Course:	**INFO-5051**
Professor:	**Tony Curcio**
Project:	**Project – *NHL Game Tracker***
Due Date:	**???**
Submitting:	**Please see the last page for instructions.**

Note: This project may be done individually, in pairs, or groups of three.. Just ensure that all names are included in the doc header of each class you submit.

How will my project be marked?

- This project counts for 25% of your final mark and will be graded using to the following grid:

Marks Available	**What are the marks awarded for?**	**Mark Assigned**
1	Good coding style including proper indentation and use of variable and class naming conventions and suitable commenting.	
2	Submission of two sql files named DDL.sql(creating database objects) and DML.sql(table insertions as require)	
Total 16	Controller Class	
2	All combo boxes are properly initialized/reinitialized as required.	
4	All command buttons are hooked up and fully functional.	
8	All data rules are implemented correctly using suitable methods created in the Controller class. Data input is correct, and data retrieval is all correct	
2	Error validation implemented correctly to catch data entry errors and message boxes alert user to the error .(e.g. User must select different home and away teams for a particular game. Can't have Toronto play itself!) Also, correct use of message boxes to display operation verification notifications (e.g. Are you sure you want to Delete?)	
Total 8	Viewer Class	
8	All command buttons fully functional Displayed data is correctly ordered in the JTable as per specifications.	
27	Total	
2	BONUS Opportunity: 1) Create the View WITHOUT a "Refresh" button so that whenever a change is made to the database the view updates in real time. 2) If the user displays the "All teams" option, add a column to the right of the team name to show which division the team plays in.	

Problem Description

Your task is to implement a Java application called *NHLGameTracker*. The application can be used to record the results of regular season games between teams in the Nation Hockey League.
The application will also display the regular season standings of the teams in any or all of the six divisions of the NHL, and this will update in real time if the user enters the result of another game.

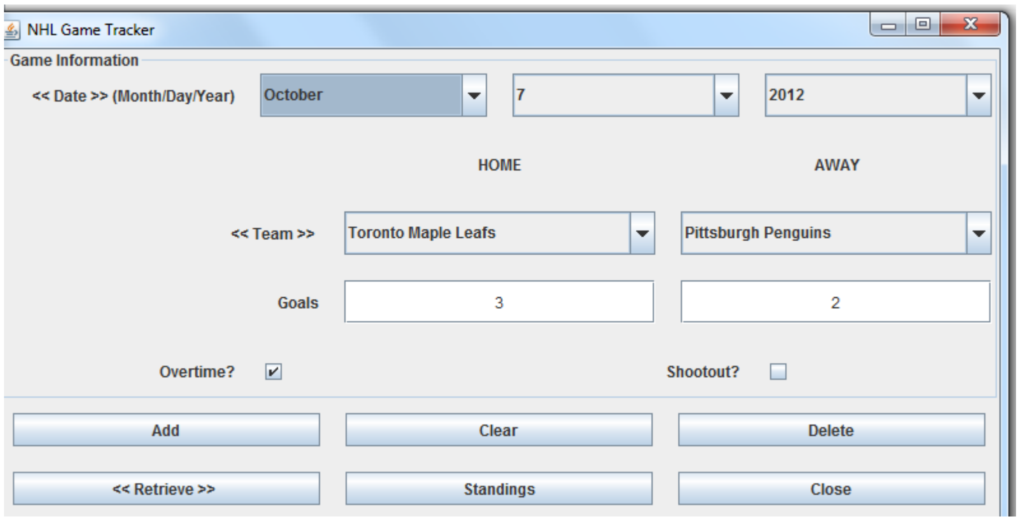
The user will enter game results data using the controller GUI form created by a class named ***your_inititals*****_NHLController.java**. You will be given the skeleton code that presents the layout. You will then have to write all the code for the event handlers to hook up the buttons, text fields, and combo boxes so that the user can insert, update, or delete results in the database tables. The results will be displayed using a JTable.

Specific Requirements:

I have provided you with the skeleton code for a controller form that uses the following controls:

- a group panel for <u>Game Information</u>
- label field prompts for:
  - << Date >> (month/day/year):
  - << Team >> (Home / Away):
  - Goals (Home / Away):
  - Overtime?:
  - Shootout?:
- text boxes for:
  - Goals (maximum 2 characters)
- Combo boxes for:
  - month / day / year
  - << Team >>
- Check box for:
  - Overtime?
  - Shootout?
- Button controls for:
  - Add
  - Clear
  - Delete
  - << Retrieve >>
  - Standings
  - Close

NOTE: Fields or buttons shown with << >> are involved in retrieving saved games from the database, in case user wants to confirm that an entry was correct. So, if the user wanted to see the data for the game played by the Toronto Maple Leafs *against the Pittsburgh Penguins on Oct. 7, 2012, in Toronto, he would enter the date and the name of the home team and press the <<retrieve>> button and this is what he/she would see:*



You must complete this task using a mysql database. You are to create the database and provide the sql files required to mimic the database creation on submission. The database has the following tables,

The ***Games*** table tracks individual games played during the season and contains the following fields:

Field Name	*Field Type*
GameID	integer (System Assigned) *primary key*
HomeTeamID	integer
AwayTeamID	integer
HomeTeamGoals	integer
AwayTeamGoals	integer
GameMonth	text (9)
GameDay	integer
GameYear	integer
Overtime	Boolean (Yes/No)
Shootout	Boolean (Yes/No)

The ***Teams*** table tracks team info and statistics and contains the following fields:

Field Name	*Field Type*	
TeamID	integer	*primary key*
TeamName	text(50)	
Division	text(30)	
GamesPlayed	integer	
Wins	integer	
RegulationLosses	integer	
OvertimeLosses	integer	
ShootoutLosses	integer	
Points	integer	

Install this database as an ODBC data source called ***NHL2013db.*** When the project is graded, it will be tested against my copy of the database with all of the above names and properties.

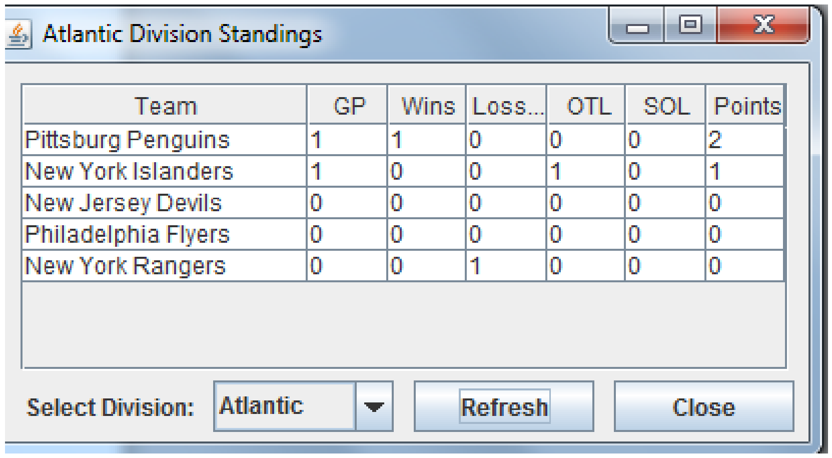
The following additional requirements must be implemented in your Controller:

- On startup, the window positions to the centre of the screen and has the NHL Game Tracker title bar.
- Initialize the Year combo box with the 10 years from 2012 to 2021 in 4 digit format.
- Initialize the Month combo box with all alphabetic months (i.e. January, February, March… etc.) – NOTE: That your program should work regardless of what case format is used to store months in the *Games* table (e.g. “JaNuArY” or “January”, etc.).
- Initialize the Day combo box with the correct numeric days of the month depending on which month and year are currently selected (e.g. 1 to 29 for Feb, 2012, a leap year).
- When the user clicks the Add button:
    - Validate the input data to ensure that all the relevant fields are all populated, that a different team has been selected for *Home* and *Away* and that the goals scored by each team are different since ties aren’t allowed in the NHL. Display a message box if the validation fails.
    - *Ideally you would also check to ensure there isn’t already another game involving either team on the same day, but you can keep it simple and skip this validation if you like.*
    - For valid input data, save all the **relevant** fields (i.e. Text, combo boxes and check boxes) to a new record in the *Games* table of the database. Note that the *GameID* field is an auto-increments field and should not be assigned by your INSERT query.
    - For valid input data, also update the *Teams* table for the two teams in the entered game. Specifically, the *GamesPlayed*, *Wins*, *RegulationLosses*, *OvertimeLosses*, *ShootoutLosses*, and *Points* fields should be updated as indicated by the game data. The *Rules for Updating a Team* are provided below.

- When the user clicks *Clear*, clear data displayed in all input fields. You should first allow the user to choose to save any new game data that has been entered as a new record in the *Games* table.
  - When the user clicks the Delete button, delete the record corresponding to the currently displayed Date, *Home* team and *Away* team – if there is one in the *Games* table. First verify that the user would like to perform the delete via a confirmation dialog box. Also update the *Teams* table for the two teams in the game that is about to be deleted. Specifically, the *GamesPlayed*, *Wins*, *RegulationLosses*, *OvertimeLosses*, *ShootoutLosses*, and *Points* fields should be updated as indicated by the game data.
- When the user clicks the << Retrieve>> button, use the values of the currently displayed Date, *Home* team **and/or** *Away* teams to retrieve and display the information for that game. Again, you should first allow the user to choose to save any new game data that has been entered as a new record in the *Games* table.

- When the user clicks the Standings button, your program will open a separate form, which you will need to code. Your program needs to call-up the form (instantiate it and make it visible) and pass to it a TableModel object to fill the JTable.

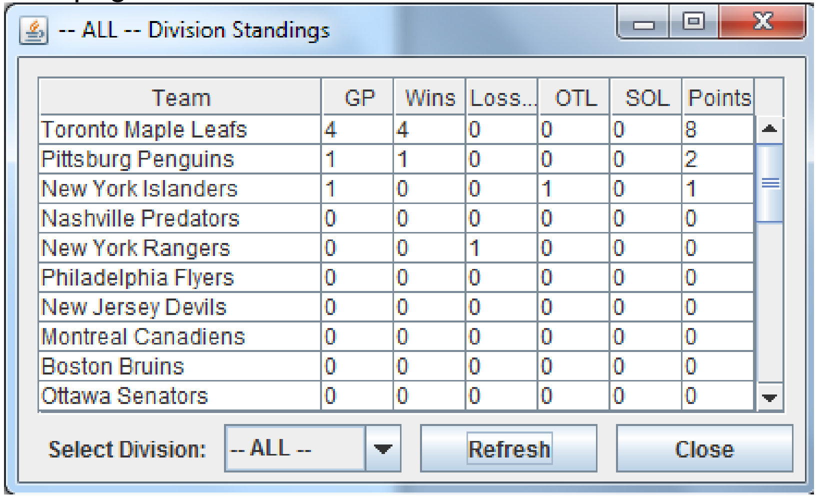
- While the form is open, the user can continue to enter game data through the controller. When the user enters the new game data into the database, the form can be refreshed by pressing the "Refresh" button to update the View. You will need to write a method to do this.
  The screen shot below is of the form showing results for the teams in the Atlantic Division of the NHL.



Team	GP	Wins	Loss...	OTL	SOL	Points
Pittsburg Penguins	1	1	0	0	0	2
New York Islanders	1	0	0	1	0	1
New Jersey Devils	0	0	0	0	0	0
Philadelphia Flyers	0	0	0	0	0	0
New York Rangers	0	0	1	0	0	0

- Note that the teams are displayed in the order of the points they have obtained based on their game results.
- The combo box next to the "Select Division:" label lists all six divisions, so the user can view the standings of teams in any one division. It also has a " All" Teams option, which will display all of the teams in the league ordered by

total points. See example the example screenshot of the “All” display on the next page:



Team	GP	Wins	Loss...	OTL	SOL	Points
Toronto Maple Leafs	4	4	0	0	0	8
Pittsburg Penguins	1	1	0	0	0	2
New York Islanders	1	0	0	1	0	1
Nashville Predators	0	0	0	0	0	0
New York Rangers	0	0	1	0	0	0
Philadelphia Flyers	0	0	0	0	0	0
New Jersey Devils	0	0	0	0	0	0
Montreal Canadiens	0	0	0	0	0	0
Boston Bruins	0	0	0	0	0	0
Ottawa Senators	0	0	0	0	0	0

Select Division: -- ALL --

- NOTE the team that is in first place in this fantasy data…
- When the user clicks <u>*Close on the View, just close the view.*</u>
- <u>*When the user clicks Close on the Controller*</u>, ask for verification and then end the program.

Database Rules for Updating a Team

When adding or deleting a game, your program should first identify the winning team and the losing team from the game information. The fields in the Teams table should then be modified as follows:

	Adding A Game		***Deleting a Game***	
Teams Field	*Winning Team*	*Losing Team*	*Winning Team*	*Losing Team*
GamesPlayed	+1	+1	-1	-1
Wins	+1	-	-1	-
RegulationLosses	-	+1 if neither *Overtime* or *Shootout* is true	-	-1 if neither *Overtime* or *Shootout* is true
OvertimeLosses	-	+1 if *Overtime* is true and *Shootout* is false	-	-1 if *Overtime* is true and *Shootout* is false
ShootoutLosses	-	+1 if *Shootout* is true	-	-1 if *Shootout* is true
Points	+2	+1 if an *Overtime* or *Shootout* loss	-2	-1 if an *Overtime* or *Shootout* loss

How should I submit my project?

Electronic Submission:

Submit your program files to the *Info5051 “Project”* electronic dropbox in *FanshaweOnline*. These files should be submitted as a single “zip” file containing all of your program’s .java .files. Test your zip file before submission to make sure it contains all the required files. All source files should include proper header comments identifying the student(s) involved.

Submit your project on time!

Project submissions must be made on time! Late projects will be subject to the School of I.T. policy on missed/late evaluations.

Submit your own work!

It is considered cheating to submit work done by another student or from another source. Helping another student cheat by sharing your work with them is also not tolerated. Students are encouraged to share ideas and to work together on practice exercises, but any code or documentation prepared for a project must be done by the individual student. Penalties for cheating or helping another student cheat may include being assigned zero on the project with even more severe penalties if you are caught cheating more than once. Just submit your own work and benefit from having made the effort on your own.