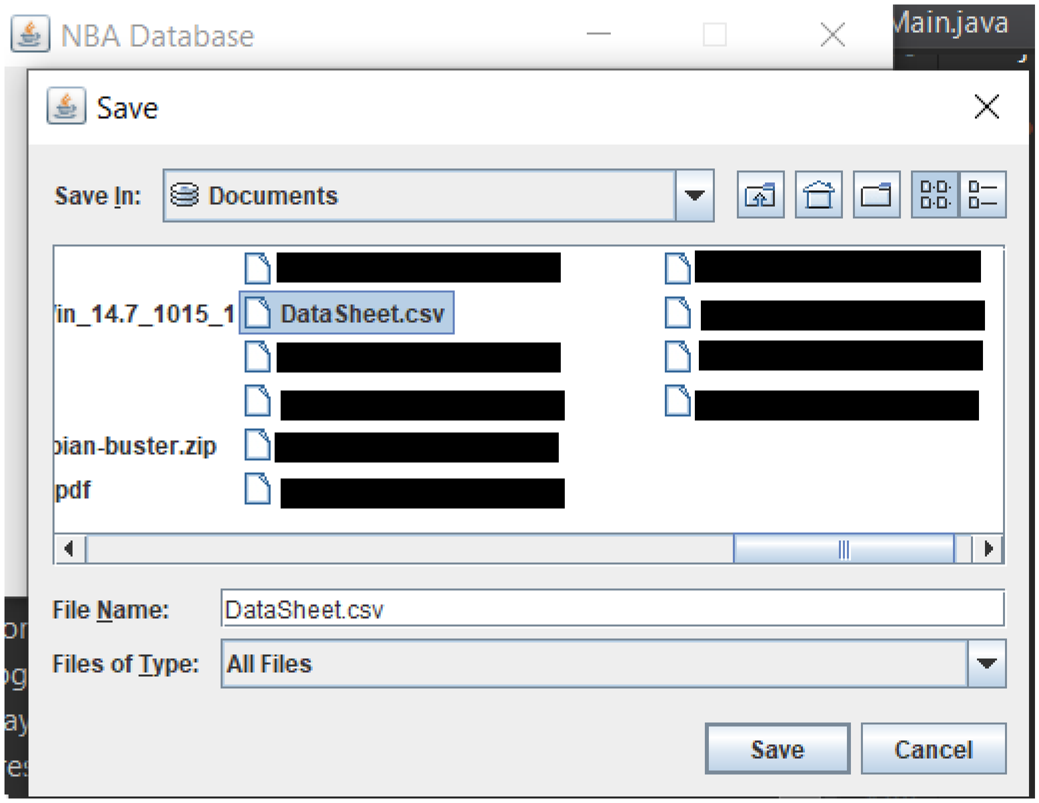
- The file you select should be a .csv file with the following layout



| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FName | LName | Team | OppTeam | GamePoin | GameAssis | GameRebc | GmPointsF | GmPointsA | Year |
| 2 | Allen | Iverson | Philadelph | New York | 25 | 9 | 6 | 101 | 72 | 2000 |
| 3 | Allen | Iverson | Philadelph | Toronto Ra | 24 | 5 | 5 | 104 | 98 | 2000 |
| 4 | Allen | Iverson | Philadelph | Orlando M | 29 | 4 | 8 | 87 | 80 | 2000 |
| 5 | Allen | Iverson | Philadelph | Miami Hea | 23 | 2 | 7 | 84 | 82 | 2000 |
| 6 | Allen | Iverson | Philadelph | Detroit Pis | 28 | 8 | 2 | 103 | 94 | 2000 |
| 7 | Allen | Iverson | Philadelph | Minnesota | 18 | 5 | 1 | 82 | 82 | 2000 |
| 8 | Allen | Iverson | Philadelph | Boston Ce | 17 | 7 | 3 | 85 | 83 | 2000 |
| 9 | Allen | Iverson | Philadelph | Cleveland | 22 | 5 | 4 | 107 | 98 | 2000 |
| 10 | Allen | Iverson | Philadelph | Miami Hea | 19 | 5 | 6 | 94 | 73 | 2000 |
| 11 | Allen | Iverson | Philadelph | Boston Ce | 26 | 8 | 2 | 114 | 90 | 2000 |
| 12 | Tim | Duncan | San Antoni | Indiana Pa | 16 | 0 | 10 | 98 | 85 | 2000 |
| 13 | Tim | Duncan | San Antoni | Minnesota | 24 | 3 | 9 | 103 | 91 | 2000 |
| 14 | Tim | Duncan | San Antoni | Golden Sta | 16 | 3 | 13 | 117 | 105 | 2000 |
| 15 | Tim | Duncan | San Antoni | Pheonix Su | 18 | 1 | 10 | 81 | 100 | 2000 |
| 16 | Tim | Duncan | San Antoni | Los Angele | 22 | 3 | 17 | 91 | 81 | 2000 |
| 17 | Tim | Duncan | San Antoni | Dallas Mav | 11 | 4 | 14 | 77 | 79 | 2000 |
| 18 | Tim | Duncan | San Antoni | Memphis ( | 20 | 2 | 13 | 91 | 78 | 2000 |
| 19 | Tim | Duncan | San Antoni | Utah Jazz | 15 | 3 | 12 | 86 | 79 | 2000 |
| 20 | Tim | Duncan | San Antoni | Washingto | 19 | 6 | 10 | 99 | 95 | 2000 |
| 21 | Tim | Duncan | San Antoni | Minnesota | 22 | 5 | 14 | 94 | 99 | 2000 |
| 22 | Ray | Allen | Milwaukee | Dallas Mav | 26 | 2 | 4 | 93 | 97 | 2000 |
| 23 | Ray | Allen | Milwaukee | Houston R | 19 | 1 | 6 | 93 | 115 | 2000 |
| 24 | Ray | Allen | Milwaukee | Detroit Pis | 12 | 6 | 5 | 97 | 88 | 2000 |
| 25 | Ray | Allen | Milwaukee | New York | 20 | 5 | 7 | 89 | 103 | 2000 |
| 26 | Ray | Allen | Milwaukee | Indiana Pa | 17 | 1 | 5 | 97 | 108 | 2000 |
| 27 | Ray | Allen | Milwaukee | Minnesota | 40 | 0 | 3 | 92 | 103 | 2000 |
| 28 | Ray | Allen | Milwaukee | Atlanta Ha | 17 | 3 | 4 | 84 | 74 | 2000 |
| 29 | Ray | Allen | Milwaukee | Cleveland | 23 | 2 | 5 | 89 | 76 | 2000 |

- Column 1: A player's first name
- Column 2: The player's last name
- Column 3: The team the player played for in a game
- Column 4: The team the player played against in a game
- Column 5: The number of points the player scored in the specified game
- Column 6: The number of assists the player had in the specified game
- Column 7: The number of rebounds the player had in the specified game
- Column 8: The total number of points the player's team scored in the specified game
- Column 9: The total number of points the opposing team scored in the specified game
- Column 10: The year the game took place
- Note: When entering data, the first row will be ignored

- Navigate to the file you want to add in the file explorer, then select it



File Name: DataSheet.csv

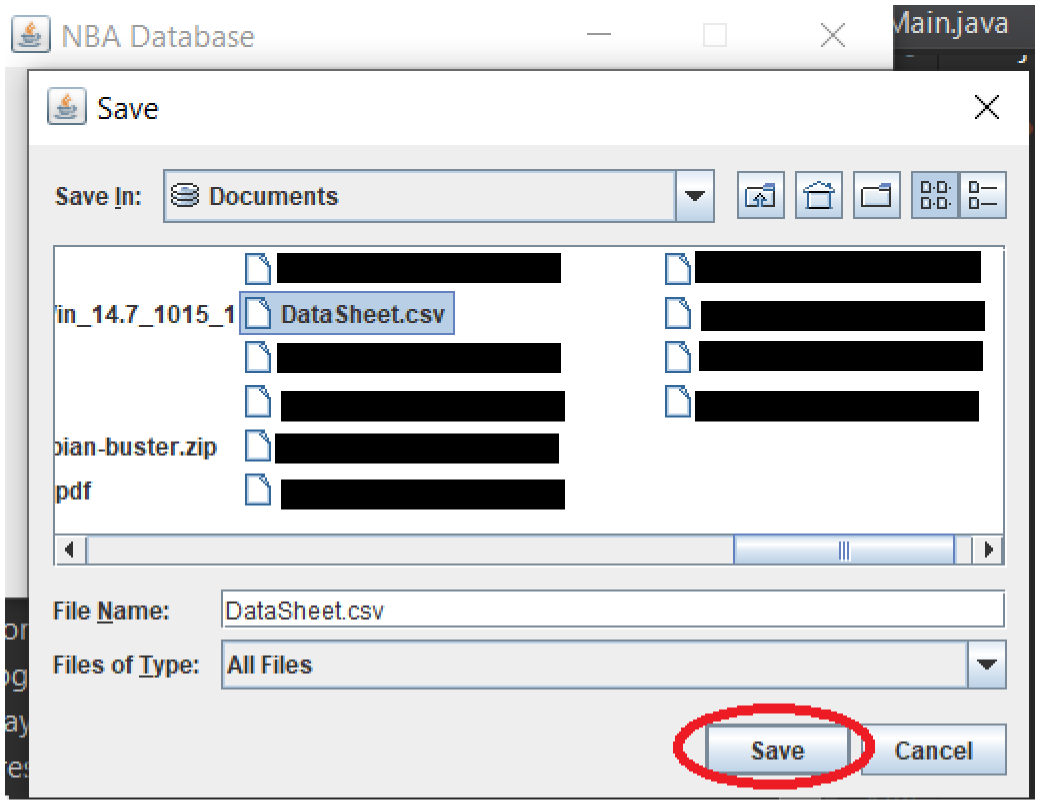
Files of Type: All Files

in_14.7_1015_1 | DataSheet.csv | [redacted]

bian-buster.zip

pdf

- Select **Save** to add the file to the database



File Name: DataSheet.csv

Files of Type: All Files

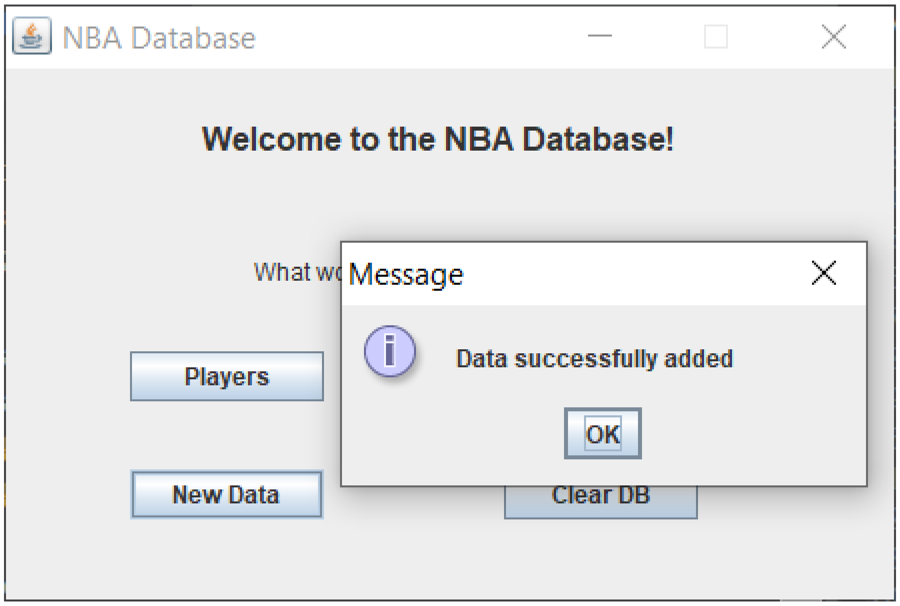
in_14.7_1015_1 | DataSheet.csv | [redacted]

bian-buster.zip

pdf

- Adding the data may take a while depending on how large the data sheet being added is

- Once the data is added, the program will then display a confirmation message

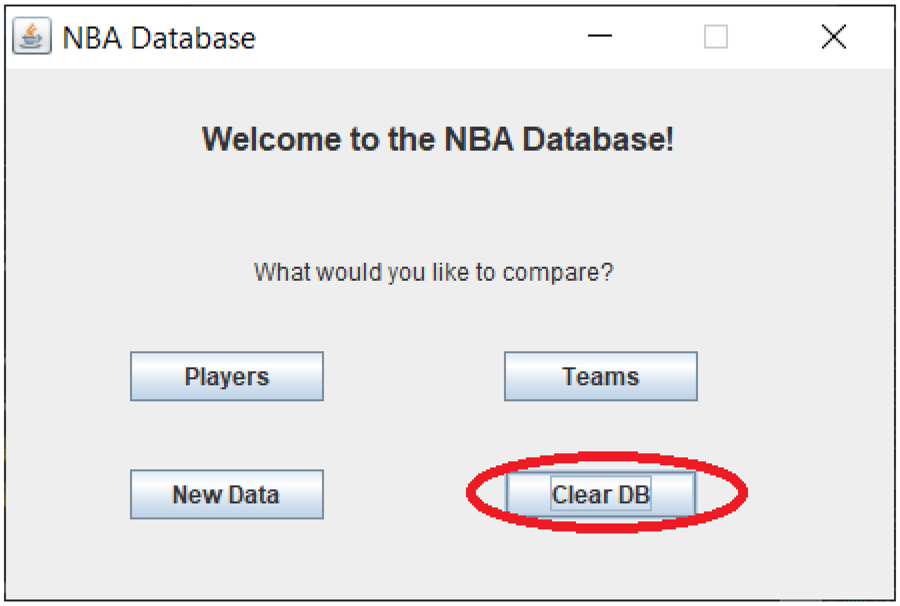


Data successfully added

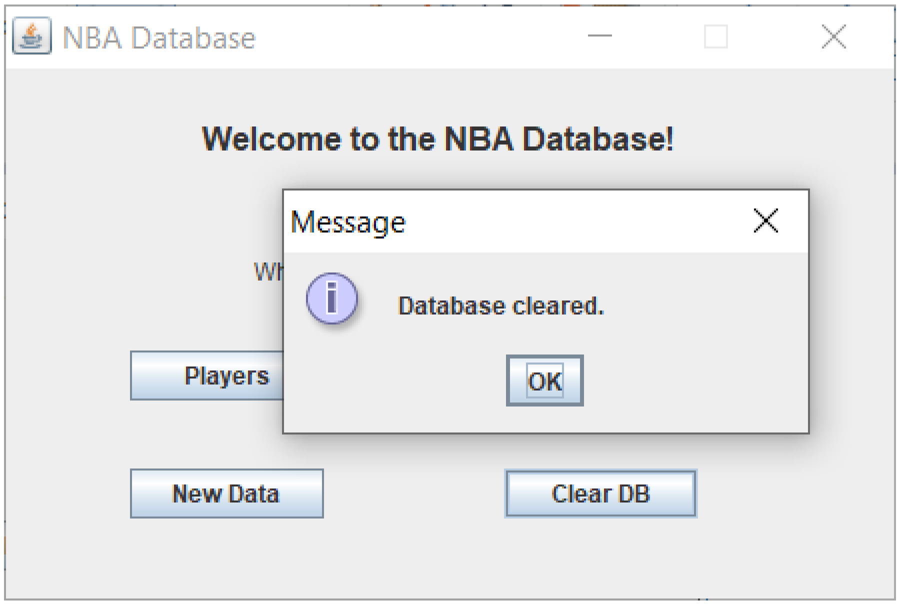
- The new data is now added into the system, and can be searched for using the above methods

## Clearing Data

- To clear all data from the database, select the **Clear DB** button on the **Main Screen**



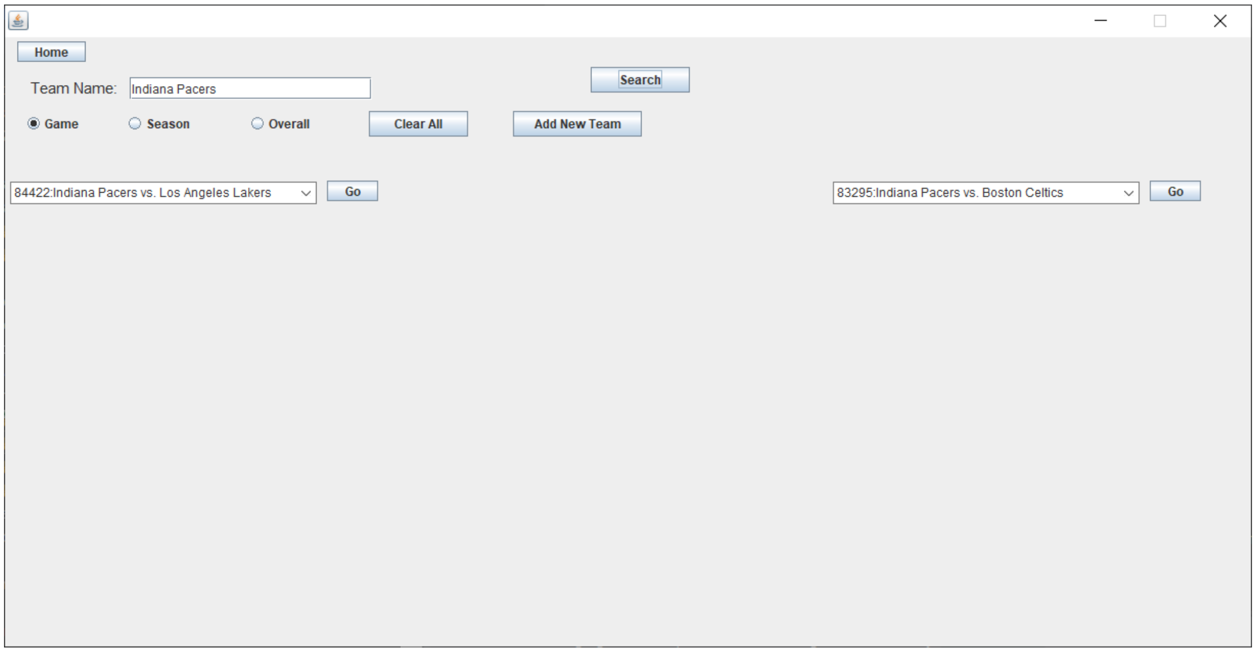
- The program will display a confirmation message, and the database will be cleared



Database cleared.

## Common Errors

- Error 1: Displaying Multiple Entries leaves empty space in between the two entries, only allowing 2 to be displayed at a time



Indiana Pacers

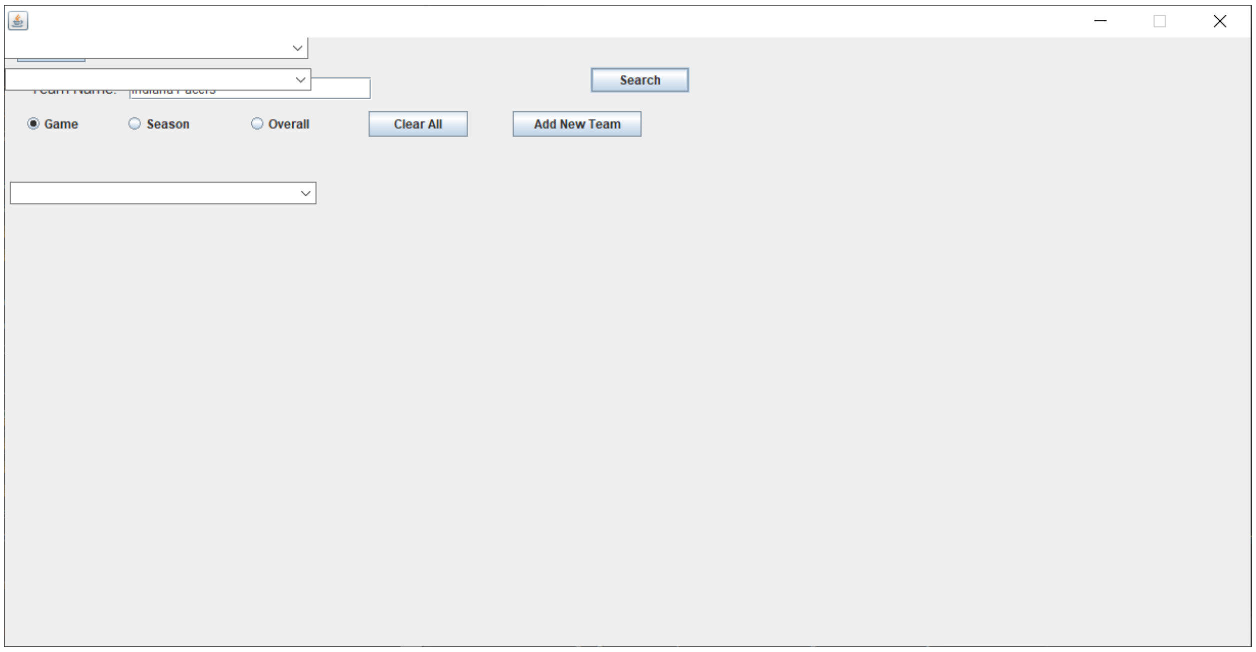
[✓ Game] Season Overall

84422:Indiana Pacers vs. Los Angeles Lakers

83295:Indiana Pacers vs. Boston Celtics

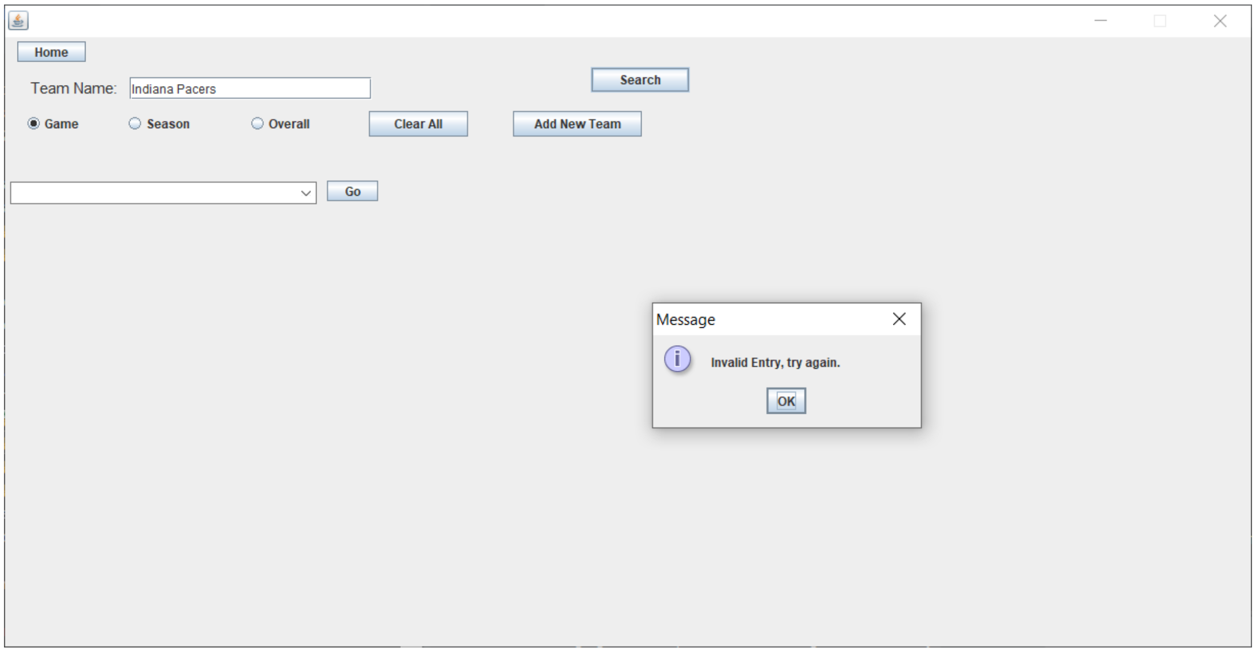
  - **Solution:** This error can occur if you are double clicking the Add New button. Make sure you only click the Add New button once for each new entry you want to display
- Error 2: The program freezes when attempting to search

Solution 4: You entered an invalid name. Make sure the spelling is correct and then search again.



- ○ Solution: Make sure your internet connection is working. The program checks for an internet connection upon launch, but if your connection drops while the program is running, this issue can happen

- Error 3: The program displays “Invalid Entry” for data that should be there



Team Name: Indiana Pacers

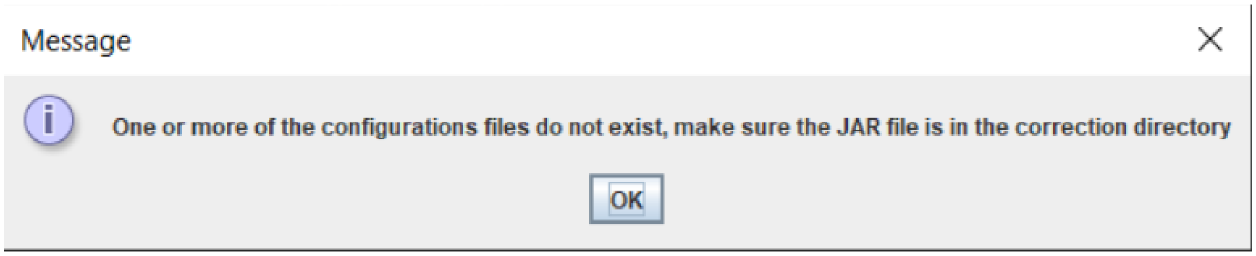
Message

Invalid Entry, try again.

- ○ Solution 1: You entered an invalid name. Make sure the spelling is correct and then search again.

- ○ Solution 2: Ensure your internet connection is working. If your internet connection drops during execution, the program may continue to display Invalid Entry, even after connection is restored
- ○ Solution 3: Ensure that the database connection is configured to the correct database, it is possible that you are connecting to the wrong database
- ○ Solution 4: If all of those fail, you may need to re-add the data to the database

- Error 4: This message appears on program launch


Message

One or more of the configurations files do not exist, make sure the JAR file is in the correction directory

- ○ Solution: Ensure that the executable jar file exists in the same folder as the "serverConfiguration" folder and that this folder includes the following files
  - ■ server.txt
  - ■ username.txt
  - ■ password.txt

# Installation, Configuration, and Developer/Maintenance Guide

## Running the Program

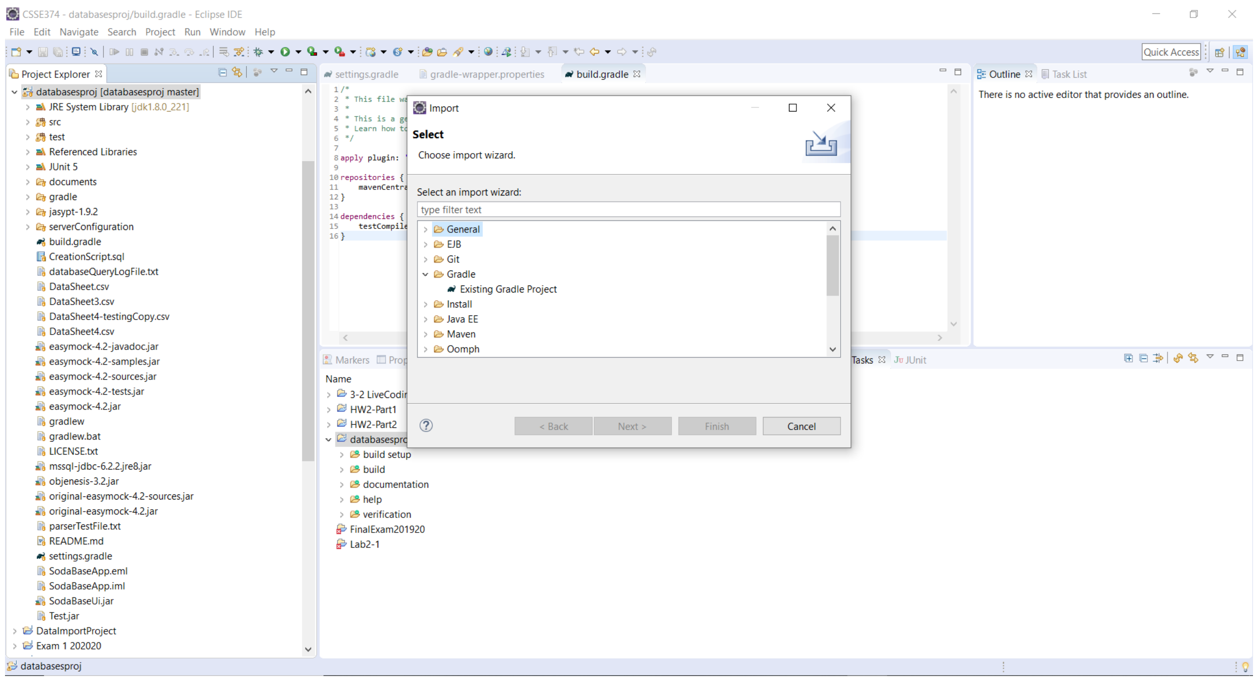
If you are not a developer, all you need to do is navigate to the java file in the command prompt and type: java -jar NBADatabase.jar

## Installation Guide

### Cloning the Repository

You can find our code repository at https://github.com/grayledw/375RefactoringProject.

You will need to clone this repository and then import the repository into Eclipse, the IDE used on the project. To import the project into Eclipse, select File > Import > Gradle > Existing Gradle Project. A screenshot of this option is shown below:


CSSE374 - databasesproj/build.gradle - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
databasesproj [databasesproj master]
Import
Select
Choose import wizard.
Select an import wizard:
type filter text
General
EJB
Git
Gradle
Existing Gradle Project
Install
Java EE
Maven
Oomph
< Back
Next >
Finish
Cancel
Outline
There is no active editor that provides an outline.

Instructions for Cloning a Git Repository:
https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository

Eclipse Download: https://www.eclipse.org/downloads/

# Continuous Integration Environment Installation and Configuration

## Introduction

Our system's continuous integration environment consists of the following components:

- GitHub
- Gradle
- Jenkins

Each of these components serves its own role in the continuous integration environment. GitHub acts as our system's version control system. Gradle is used for build automation and regression testing. Jenkins is an open source automation server that acts as a bridge between GitHub and Gradle. When a new change is made to our codebase on GitHub, this means that we want to rebuild our project and run our regression test to make sure our regression tests pass ensuring that our repository stays clean with a working version of the system. Gradle has the ability to specify how to build the project and run tests, but it needs something to tell it to perform the build. This is where Jenkins comes in; Jenkins allows you to create jobs to perform some action - such as a Gradle build.

In the following sections, instructions will be given on how to install and configure our continuous integration environment.

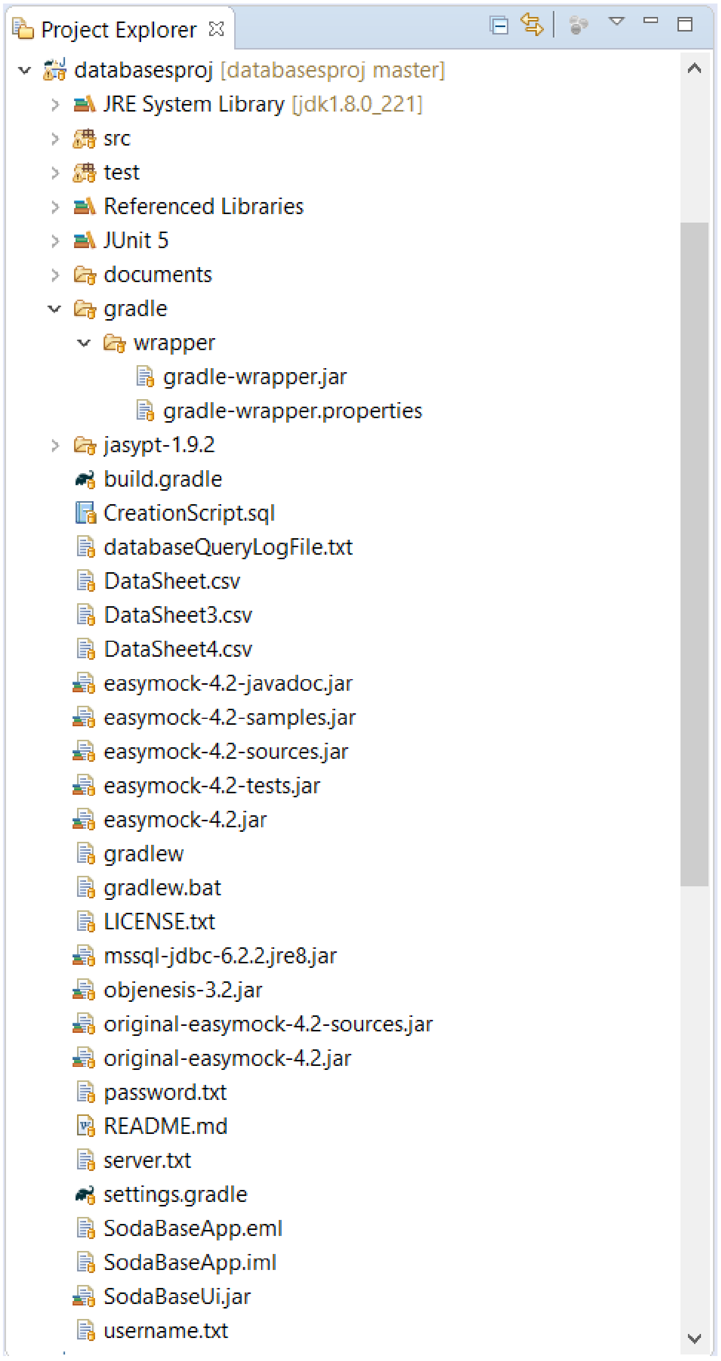
### Understanding Gradle

It is important to understand how Gradle works in order to successfully complete and understand the following steps while setting up our Continuous Integration Environment. When you have downloaded the codebase and imported it into Eclipse, the IDE used in the development of this project, you will notice that there is a small dinosaur icon in the upper left of the project's folder in the Project Explorer view - this means that the project has "Gradle Nature". There are a few files in the project directory associated with Gradle:

1) Gradle Wrapper - The Gradle Wrapper is located in the gradle/wrapper folder. This folder contains a properties file detailing properties of Gradle such as the distribution URL for the version of Gradle the project uses. It also contains "gradle-wrapper.jar" which is an executable file that contains functionality needed to fetch and install Gradle.
2) settings.gradle - This file contains the settings for Gradle.
3) build.gradle - This file is the most important Gradle file in terms of the continuous integration environment.

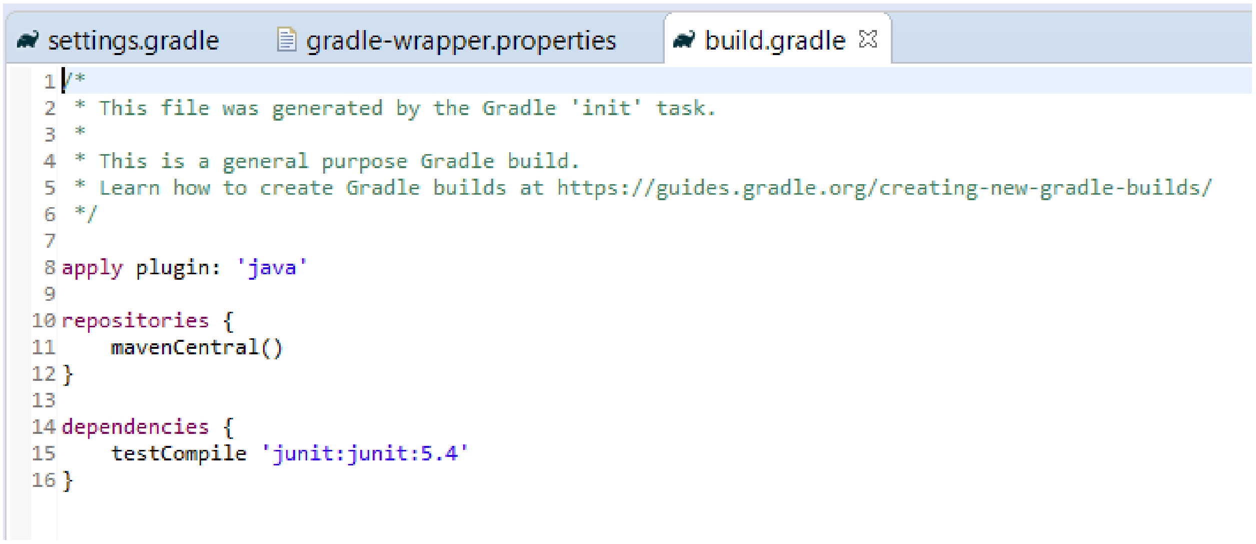
Note that if you can't view the small dinosaur Gradle icon in Eclipse next to the project, you probably don't have the "Gradle Buildship Integration" plugin installed. The plugin can be downloaded by going to Help > Eclipse Marketplace and then searching for "Gradle Buildship Integration" and installing the plugin.

Below is a picture of what the folder structure should look like:



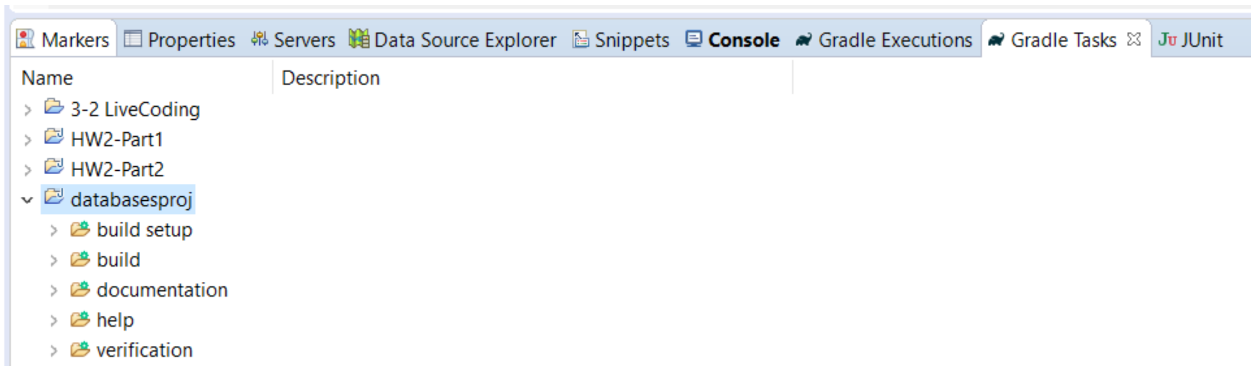
```
Project Explorer
v databasesproj [databasesproj master]
  > JRE System Library [jdk1.8.0_221]
  > src
  > test
  > Referenced Libraries
  > JUnit 5
  > documents
  v gradle
    v wrapper
        gradle-wrapper.jar
        gradle-wrapper.properties
  > jasypt-1.9.2
    build.gradle
    CreationScript.sql
    databaseQueryLogFile.txt
    DataSheet.csv
    DataSheet3.csv
    DataSheet4.csv
    easymock-4.2-javadoc.jar
    easymock-4.2-samples.jar
    easymock-4.2-sources.jar
    easymock-4.2-tests.jar
    easymock-4.2.jar
    gradlew
    gradlew.bat
    LICENSE.txt
    mssql-jdbc-6.2.2.jre8.jar
    objenesis-3.2.jar
    original-easymock-4.2-sources.jar
    original-easymock-4.2.jar
    password.txt
    README.md
    server.txt
    settings.gradle
    SodaBaseApp.eml
    SodaBaseApp.iml
    SodaBaseUi.jar
    username.txt
```

Open the build.gradle file. The contents of this file specify important information Gradle will need in order to build. This file specifies that the Maven repository will be used to handle any dependencies and the tests will be compiled with JUnit 5.4



```
/*
 * This file was generated by the Gradle 'init' task.
 *
 * This is a general purpose Gradle build.
 * Learn how to create Gradle builds at https://guides.gradle.org/creating-new-gradle-builds/
 */

apply plugin: 'java'

repositories {
    mavenCentral()
}

dependencies {
    testCompile 'junit:junit:5.4'
}
```

The Gradle tasks and executions can be viewed within Eclipse by Window > Show View > Other > Gradle and then selecting both the “Gradle Tasks” and “Gradle Executions” options. When these appear they should look like this:



## Installing and Configuring Jenkins

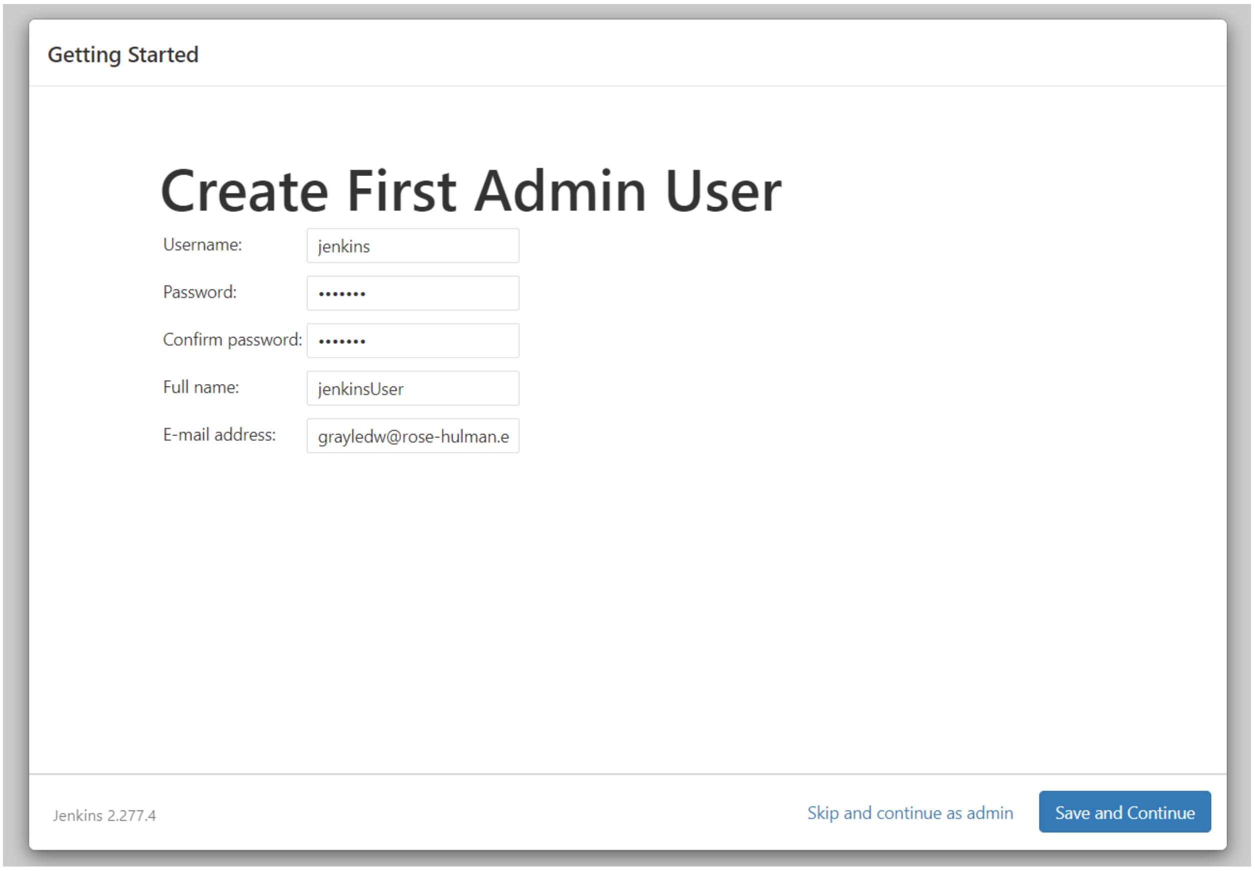
Instructions for installing Jenkins can be found at https://www.jenkins.io/doc/book/installing/. Please download the appropriate installer and run the executable installer once it has downloaded to your machine. The following steps will walk you through which options to select in the installer and how to configure Jenkins once it is installed.

Once you have run the Jenkins installer, click next through the installer accepting all of the defaults, except for the screen asking which user to run Jenkins as - on this screen it is sufficient to run Jenkins as LocalSystem.

After you have finished working through the Jenkins installer, open up your favorite browser and open “localhost:8080”. This should open Jenkins.

There will now be a few steps to work through in order to configure Jenkins.

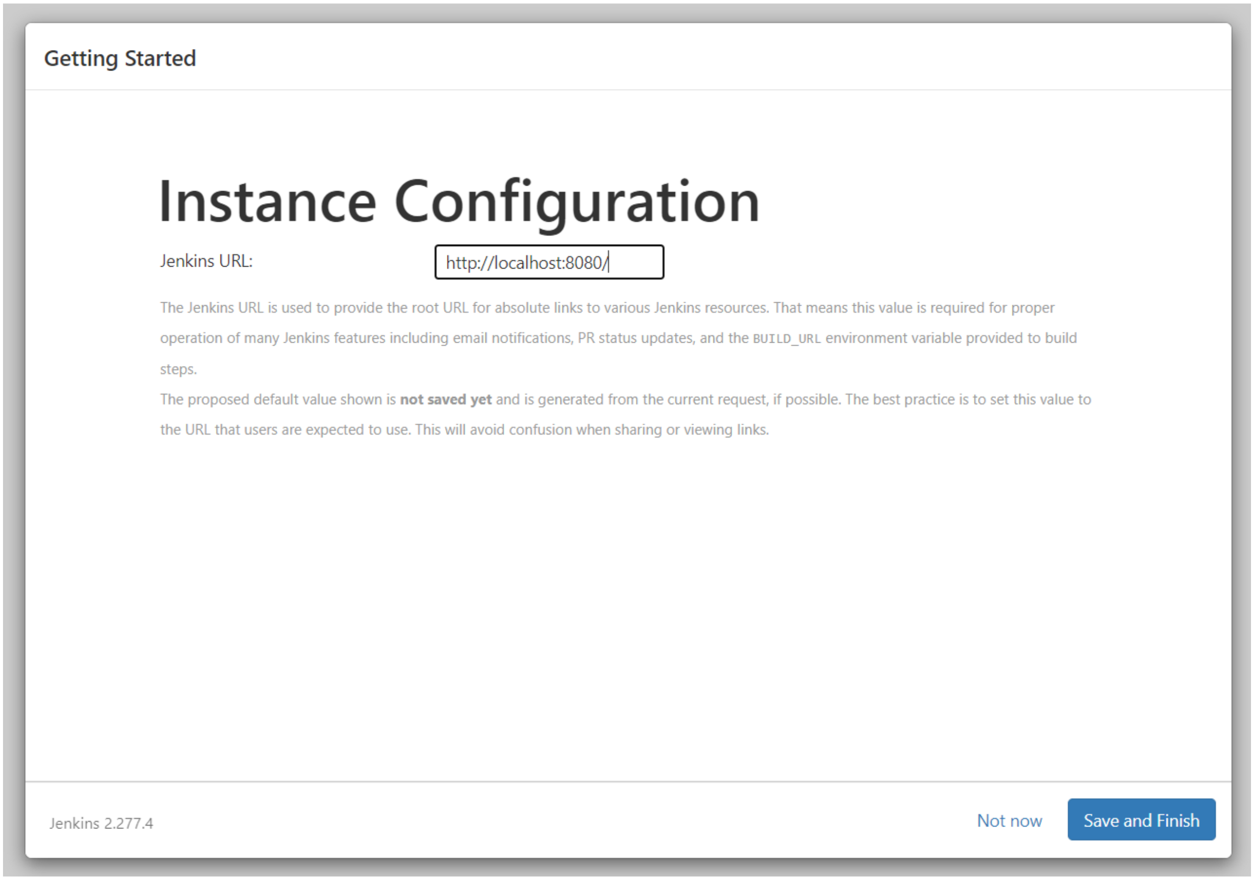
When asked to “Create First Admin User”, please enter whatever credentials you see fit. Since this instance of Jenkins will run locally on your machine, there will not be multiple accounts. If this instance on Jenkins was being hosted online, then more security measures should be taken.



Create First Admin User

| | |
|---|---|
| Username: | jenkins |
| Password: | ••••••• |
| Confirm password: | ••••••• |
| Full name: | jenkinsUser |
| E-mail address: | grayledw@rose-hulman.e |

On the Instance Configuration screen, accept the default location of “localhost:8080”.


Getting Started

Instance Configuration

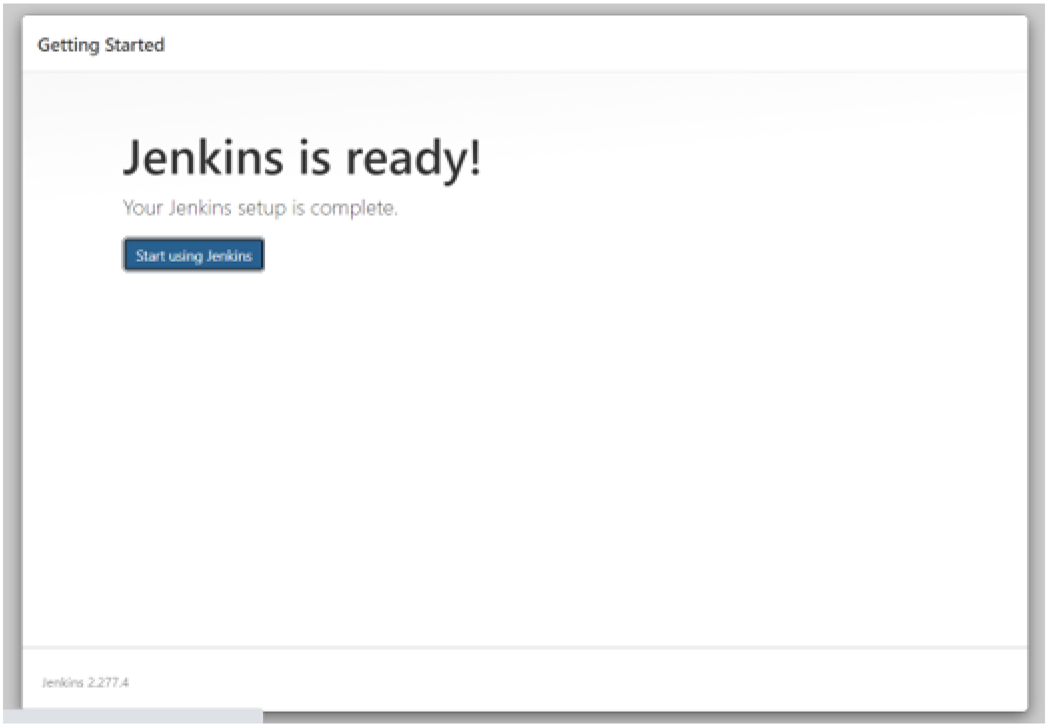
Jenkins URL: http://localhost:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.277.4

Once you complete the two steps above, you should see a screen saying “Jenkins is ready!” Click the “Start using Jenkins” button on this screen and then you should be redirected to your Jenkins dashboard. Screenshots of these screens are shown below.

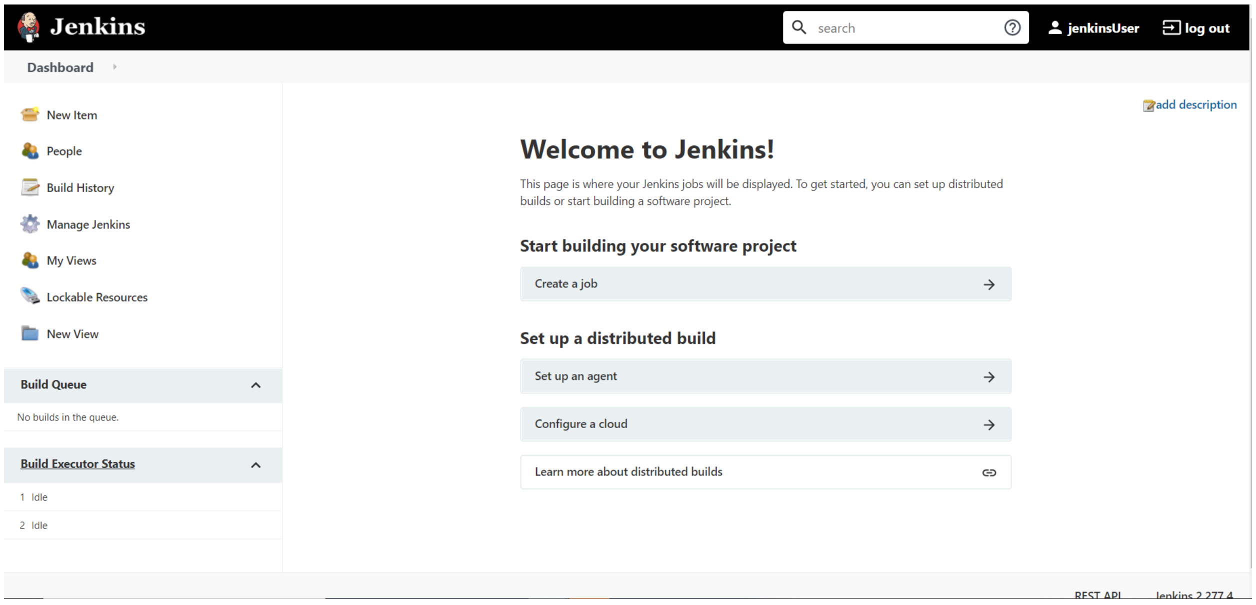


Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

Jenkins 2.277.4



Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

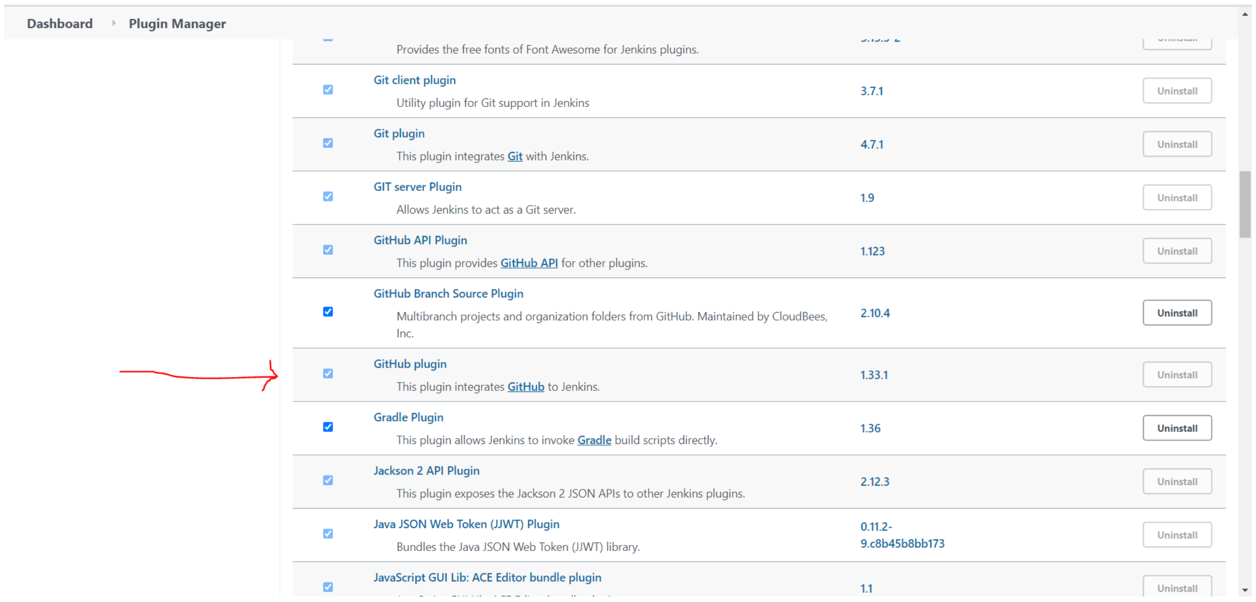
Start building your software project

Set up a distributed build

## Creating A Build Job on Jenkins

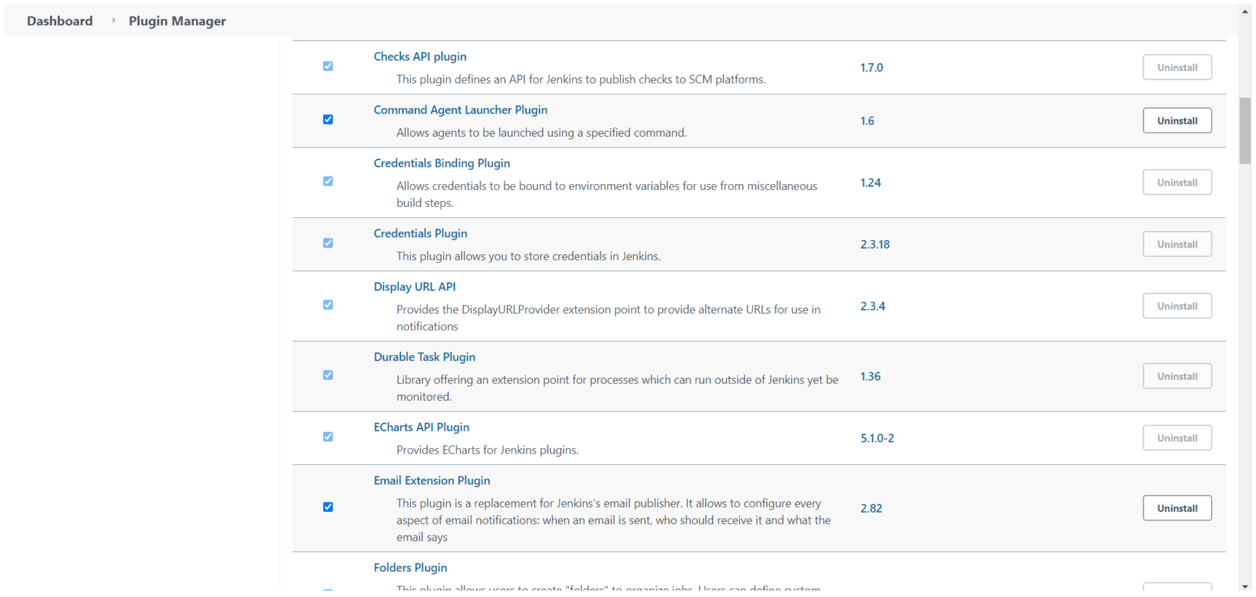
The following steps will now show you how to create a new job in Jenkins so that we can trigger our project's Gradle build.

Before actually creating our job, we need to make sure that the GitHub and Credentials plugins are installed. Click “Manage Jenkins”. First click, “Manage Plugins”. Ensure that the two plugins shown in the screenshots below are installed.



Dashboard › Plugin Manager

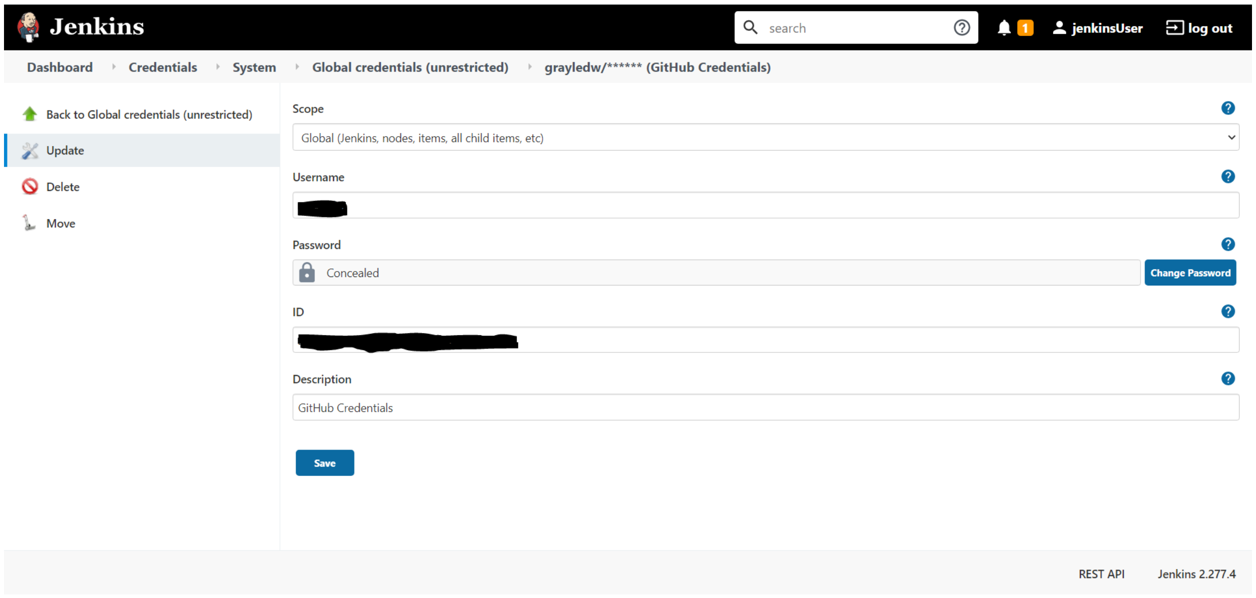
| | | | |
|---|---|---|---|
| | Provides the free fonts of Font Awesome for Jenkins plugins. | | |
| ✓ | Git client plugin<br>Utility plugin for Git support in Jenkins | 3.7.1 | Uninstall |
| ✓ | Git plugin<br>This plugin integrates Git with Jenkins. | 4.7.1 | Uninstall |
| ✓ | GIT server Plugin<br>Allows Jenkins to act as a Git server. | 1.9 | Uninstall |
| ✓ | GitHub API Plugin<br>This plugin provides GitHub API for other plugins. | 1.123 | Uninstall |
| ✓ | GitHub Branch Source Plugin<br>Multibranch projects and organization folders from GitHub. Maintained by CloudBees, Inc. | 2.10.4 | Uninstall |
| ✓ | GitHub plugin<br>This plugin integrates GitHub to Jenkins. | 1.33.1 | Uninstall |
| ✓ | Gradle Plugin<br>This plugin allows Jenkins to invoke Gradle build scripts directly. | 1.36 | Uninstall |
| ✓ | Jackson 2 API Plugin<br>This plugin exposes the Jackson 2 JSON APIs to other Jenkins plugins. | 2.12.3 | Uninstall |
| ✓ | Java JSON Web Token (JJWT) Plugin<br>Bundles the Java JSON Web Token (JJWT) library. | 0.11.2-9.c8b45b8bb173 | Uninstall |
| ✓ | JavaScript GUI Lib: ACE Editor bundle plugin | 1.1 | Uninstall |



Dashboard › Plugin Manager

| | | | |
|---|---|---|---|
| ✓ | Checks API plugin<br>This plugin defines an API for Jenkins to publish checks to SCM platforms. | 1.7.0 | Uninstall |
| ✓ | Command Agent Launcher Plugin<br>Allows agents to be launched using a specified command. | 1.6 | Uninstall |
| ✓ | Credentials Binding Plugin<br>Allows credentials to be bound to environment variables for use from miscellaneous build steps. | 1.24 | Uninstall |
| ✓ | Credentials Plugin<br>This plugin allows you to store credentials in Jenkins. | 2.3.18 | Uninstall |
| ✓ | Display URL API<br>Provides the DisplayURLProvider extension point to provide alternate URLs for use in notifications | 2.3.4 | Uninstall |
| ✓ | Durable Task Plugin<br>Library offering an extension point for processes which can run outside of Jenkins yet be monitored. | 1.36 | Uninstall |
| ✓ | ECharts API Plugin<br>Provides ECharts for Jenkins plugins. | 5.1.0-2 | Uninstall |
| ✓ | Email Extension Plugin<br>This plugin is a replacement for Jenkins's email publisher. It allows to configure every aspect of email notifications: when an email is sent, who should receive it and what the email says | 2.82 | Uninstall |
| | Folders Plugin | | |

Once you have ensured that two above to plugins are installed, navigate back to “Manage Jenkins” and click “Manage Credentials”. Once you are on the Manage Credentials screen click on “Jenkins” under “Stores scoped to Jenkins”, then click on “Global credentials (unrestricted)” under “System”, and then click “Add Credentials”. Here we will be creating a credential for GitHub that gives us access to our repository. Please use the settings you see in the image below as well as your own Username/Password for your GitHub account. Please leave ID blank as this will be auto-generated by GitHub once you save your credentials. Once you have finished entering this information, please save the newly added credential.



Jenkins | search | 1 | jenkinsUser | log out

Dashboard › Credentials › System › Global credentials (unrestricted) › grayledw/****** (GitHub Credentials)

- Back to Global credentials (unrestricted)
- Update
- Delete
- Move

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: [redacted]

Password: Concealed [Change Password]

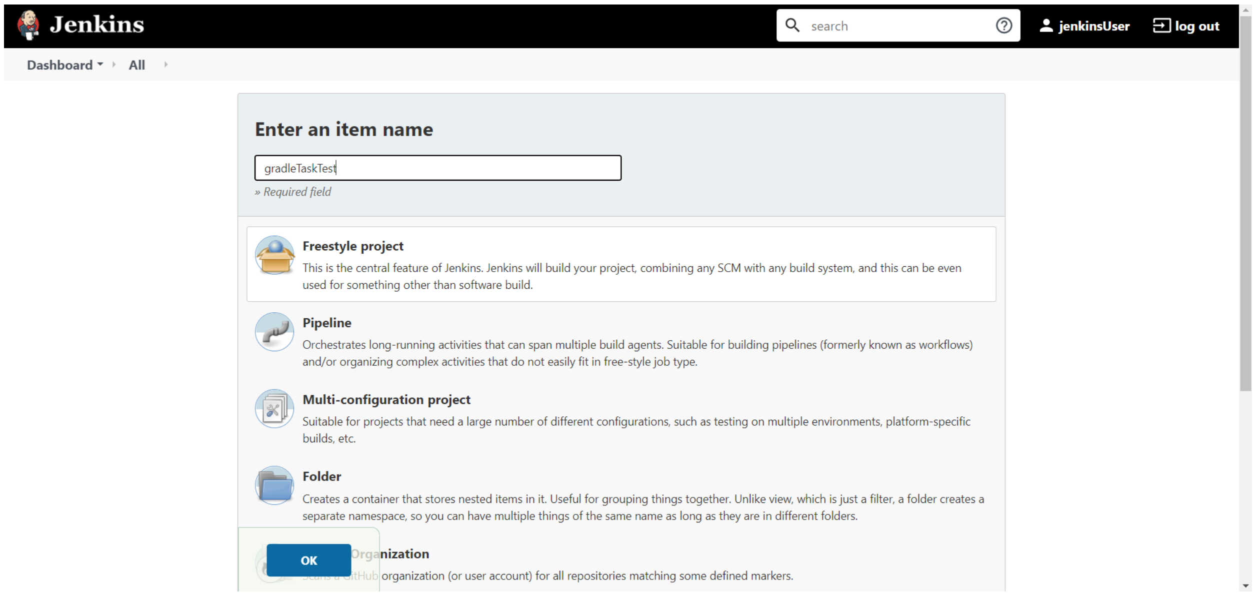
ID: [redacted]

Description: GitHub Credentials

Save

REST API | Jenkins 2.277.4

Next, navigate back to your Jenkins dashboard. On the right hand side of your Jenkins dashboard, select "New Item". On the screen that follows, select "Freestyle Project" and provide a name - "gradleTests" was the name used in this installation guide.



Jenkins | search | jenkinsUser | log out

Dashboard › All ›

Enter an item name

gradleTaskTest

» Required field

**Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

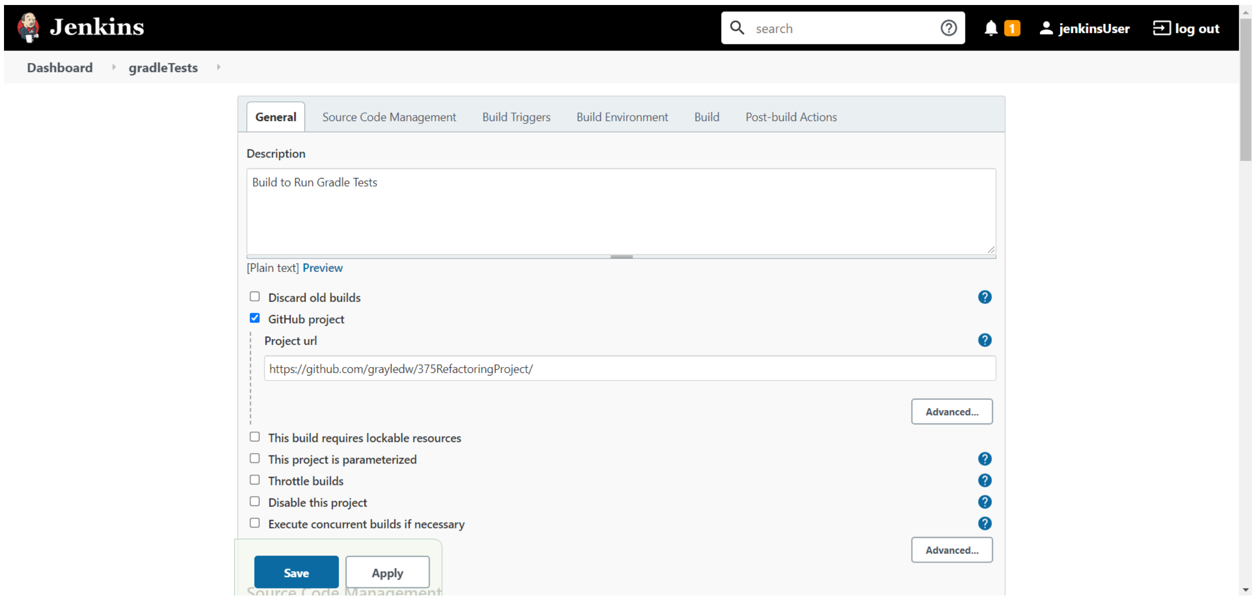
**Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

**Organization**
organization (or user account) for all repositories matching some defined markers.

Now we will configure the Gradle build. Please use the images below as a guide for what information we need to provide.

In the image below we provide a brief description of what this Jenkins task will do as well as specify that this task is for a GitHub project and provide the Project url. The project url is https://github.com/grayledw/375RefactoringProject.



Description

Build to Run Gradle Tests

[Plain text] Preview

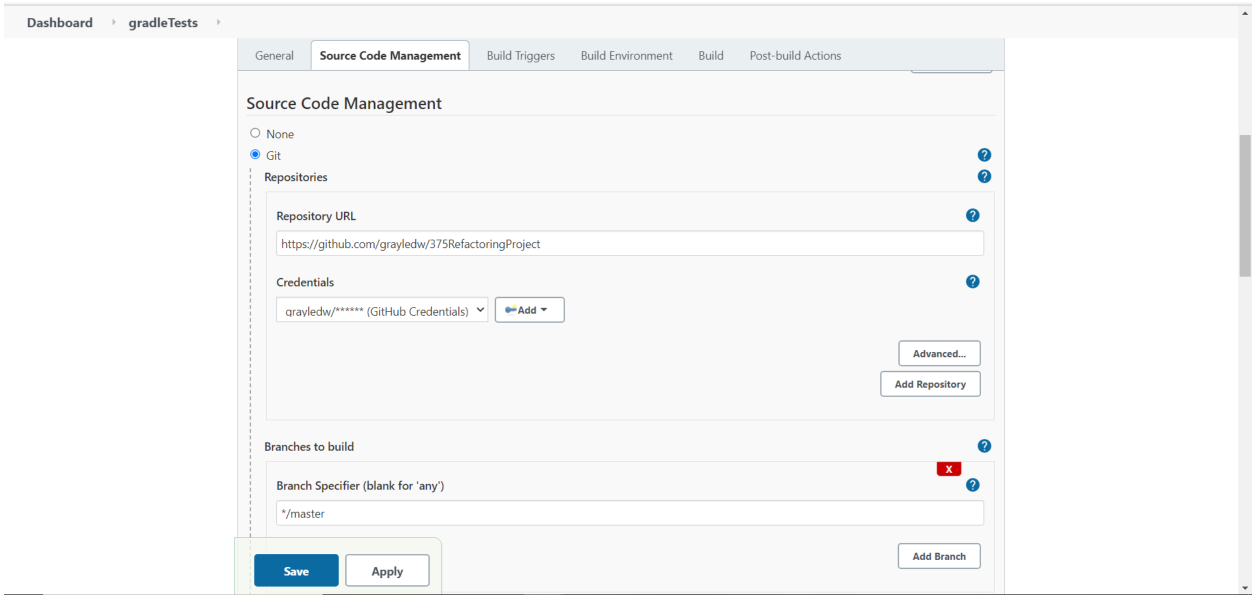
- [ ] Discard old builds
- [x] GitHub project

Project url

https://github.com/grayledw/375RefactoringProject/

- [ ] This build requires lockable resources
- [ ] This project is parameterized
- [ ] Throttle builds
- [ ] Disable this project
- [ ] Execute concurrent builds if necessary

In the image below we tell Jenkins that we are using Git for Source Code Management, give it the repository url - https://github.com/grayledw/375RefactoringProject, specify our GitHub credentials (we provided this to Jenkins in an earlier step - if your credentials don't appear please go back to Credentials step and verify that you completed everything as described), and then we specify which branch we want to perform the Jenkins job on.



Source Code Management

- None
- [✓ Git]

Repositories

Repository URL

https://github.com/grayledw/375RefactoringProject

Credentials

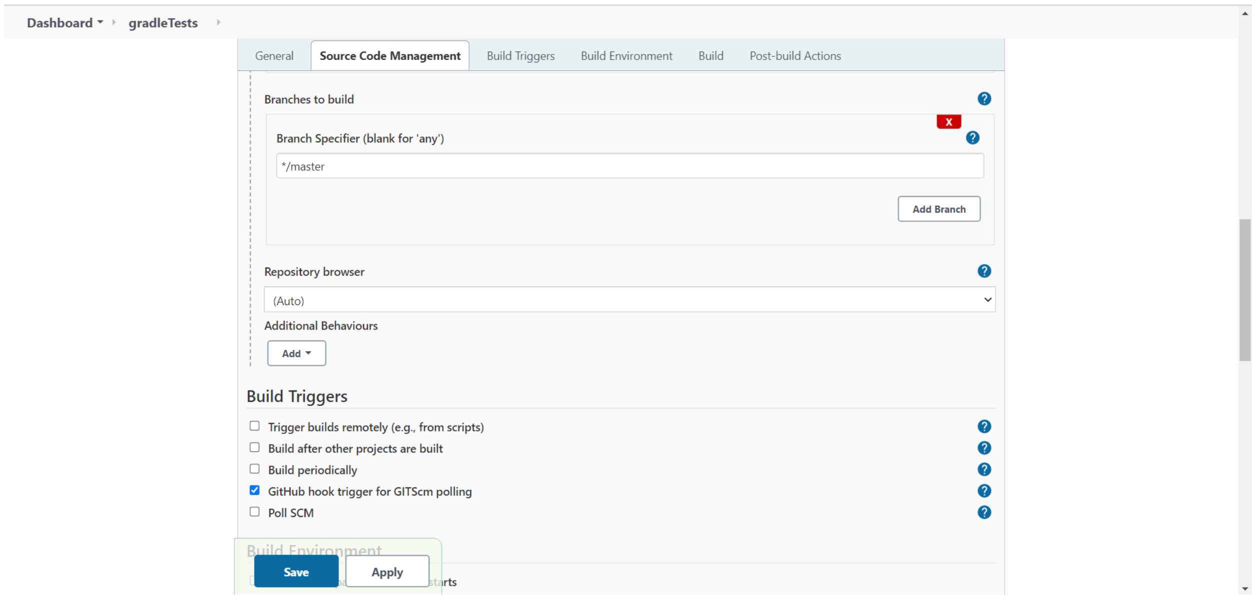
grayledw/****** (GitHub Credentials)

Branches to build

Branch Specifier (blank for 'any')

*/master

In the image below, we specify a Build Trigger for GitHub to allow GitHub to contact Jenkins when a change is made to the repository to trigger the Jenkins job. To do this select the “GitHub hook trigger for GITScm polling”. In later steps we will explain why with our current configuration this doesn’t fully work, but this is how you would do it.



Branches to build

Branch Specifier (blank for 'any')

*/master

Repository browser

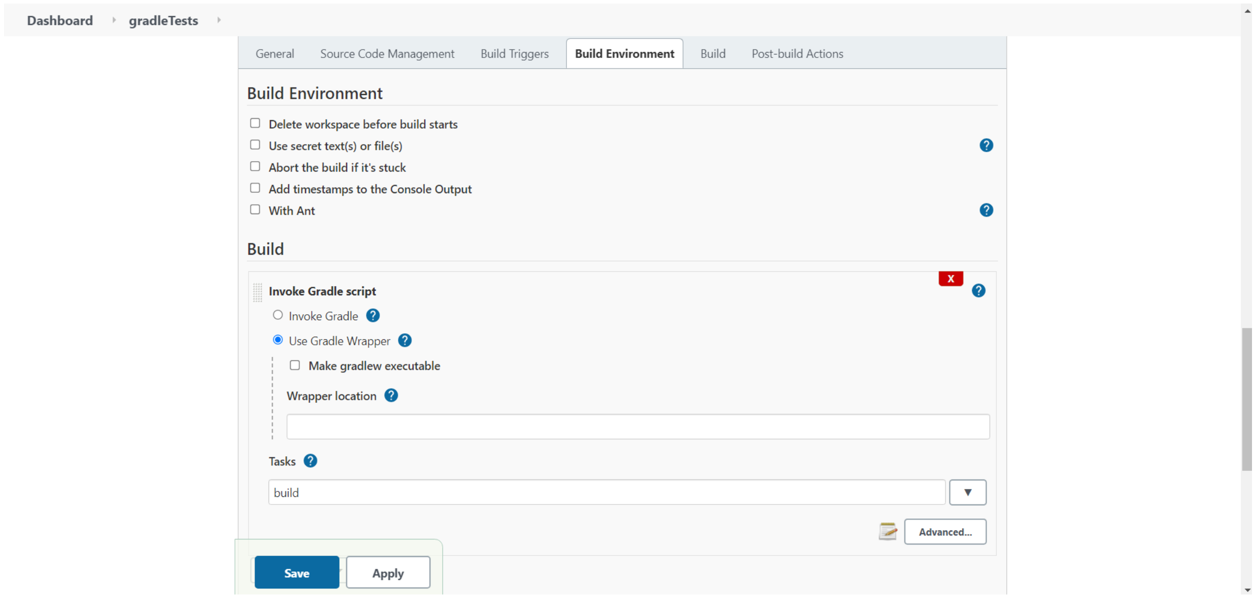
(Auto)

Additional Behaviours

Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build after other projects are built
- [ ] Build periodically
- [x] GitHub hook trigger for GITScm polling
- [ ] Poll SCM

In the image below we specify how we will build the project. Select “Use Gradle Wrapper” and then in the Tasks input box enter “build”.



Build Environment

- [ ] Delete workspace before build starts
- [ ] Use secret text(s) or file(s)
- [ ] Abort the build if it's stuck
- [ ] Add timestamps to the Console Output
- [ ] With Ant

Build

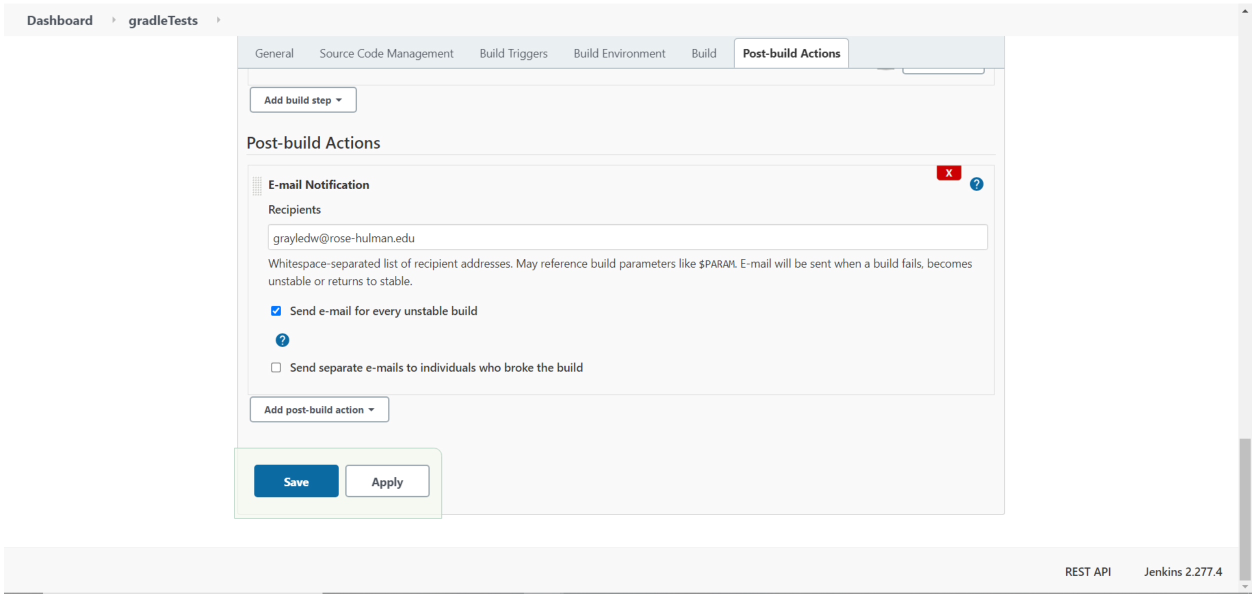
Invoke Gradle script

- ( ) Invoke Gradle
- (●) Use Gradle Wrapper
  - [ ] Make gradlew executable
  - Wrapper location

Tasks

build

In the image below, we can specify post-build actions. Here we specify an email address to be notified of the Jenkins job results whenever the job is ran. This also does not fully work due to our current configuration, but this is how you would do it.


Dashboard › gradleTests ›

Post-build Actions

E-mail Notification

Recipients

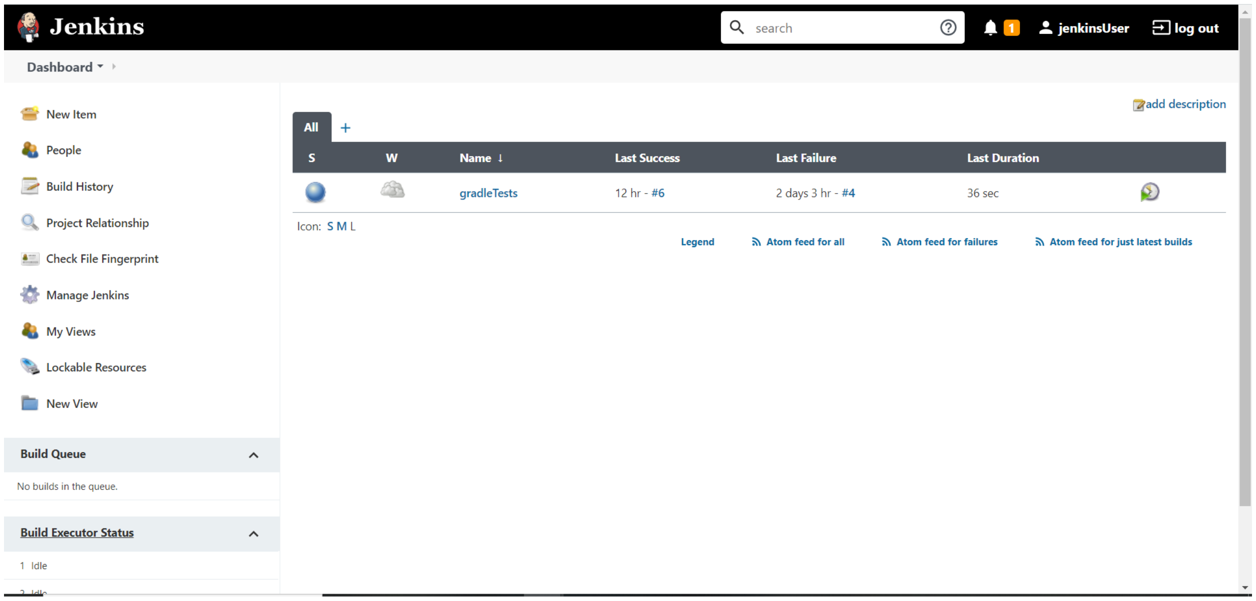
grayledw@rose-hulman.edu

Whitespace-separated list of recipient addresses. May reference build parameters like $PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.

[✓] Send e-mail for every unstable build

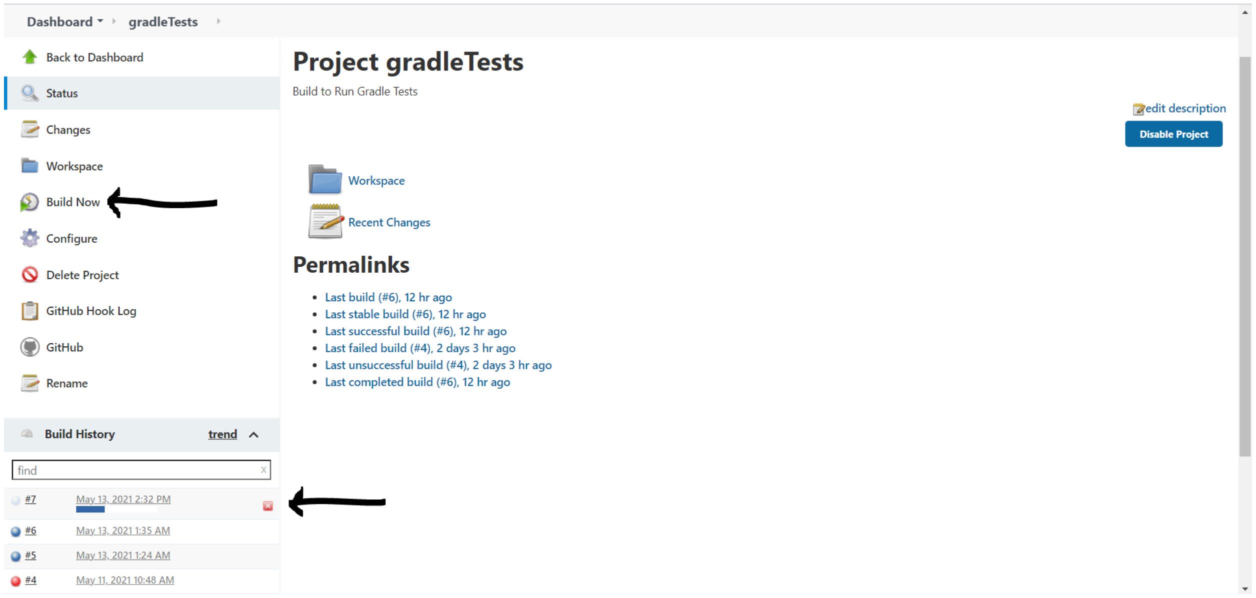
[ ] Send separate e-mails to individuals who broke the build

We have now fully configured our Jenkins job. Press “Save” and then navigate back to your Jenkins dashboard. Here you should be able to see your newly created job.



| S | W | Name ↓ | Last Success | Last Failure | Last Duration | |
|---|---|---|---|---|---|---|
| ● | | gradleTests | 12 hr - #6 | 2 days 3 hr - #4 | 36 sec | |

Click on your new job. This will now take you to a Dashboard for your job. Click “Build Now” and then on the right hand side of the screen you should see “Build History” with a new build. If the build appears as a blue/gray color it succeeded and if it is red then it has failed.



Dashboard ▾ › gradleTests ›

Back to Dashboard
Status
Changes
Workspace
Build Now
Configure
Delete Project
GitHub Hook Log
GitHub
Rename

Build History trend

find

#7 May 13, 2021 2:32 PM
#6 May 13, 2021 1:35 AM
#5 May 13, 2021 1:24 AM
#4 May 11, 2021 10:48 AM

**Project gradleTests**

Build to Run Gradle Tests

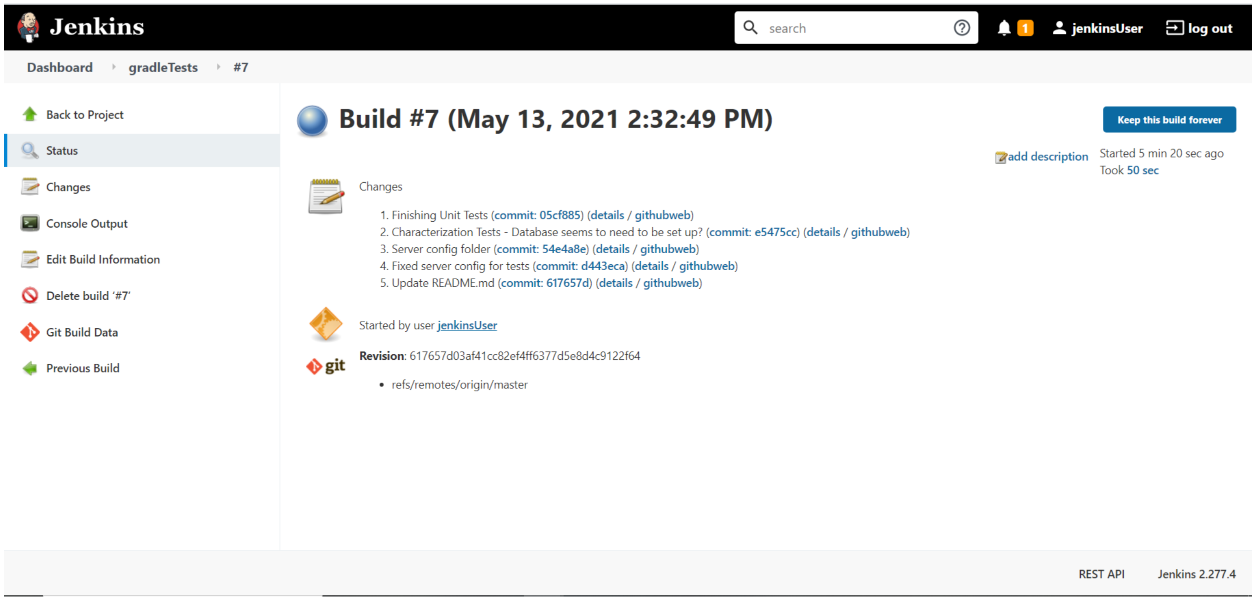
edit description
Disable Project

Workspace
Recent Changes

**Permalinks**

- Last build (#6), 12 hr ago
- Last stable build (#6), 12 hr ago
- Last successful build (#6), 12 hr ago
- Last failed build (#4), 2 days 3 hr ago
- Last unsuccessful build (#4), 2 days 3 hr ago
- Last completed build (#6), 12 hr ago

Once your Jenkins job has finished, you can view extra details on the results of the job build clicking on the build number. This will take you to a results screen shown below.



Jenkins search jenkinsUser log out

Dashboard › gradleTests › #7

Back to Project
Status
Changes
Console Output
Edit Build Information
Delete build '#7'
Git Build Data
Previous Build

**Build #7 (May 13, 2021 2:32:49 PM)**

Keep this build forever

add description

Started 5 min 20 sec ago
Took 50 sec

Changes

1. Finishing Unit Tests (commit: 05cf885) (details / githubweb)
2. Characterization Tests - Database seems to need to be set up? (commit: e5475cc) (details / githubweb)
3. Server config folder (commit: 54e4a8e) (details / githubweb)
4. Fixed server config for tests (commit: d443eca) (details / githubweb)
5. Update README.md (commit: 617657d) (details / githubweb)

Started by user jenkinsUser

**Revision**: 617657d03af41cc82ef4ff6377d5e8d4c9122f64

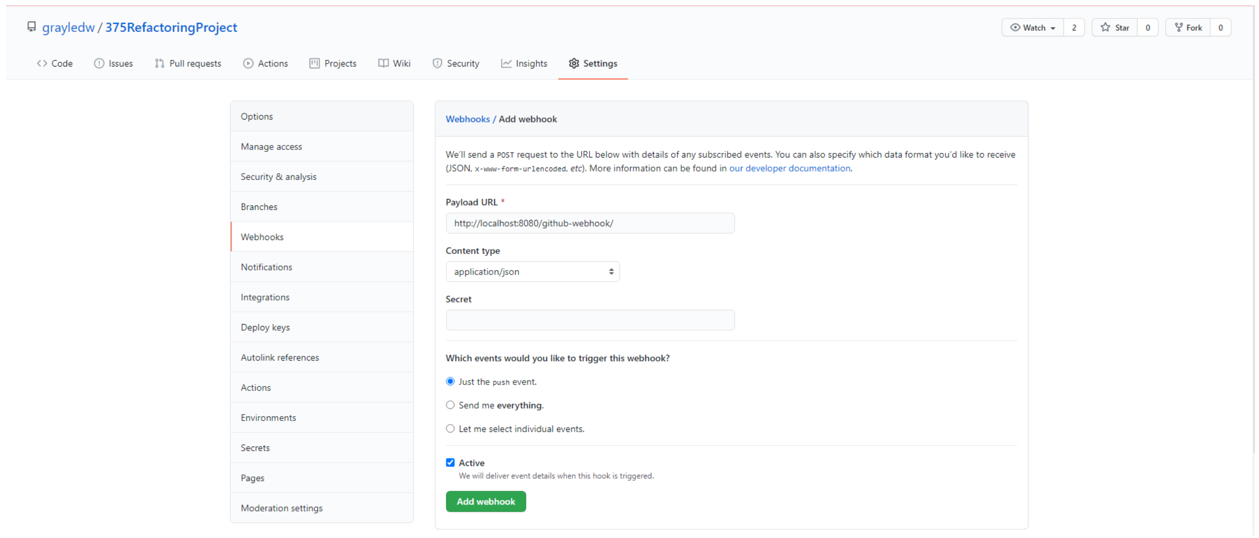
- refs/remotes/origin/master

REST API Jenkins 2.277.4

## Configuring GitHub To Trigger Jenkins Job When a Change to the Repository is Made

In this section we will configure GitHub to trigger our Jenkins build when a change is made to the repository. Unfortunately, this will not work because Jenkins is running locally and not available over the public internet, but these are the steps you would follow to configure this if Jenkins was hosted someplace available over the public internet.

First, go to the GitHub repository for the project, and then select “Settings”. Select “Webhooks” from the options on the right and then select “Add Webhook”. Use the settings provided below for your new webhook and then press “Add webhook.”



Webhooks / Add webhook

We'll send a `POST` request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *
http://localhost:8080/github-webhook/

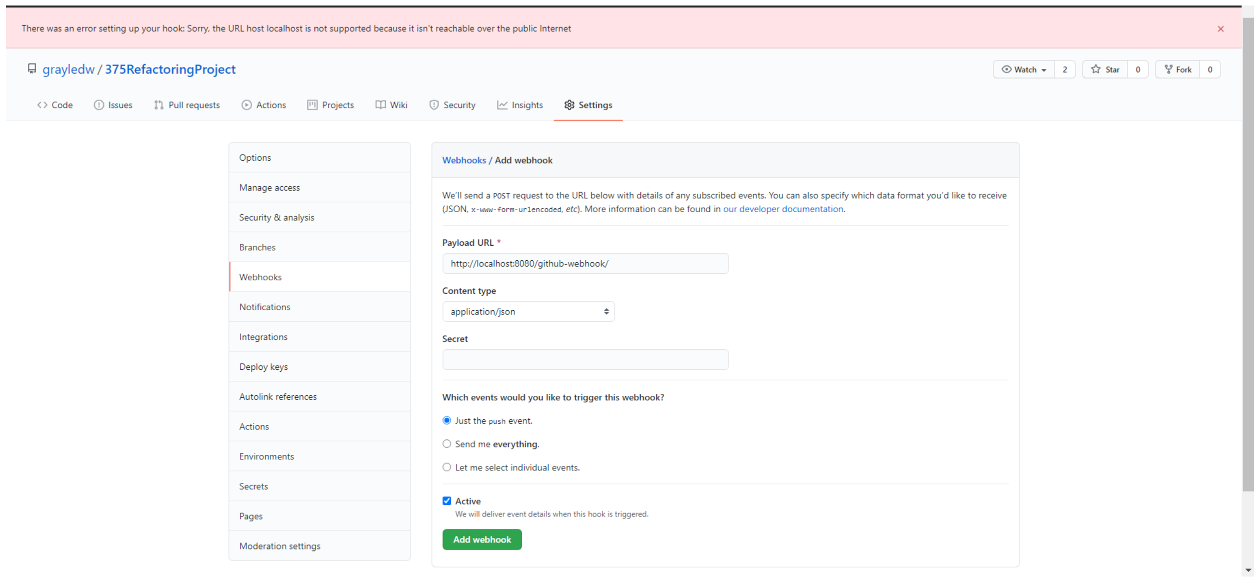
Content type
application/json

Secret

Which events would you like to trigger this webhook?
[● Just the `push` event.]
○ Send me **everything**.
○ Let me select individual events.

[✓ Active]
We will deliver event details when this hook is triggered.

After this runs, you will observe that the webhook is unable to be added. This fails due to Jenkins running locally and not being available over the public internet. Below is a screenshot of what this error looks like.



There was an error setting up your hook: Sorry, the URL host localhost is not supported because it isn't reachable over the public Internet

Webhooks / Add webhook

We'll send a `POST` request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *
http://localhost:8080/github-webhook/

Content type
application/json

Secret

Which events would you like to trigger this webhook?
[● Just the `push` event.]
○ Send me **everything**.
○ Let me select individual events.

[✓ Active]
We will deliver event details when this hook is triggered.

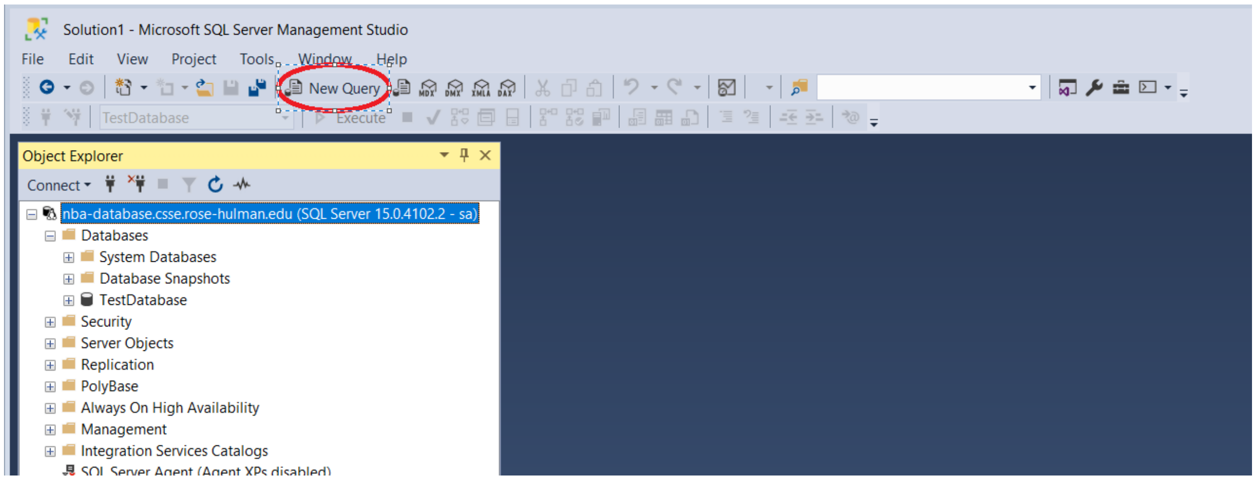
If you use your machine’s IP address instead the webhook will be created but when it is triggered it still will not be able to connect to Jenkins. If Jenkins was made available over the public internet then this step would succeed and our Jenkins jobs would be successfully triggered when a change to the repository is made.

## System Configuration

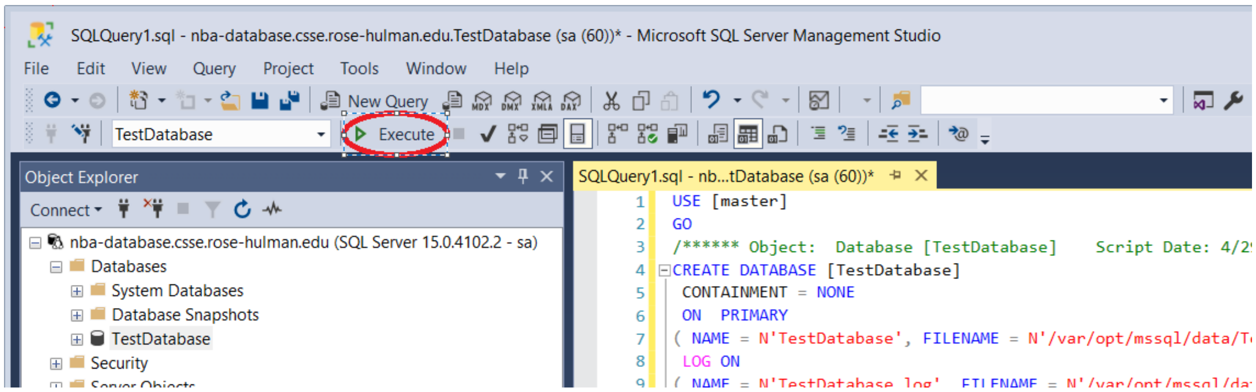
### Create new database on different server

By default, the program connects to the database created by our project team, running on a Rose-Hulman server. However if you would like to create your own database on a server you own, the steps below walk you through that process.

You need to first set up a SQL database using a server of your choice.
See here for steps on how to set up an apache server to run the SQL database. The decision is up to you, but once you have the server up and running, you'll need to configure it properly using a server manager so that it can properly run the database.
We recommend that you use Microsoft SQL Server Management Studio, as that is what the following images are depicting and it is what is used to access the current server. What you'll need to do is click on 'New Query' at the top:


Solution1 - Microsoft SQL Server Management Studio
File
Edit
View
Project
Tools
Window
Help
New Query
TestDatabase
Execute
Object Explorer
Connect
nba-database.csse.rose-hulman.edu (SQL Server 15.0.4102.2 - sa)
Databases
System Databases
Database Snapshots
TestDatabase
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)

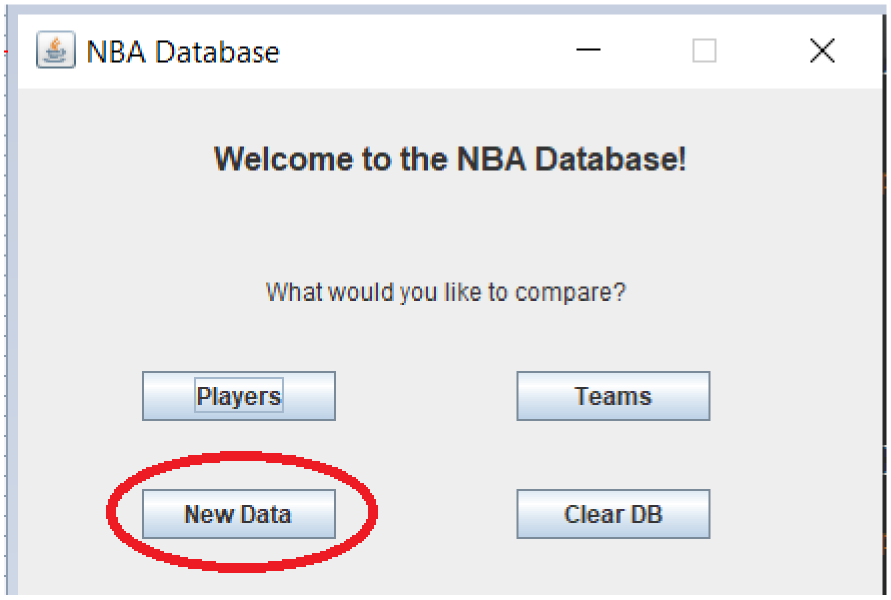
Within the query window, copy the contents of the CreationScript.sql file that we have provided to you, and paste it in the text area that appears:


SQLQuery1.sql - nba-database.csse.rose-hulman.edu.TestDatabase (sa (60))* - Microsoft SQL Server Management Studio
File
Edit
View
Query
Project
Tools
Window
Help
New Query
TestDatabase
Execute
Object Explorer
Connect
nba-database.csse.rose-hulman.edu (SQL Server 15.0.4102.2 - sa)
Databases
System Databases
Database Snapshots
TestDatabase
Security
Server Objects
SQLQuery1.sql - nb...tDatabase (sa (60))*

```
USE [master]
GO
/****** Object:  Database [TestDatabase]    Script Date: 4/2
CREATE DATABASE [TestDatabase]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'TestDatabase', FILENAME = N'/var/opt/mssql/data/T
 LOG ON
( NAME = N'TestDatabase_log', FILENAME = N'/var/opt/mssql/da
```

Then click ‘Execute’. This creates all the database tables, rules, views, and stored procedures so that the program can be properly configured with the database. Once this is complete, your server should now have a database called ‘TestDatabase’, which includes everything needed for our system.

**NOTE**: If you create your own database using a new server, you’ll need to populate the database using the ‘New Data’ button on the home page of our application:



#heading=h.5kir7tr8ihqy
Instructions for this step are described further in detail in the new data section of the user guide.

## Configure server name, username, and password

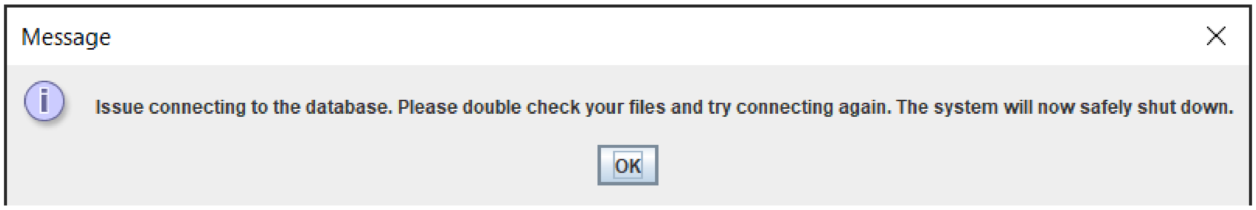
If you decide to use your own server with a different username and password, you will need to specify that in our system. We have made this process easy by allowing you to edit the config files so that you can directly put in the correct server name, username and password. **Note**: The database name created by the creation script is always going to be the same, so there is no way to change that name unless you would like to do so on your end, which we **strongly** advise against.

**To change the server name:** Edit the server.txt file located within the main directory in the project. Do not use spaces or quotes. Make sure that you have pasted the exact same name as your server into the text file.

**To change the username:** Edit the username.txt file located within the main directory in the project. Do not use spaces or quotes. Make sure that you have pasted the exact same name as your username for your server into the text file.

**To change the password:** Edit the password.txt file located within the main directory in the project. Do not use spaces or quotes. Make sure that you have pasted the <u>exact same name</u> as your password for your server into the text file.

No further changes are required beyond this point to configure your own server. The program handles reading in these values and connecting to the database. If the system is not able to establish a proper connection to the database that you specified, it will throw the following error:



Message

Issue connecting to the database. Please double check your files and try connecting again. The system will now safely shut down.

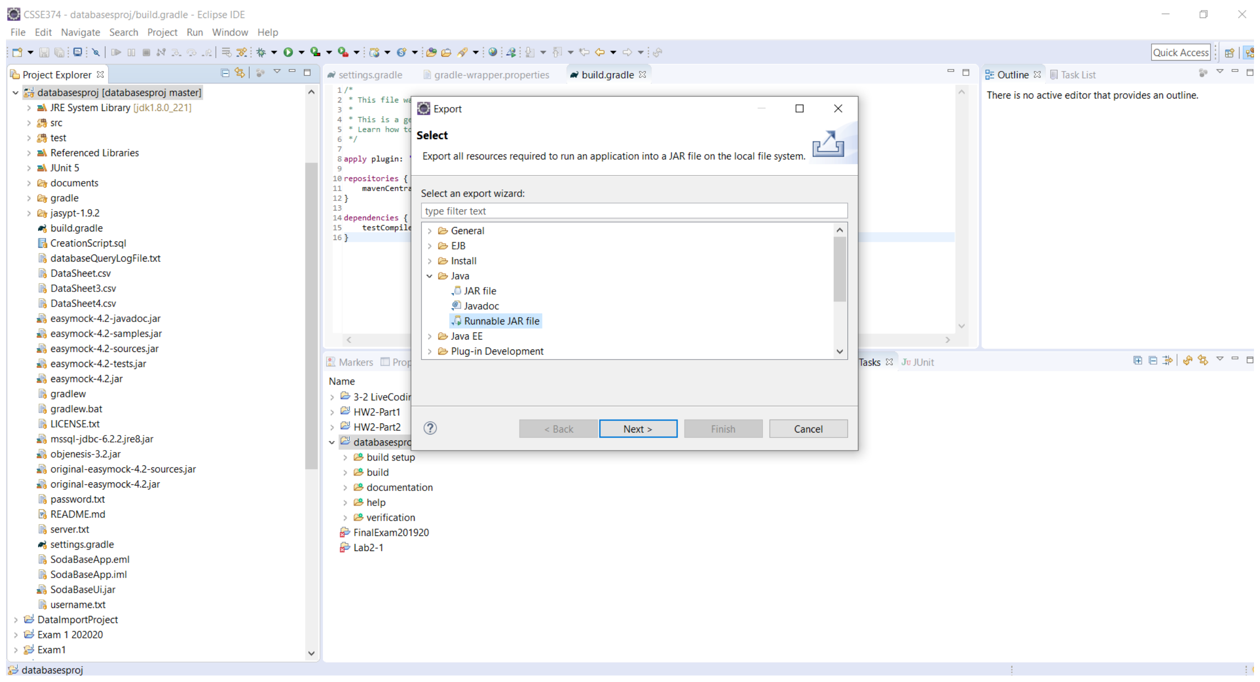
OK

This tells the user that the information they put in for connecting to the database is faulty, and the system is going to shut down so the user can put in the proper information. Beyond this, you will need to double check your information in the server.txt, username.txt, and password.txt files to make sure they are correctly specified.

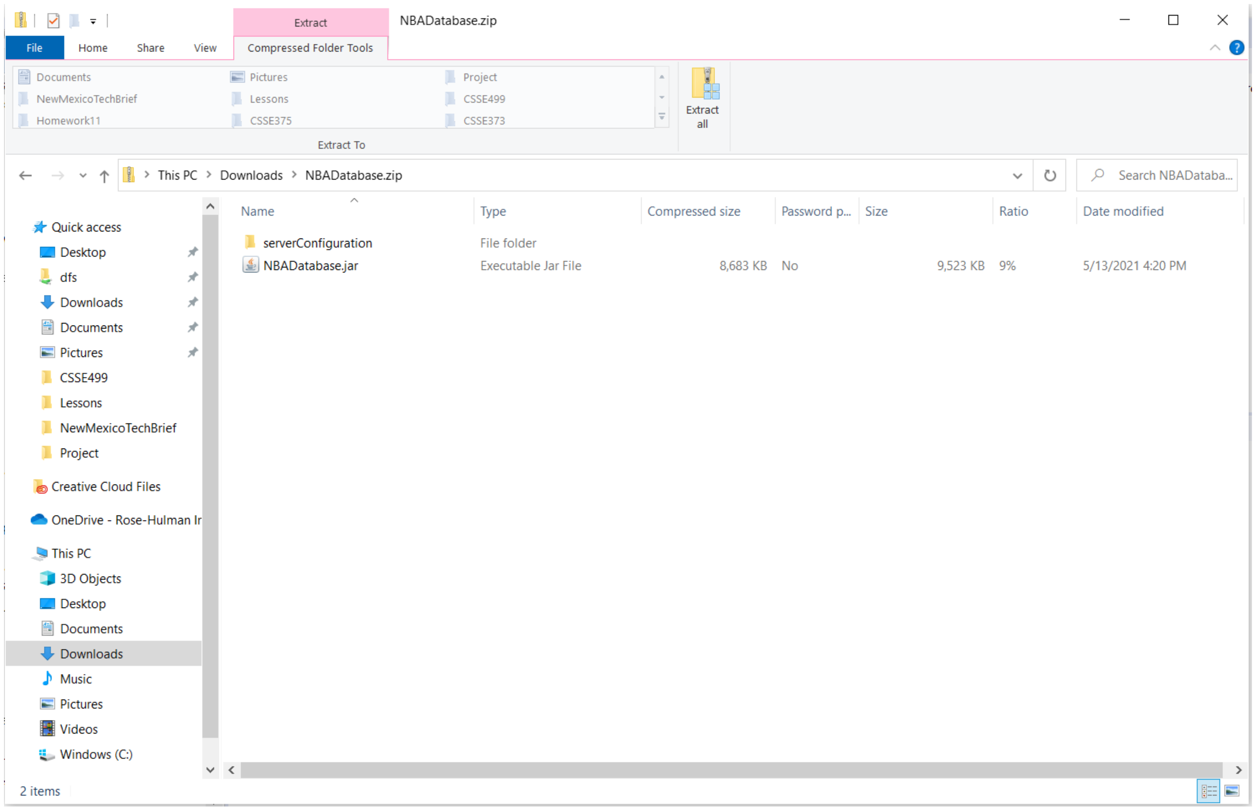
## System Maintenance

### Releasing a New Version of the System

In order to release a new version of the system, first a new .jar file must be generated. To do this, right click on the project in Eclipse, click Export, and then select “Runnable JAR File”.



Next, provide a location to save the JAR file. Use the name "NBADatabase.jar". Next the JAR file must be packaged with the serverConfiguration folder in order to run properly. Locate the newly created JAR file and ZIP it along with the serverConfiguration folder so that the serverConfiguration folder is at the same level as the JAR file. The ZIP should look the same as below:

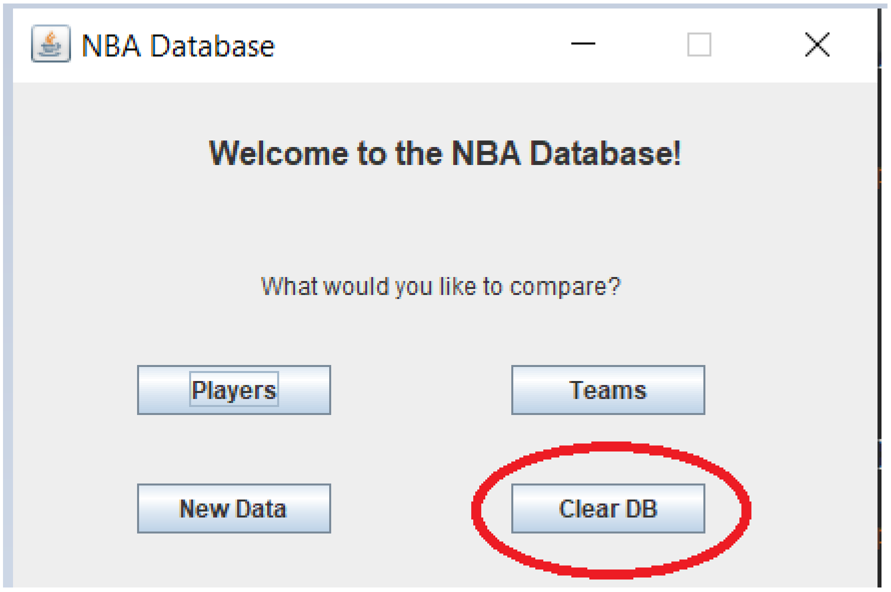


| Name | Type | Compressed size | Password p... | Size | Ratio | Date modified |
|---|---|---|---|---|---|---|
| serverConfiguration | File folder | | | | | |
| NBADatabase.jar | Executable Jar File | 8,683 KB | No | 9,523 KB | 9% | 5/13/2021 4:20 PM |

Finally, upload the newest version of the project to Moodle.

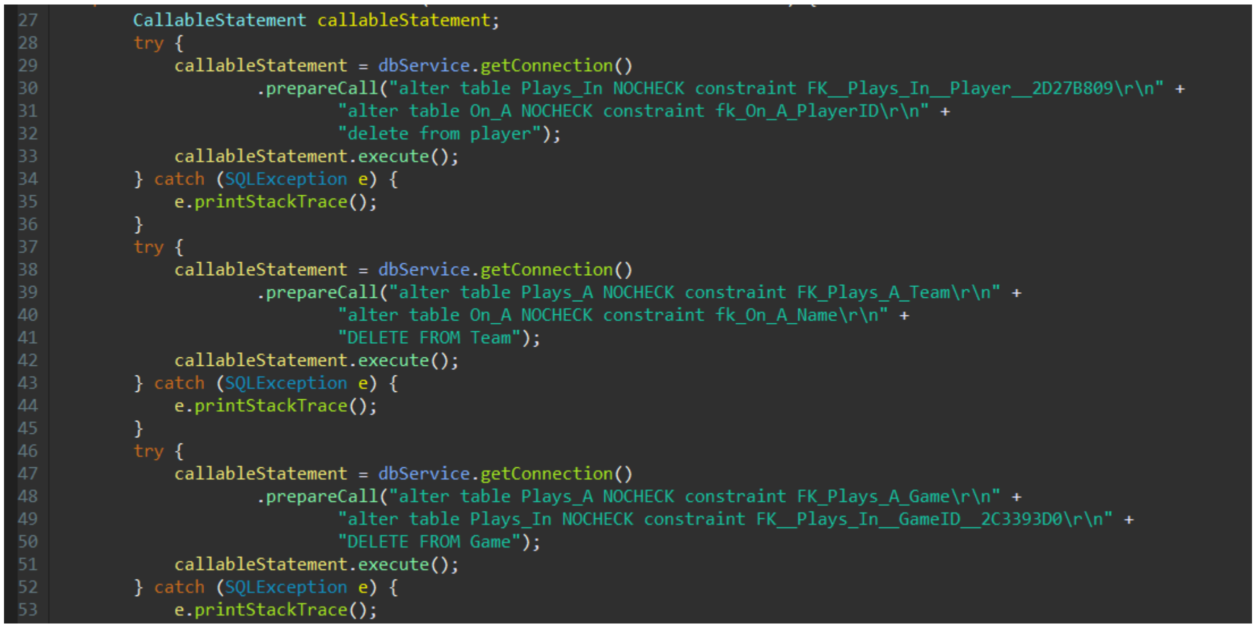
## Database Maintenance

We have implemented a feature in our system that clears and rebuilds the database in case it gets too large and you would like to delete all of the data or start over with data generated by you. All you need to do is click "Clear DB" on the homepage of the application, and it automatically clears all of the data in the database for you:



What this actually does is a couple things (this is also within the **RebuildDatabase.java** class):

1. Disables all foreign key constraints between tables so that the data doesn't cause issues with other tables when deleted:



```java
        CallableStatement callableStatement;
        try {
            callableStatement = dbService.getConnection()
                    .prepareCall("alter table Plays_In NOCHECK constraint FK__Plays_In__Player__2D27B809\r\n" +
                            "alter table On_A NOCHECK constraint fk_On_A_PlayerID\r\n" +
                            "delete from player");
            callableStatement.execute();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        try {
            callableStatement = dbService.getConnection()
                    .prepareCall("alter table Plays_A NOCHECK constraint FK_Plays_A_Team\r\n" +
                            "alter table On_A NOCHECK constraint fk_On_A_Name\r\n" +
                            "DELETE FROM Team");
            callableStatement.execute();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        try {
            callableStatement = dbService.getConnection()
                    .prepareCall("alter table Plays_A NOCHECK constraint FK_Plays_A_Game\r\n" +
                            "alter table Plays_In NOCHECK constraint FK__Plays_In__GameID__2C3393D0\r\n" +
                            "DELETE FROM Game");
            callableStatement.execute();
        } catch (SQLException e) {
            e.printStackTrace();
```

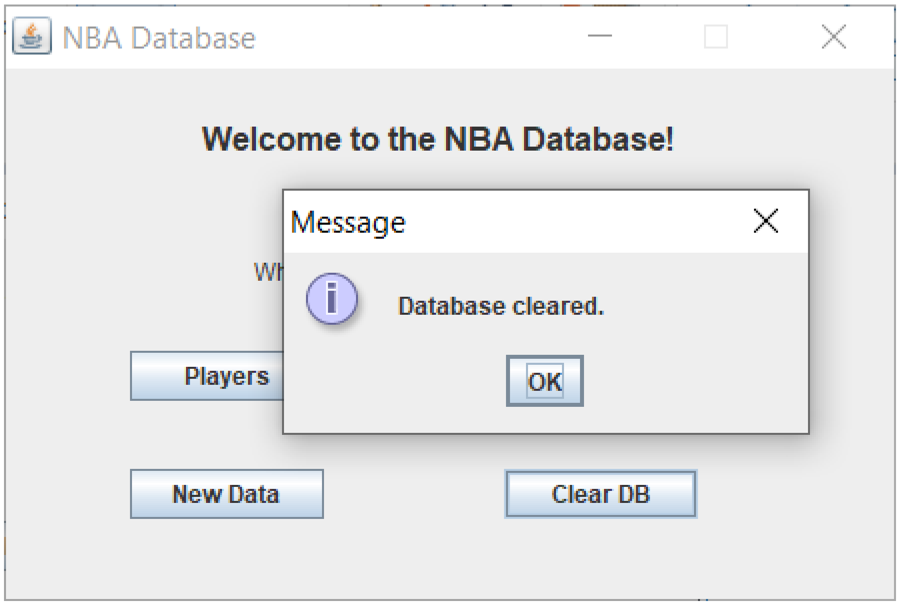
2. Deletes all the data rows in each table:

```java
        try {
            callableStatement = dbService.getConnection()
                    .prepareCall("DELETE FROM On_A");
            callableStatement.execute();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        try {
            callableStatement = dbService.getConnection()
                    .prepareCall("Delete FROM Plays_A");
            callableStatement.execute();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        try {
            callableStatement = dbService.getConnection()
                    .prepareCall("DELETE FROM Plays_In");
            callableStatement.execute();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        try {
            callableStatement = dbService.getConnection()
                    .prepareCall("DELETE FROM Season");
            callableStatement.execute();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

3. Re-enables all the foreign key constraints between tables:

```java
public static void reEnableConstraints(DatabaseConnectionService dbService) {
    CallableStatement callableStatement;
    try {
        callableStatement = dbService.getConnection()
                .prepareCall("alter table Plays_In CHECK constraint FK__Plays_In__Player__2D27B809\r\n" +
                        "alter table On_A CHECK constraint fk_On_A_PlayerID\r\n" +
                        "alter table Plays_A CHECK constraint FK_Plays_A_Team\r\n" +
                        "alter table On_A CHECK constraint fk_On_A_Name\r\n" +
                        "alter table Plays_A CHECK constraint FK_Plays_A_Game\r\n" +
                        "alter table Plays_In CHECK constraint FK__Plays_In__GameID__2C3393D0");
        callableStatement.execute();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Upon successfully clearing the database, you will be shown the following popup denoting that the database has been cleared:

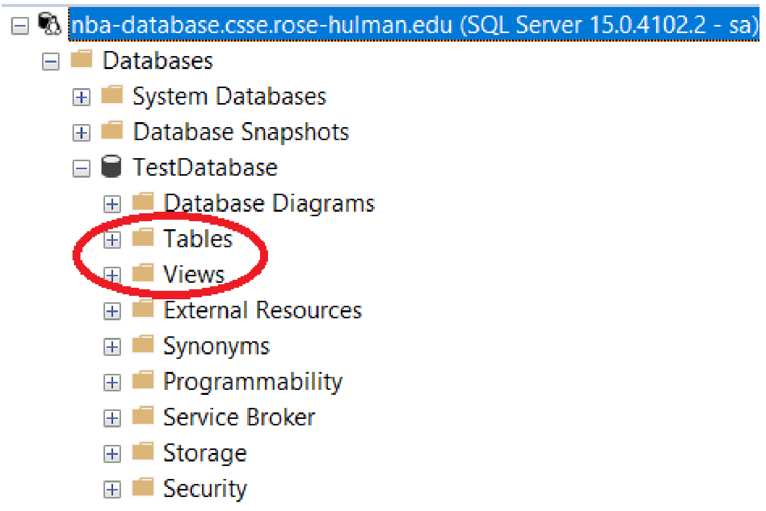


Database cleared.

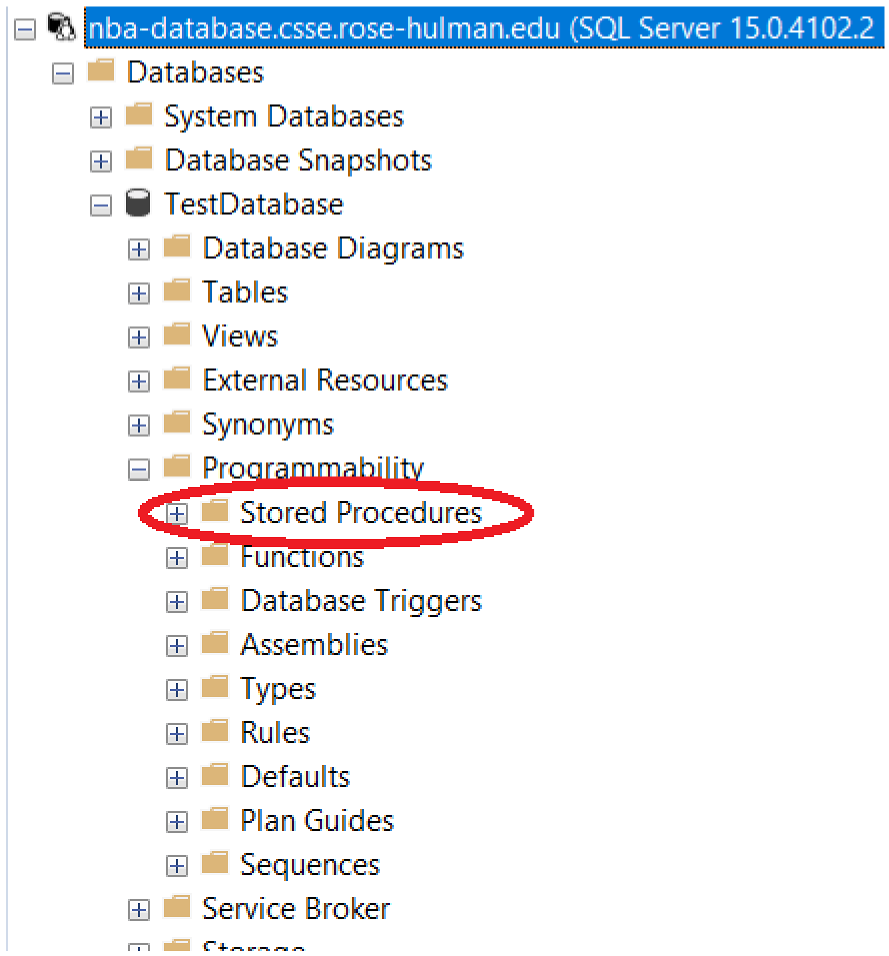
## Database Procedures, Views, and Tables

Note: We **very strongly** advise against any further changes to the database than what is provided to you by our system. Other changes could mess up how other parts of the system query the database and could lead to unexpected errors down the road. Feel free to view any of the data. **However, changes to the database tables, views, or stored procedures will almost certainly lead to the application failing down the road**; so it is best to just not mess with the database at all. If you would like to view any of the information without editing it, you can find that information in your specific server manager.

Below, you can find the tables and views used in our system.



If you would like to view the stored procedures used, they can be found below:



## Software Requirements Specification

Below are the requirements developed for our system. These cover how the system will respond to specific user actions made while interacting with the system's UI. Additionally, it covers how the system responds to actions the users take which are invalid or inoperative for the system.

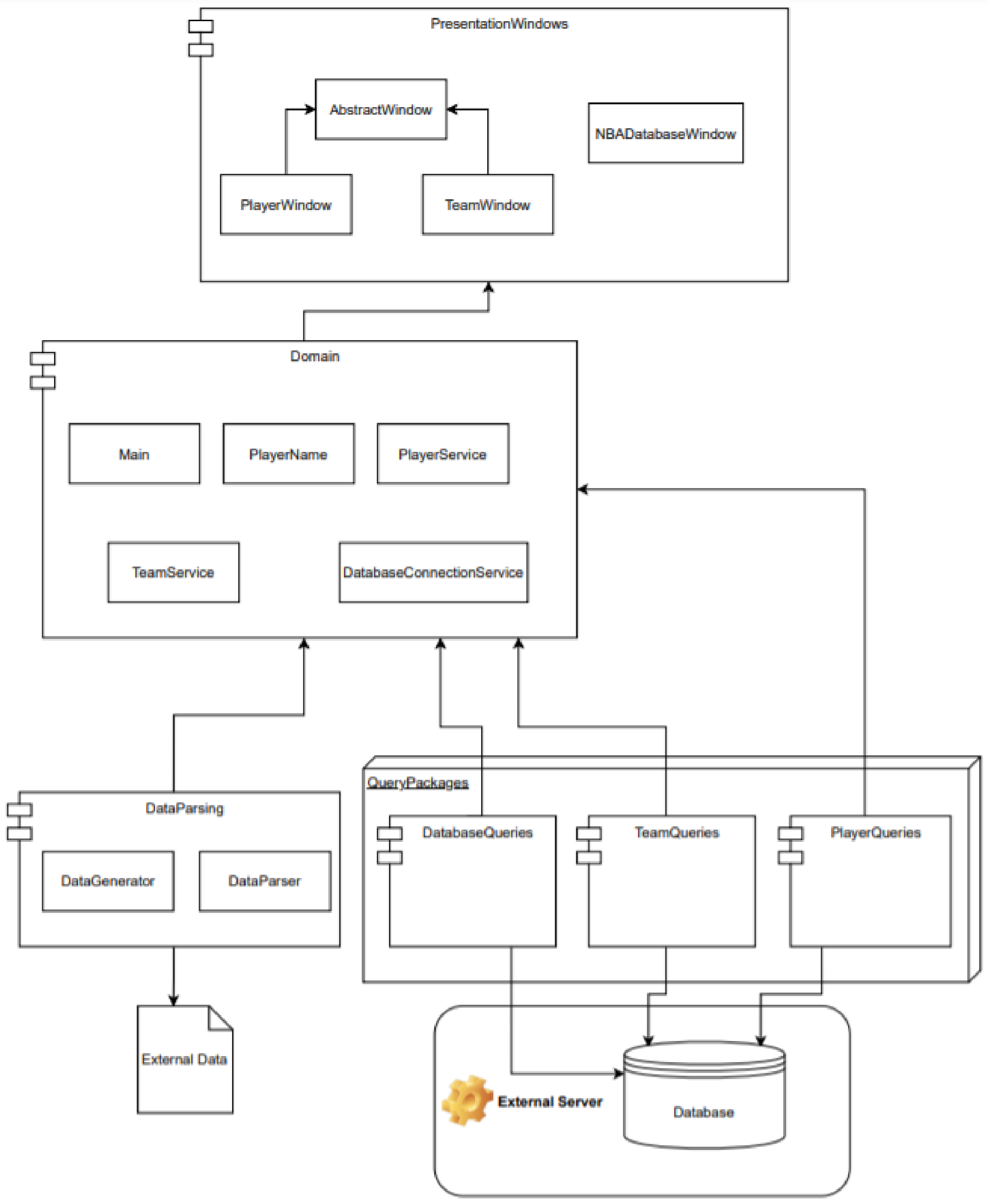
- The system shall allow users to insert new data they generate into the database
- The system shall allow users to clear the database they are currently using
- The system shall allow users to select whether they want to compare team or player data
- The system shall allows users to enter the player name of the player they wish to view data for
- The system shall allow users to specify what kind of data they wish to view for a specific player; Game data, Season data, or Career (Overall) Data.
- The system shall allow users to specify what year they would like to view Season or Game data for a specified player
- The system shall notify users when they have input an invalid name for a specific player

- The system shall allows users to enter the team name of the team they wish to view data for
- The system shall allow users to specify what kind of data they wish to view for a specific team; Game data, Season data, or Franchise (Overall) Data.
- The system shall allow users to specify what year they would like to view Season or Game data for a team
- The system shall notify users when they have input an invalid name for a specific team
- The system shall implement automated tests such that every time new code is pushed to master branch, the tests verify that all previous functionality is maintained

## Software and Architecture Design Specification

Below is the architecture we developed for our system. We designed our system to roughly follow the 3 principal layers (Presentation, Domain, Data Source). However, we really have two separate “Data Source layers”, one for interactions with the database and the other for interactions with external sheets. We felt this design decision was necessary since the database queries were pretty extensive and also very unrelated to the code which interacted with the external sheets used for new data.

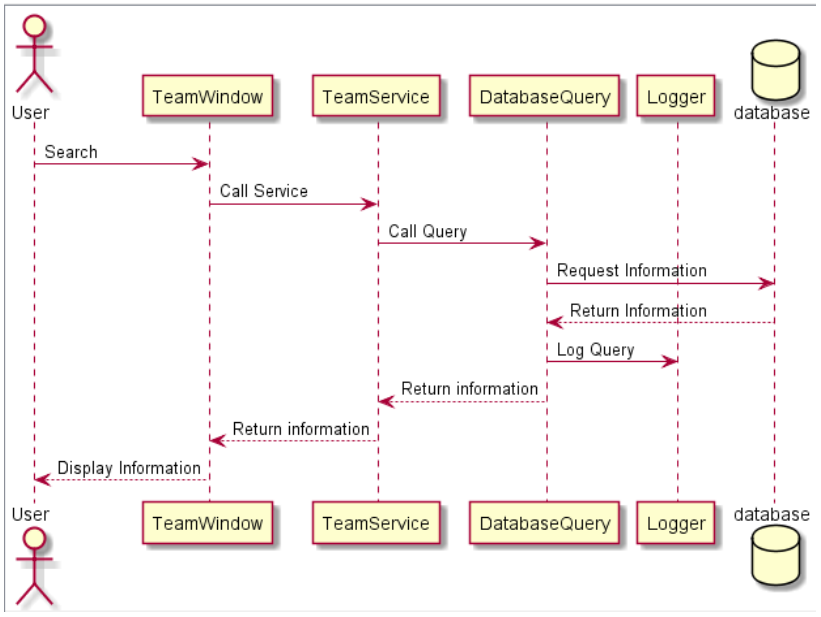
## Top Level Design



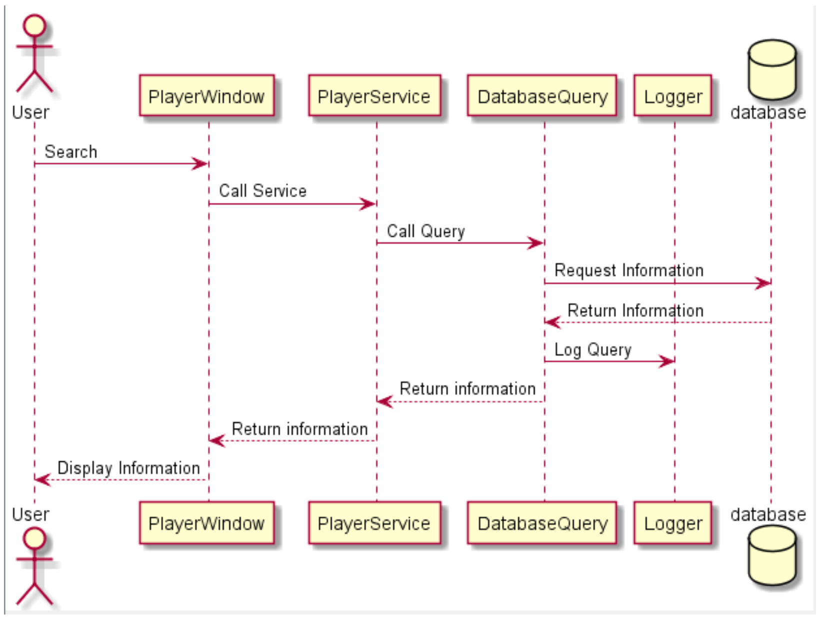
## Sequence Diagrams

The following are Sequence Diagrams for several common user tasks

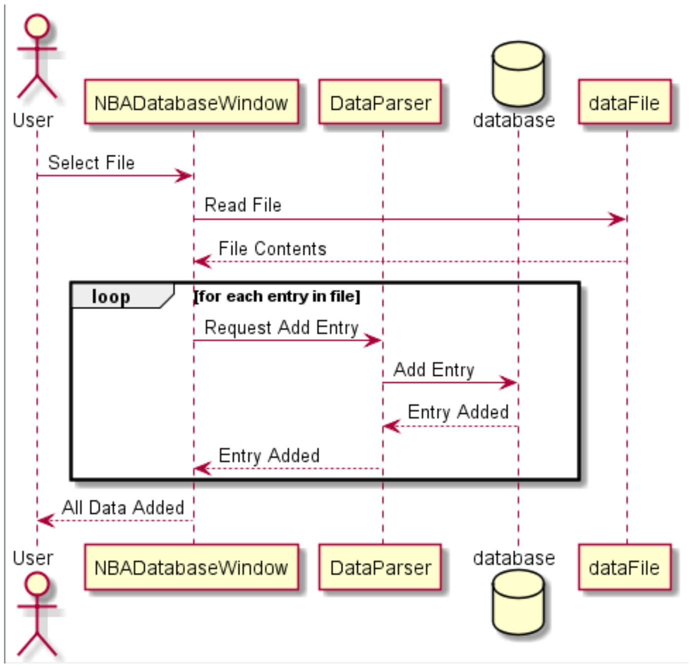
## Request Team Information



## Request Player Information

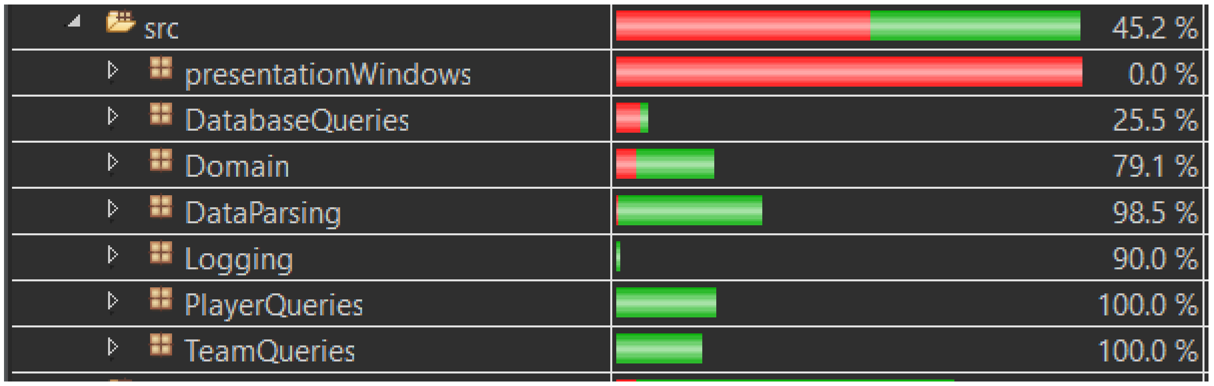


### Add Data



## Testing Plan/Strategy

### Testing Strategy



| | |
|---|---|
| src | 45.2 % |
| presentationWindows | 0.0 % |
| DatabaseQueries | 25.5 % |
| Domain | 79.1 % |
| DataParsing | 98.5 % |
| Logging | 90.0 % |
| PlayerQueries | 100.0 % |
| TeamQueries | 100.0 % |

Our project is a database project and as such, the focus of our testing effort is on the domain and database layers. As you can see, we haven't written any tests for the UI of our project but have tried to maximize coverage on the systems which touch the database. When writing unit tests, our goal was to ensure that queries are being constructed correctly before being executed and that the sets of results are being handled properly and brought back up to the presentation layer. As for characterization tests, our goal is to confirm that the core functionality of the database is working by calling the same methods which the presentation layer would call, not mocking anything, and confirming that the results come back as expected.

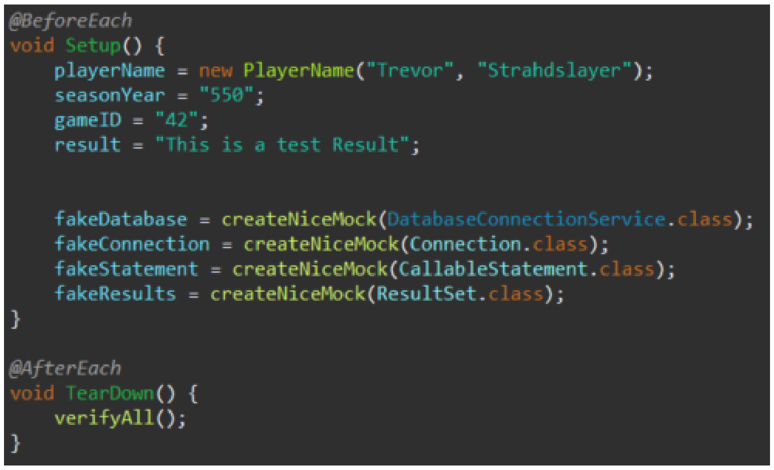
| | |
|---|---|
| Unit Tests | **DatabaseConnectionServiceTests**<br>TestConnect<br>**DatabaseQueryLoggerTests**<br>TestLogging<br>**DataParserTests**<br>testInsertNextLineToDB<br>**TeamQueryTests**<br>testFranchiseDataGetResults, testGamesDataGetResults, testGamesPlayedDataGetResults, testSeasonDataGetResults, testSeasonsPlayedDataGetResults, testTeamFranchiseDataQuery, TestTeamGameDataQuery, testTeamGamesPlayedDataQuery, testTeamSeasonDataQuery, testTeamSeasonsPlayedDataQuery<br>**PlayerQueryTests**<br>testCareerDataGetResults, testGamesDataGetResults, testGamesPlayedGetResults, testPlayerGameDataQuery, testPlayerGamesPlayedInSeasonDataQuery, testPlayerSeasonDataQuery, testPlayerSeasonsPlayedDataQuery, testSeasonDataGetResults, testSeasonsPlayedGetResults |
| Functional Tests | **PlayerServiceFunctionalTests**<br>TestGetCareerInfo, TestGetGameInfo, TestGetPlayerCareerInfo, TestGetPlayerGamesPlayedInfo, TestGetPlayerSeasonsPlayedInfo, TestGetSeasonInfo<br>**DataGeneratorTests**<br>testGeneratedFileAgainstMasterCopy<br>**TeamServiceFunctionalTests**<br>TestGetTeamFranchiseInfo, TestGetTeamGameInfo, TestGetTeamPlayedInfo, TestGetTeamSeasonInfo, TestGetTeamSeasonsPlayedInfo |
| Integration Tests | **PlayerServiceTests**<br>testGetPlayerInformationCareer, testGetPlayerInformationGame, testGetPlayerInformationSeason<br>**TeamServiceTests**<br>testGetPlayerInformationCareer, testGetPlayerInformationGame, testGetPlayerInformationSeason |

## Unit Tests

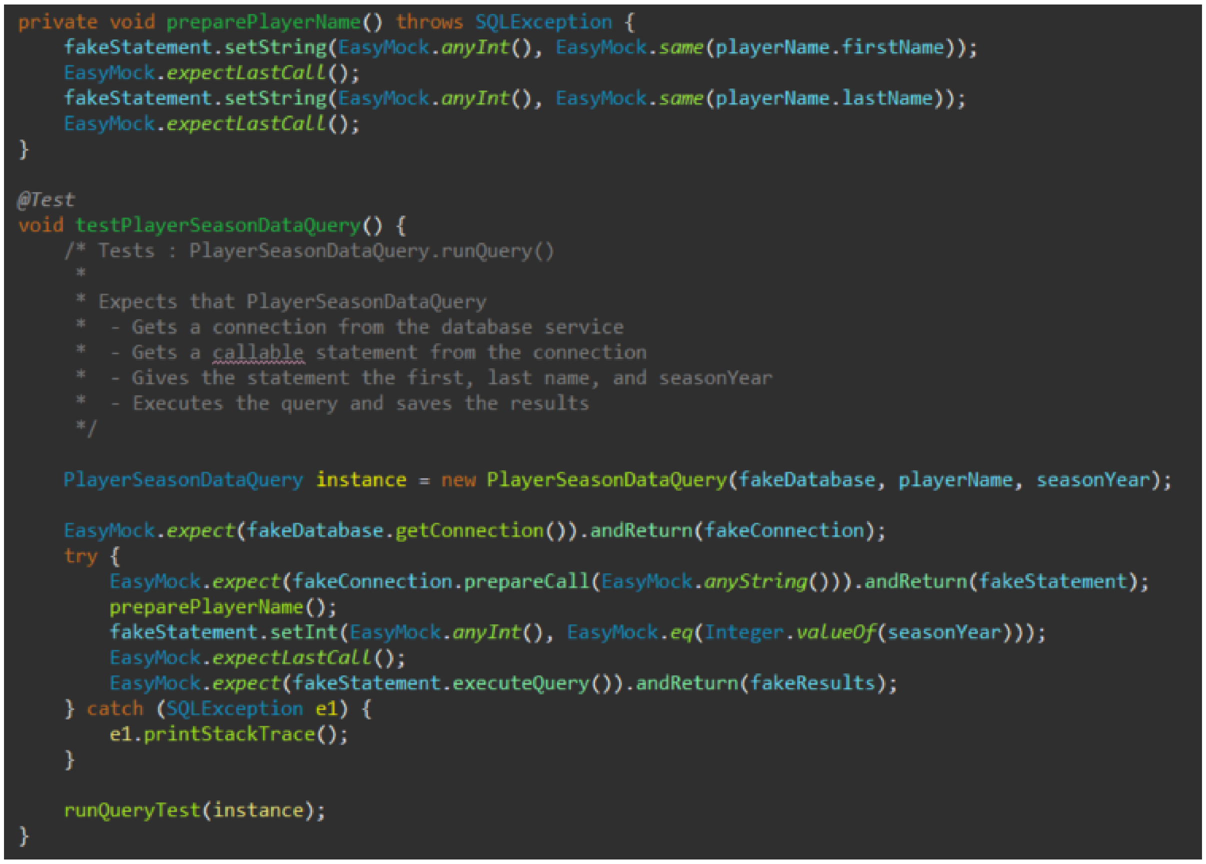
### Query Tests

Our Query tests also all follow a similar structure to each other. We begin by constructing mocks of all the database connection objects.
In the individual tests, the query object in question is constructed.
Depending on the query, different data is expected to be added to the mocked database connection objects.

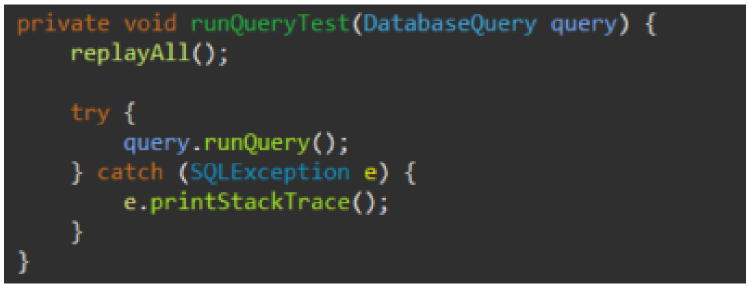
We also expect that the query returns the appropriate results which it gets from those mocked database objects.



```java
@BeforeEach
void Setup() {
    playerName = new PlayerName("Trevor", "Strahdslayer");
    seasonYear = "550";
    gameID = "42";
    result = "This is a test Result";


    fakeDatabase = createNiceMock(DatabaseConnectionService.class);
    fakeConnection = createNiceMock(Connection.class);
    fakeStatement = createNiceMock(CallableStatement.class);
    fakeResults = createNiceMock(ResultSet.class);
}

@AfterEach
void TearDown() {
    verifyAll();
}
```



```java
private void preparePlayerName() throws SQLException {
    fakeStatement.setString(EasyMock.anyInt(), EasyMock.same(playerName.firstName));
    EasyMock.expectLastCall();
    fakeStatement.setString(EasyMock.anyInt(), EasyMock.same(playerName.lastName));
    EasyMock.expectLastCall();
}

@Test
void testPlayerSeasonDataQuery() {
    /* Tests : PlayerSeasonDataQuery.runQuery()
     *
     * Expects that PlayerSeasonDataQuery
     *  - Gets a connection from the database service
     *  - Gets a callable statement from the connection
     *  - Gives the statement the first, last name, and seasonYear
     *  - Executes the query and saves the results
     */

    PlayerSeasonDataQuery instance = new PlayerSeasonDataQuery(fakeDatabase, playerName, seasonYear);

    EasyMock.expect(fakeDatabase.getConnection()).andReturn(fakeConnection);
    try {
        EasyMock.expect(fakeConnection.prepareCall(EasyMock.anyString())).andReturn(fakeStatement);
        preparePlayerName();
        fakeStatement.setInt(EasyMock.anyInt(), EasyMock.eq(Integer.valueOf(seasonYear)));
        EasyMock.expectLastCall();
        EasyMock.expect(fakeStatement.executeQuery()).andReturn(fakeResults);
    } catch (SQLException e1) {
        e1.printStackTrace();
    }

    runQueryTest(instance);
}
```

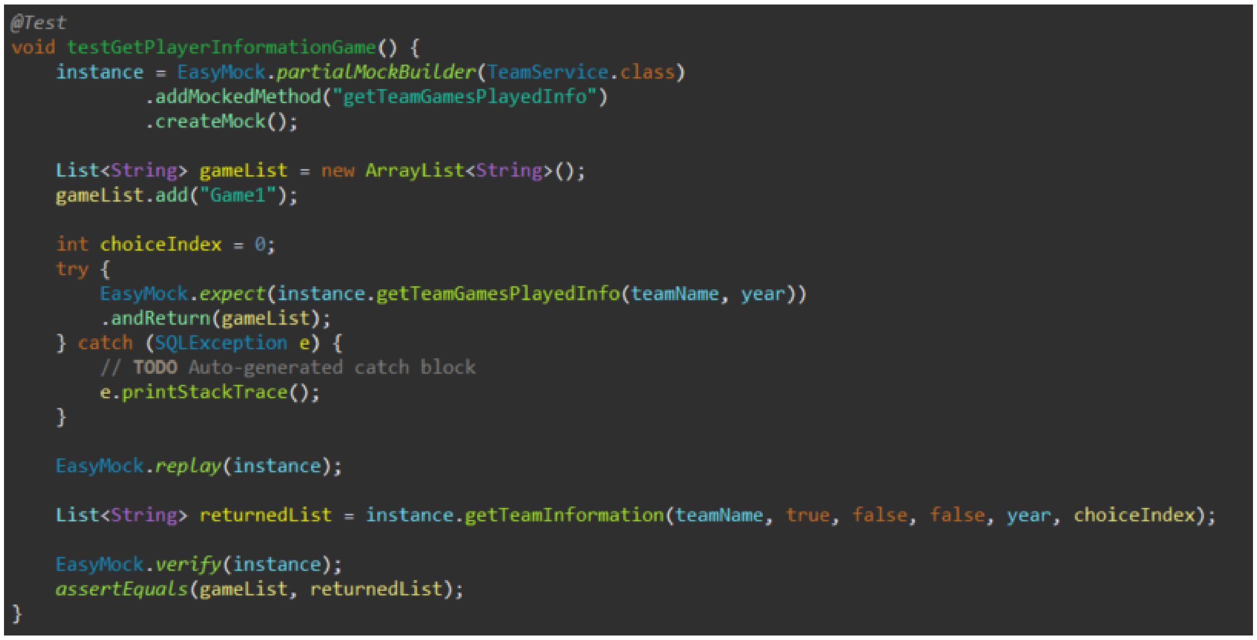


```java
private void runQueryTest(DatabaseQuery query) {
    replayAll();

    try {
        query.runQuery();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

## Integration Tests

### Service Tests

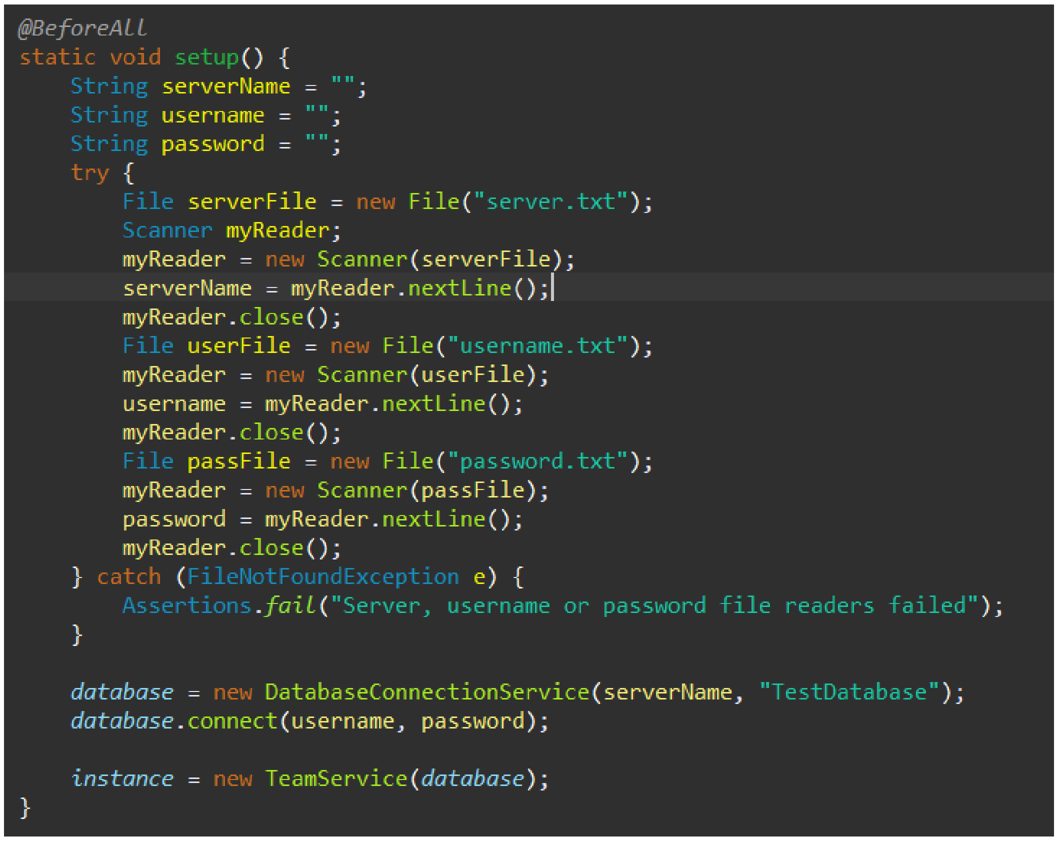
Our Service tests all follow the same general structure. We create a partial mock of the service object, mock the method which constructs the appropriate query and confirm that it is called. We expect that the data returned by the query is also returned by the service.



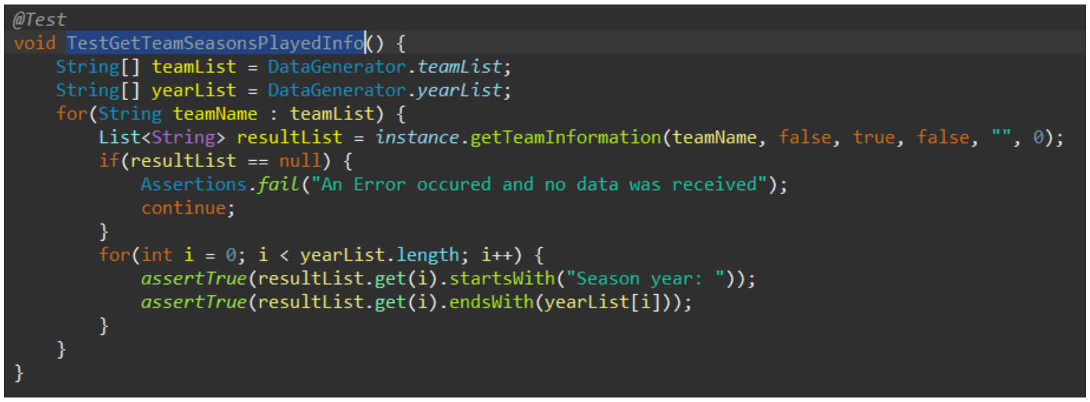
```java
@Test
void testGetPlayerInformationGame() {
    instance = EasyMock.partialMockBuilder(TeamService.class)
            .addMockedMethod("getTeamGamesPlayedInfo")
            .createMock();

    List<String> gameList = new ArrayList<String>();
    gameList.add("Game1");

    int choiceIndex = 0;
    try {
        EasyMock.expect(instance.getTeamGamesPlayedInfo(teamName, year))
        .andReturn(gameList);
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    EasyMock.replay(instance);

    List<String> returnedList = instance.getTeamInformation(teamName, true, false, false, year, choiceIndex);

    EasyMock.verify(instance);
    assertEquals(gameList, returnedList);
}
```

## Functional Tests

The functional tests all begin by setting up a valid database connection the same way the project would in the main method. Each functional test focuses on one or more methods from a Service class. These are the methods that the UI calls when a button is pressed and the remaining UI elements mostly determine which Service method and with what parameters are you calling that method. As such, we can cover most of the project's functionality through this type of test.
In each test, we loop through all the valid inputs to this service method as though a user was trying every single option in the UI. Since much of the data is randomly generated, we are not looking for specific data to be returned, merely that the data makes sense for the given service method.



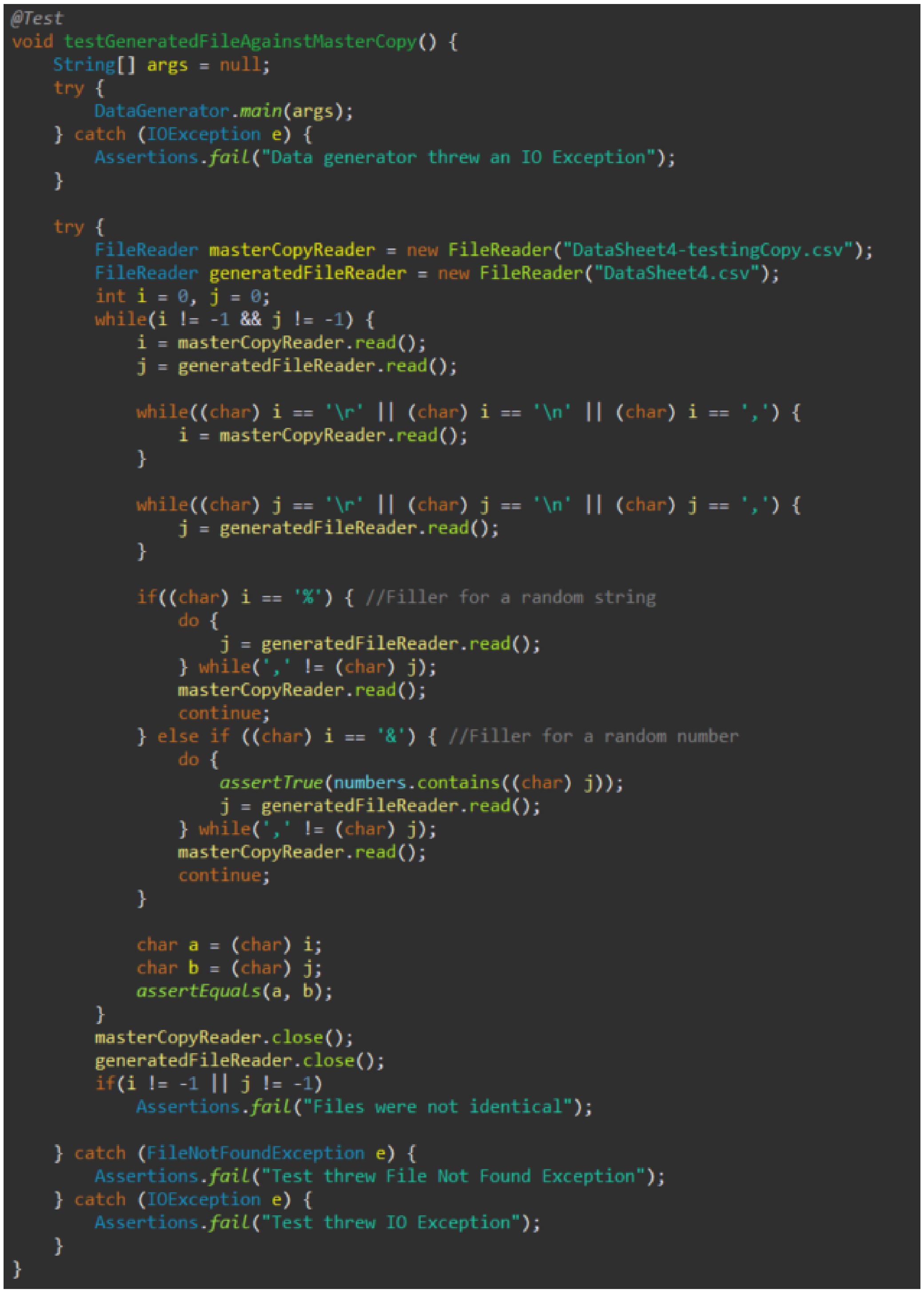
```
@BeforeAll
static void setup() {
    String serverName = "";
    String username = "";
    String password = "";
    try {
        File serverFile = new File("server.txt");
        Scanner myReader;
        myReader = new Scanner(serverFile);
        serverName = myReader.nextLine();
        myReader.close();
        File userFile = new File("username.txt");
        myReader = new Scanner(userFile);
        username = myReader.nextLine();
        myReader.close();
        File passFile = new File("password.txt");
        myReader = new Scanner(passFile);
        password = myReader.nextLine();
        myReader.close();
    } catch (FileNotFoundException e) {
        Assertions.fail("Server, username or password file readers failed");
    }

    database = new DatabaseConnectionService(serverName, "TestDatabase");
    database.connect(username, password);

    instance = new TeamService(database);
}
```



```
@Test
void TestGetTeamSeasonsPlayedInfo() {
    String[] teamList = DataGenerator.teamList;
    String[] yearList = DataGenerator.yearList;
    for(String teamName : teamList) {
        List<String> resultList = instance.getTeamInformation(teamName, false, true, false, "", 0);
        if(resultList == null) {
            Assertions.fail("An Error occured and no data was received");
            continue;
        }
        for(int i = 0; i < yearList.length; i++) {
            assertTrue(resultList.get(i).startsWith("Season year: "));
            assertTrue(resultList.get(i).endsWith(yearList[i]));
        }
    }
}
```

## Data Generator Test

The DataGeneratorTest is for ensuring that valid sample data is created by the DataGenerator class. It prints this data to a .csv file to then be manually inserted to the database. We have a master version of this datasheet that we test the generated version against. Some of this data is randomized so the ‘&’ character represents a random number and the ‘%’ character represents a random string.



```
@Test
void testGeneratedFileAgainstMasterCopy() {
    String[] args = null;
    try {
        DataGenerator.main(args);
    } catch (IOException e) {
        Assertions.fail("Data generator threw an IO Exception");
    }

    try {
        FileReader masterCopyReader = new FileReader("DataSheet4-testingCopy.csv");
        FileReader generatedFileReader = new FileReader("DataSheet4.csv");
        int i = 0, j = 0;
        while(i != -1 && j != -1) {
            i = masterCopyReader.read();
            j = generatedFileReader.read();

            while((char) i == '\r' || (char) i == '\n' || (char) i == ',') {
                i = masterCopyReader.read();
            }

            while((char) j == '\r' || (char) j == '\n' || (char) j == ',') {
                j = generatedFileReader.read();
            }

            if((char) i == '%') { //Filler for a random string
                do {
                    j = generatedFileReader.read();
                } while(',' != (char) j);
                masterCopyReader.read();
                continue;
            } else if ((char) i == '&') { //Filler for a random number
                do {
                    assertTrue(numbers.contains((char) j));
                    j = generatedFileReader.read();
                } while(',' != (char) j);
                masterCopyReader.read();
                continue;
            }

            char a = (char) i;
            char b = (char) j;
            assertEquals(a, b);
        }
        masterCopyReader.close();
        generatedFileReader.close();
        if(i != -1 || j != -1)
            Assertions.fail("Files were not identical");

    } catch (FileNotFoundException e) {
        Assertions.fail("Test threw File Not Found Exception");
    } catch (IOException e) {
        Assertions.fail("Test threw IO Exception");
    }
}
```