**README file for CS310\_Team\_Project**

**Project Summary**

This program allows us to experience the power of the Database by using jQuery to submit the connection between the client and the server, which we need to create multiple methods to alter or update the information that we want to edit, and we also can access to the main core of the Database to list out the data that we’ve inserted into the jQuery. It was resembling the idea of a checking attendance machine for a company. This was the class team project based on the concept of the Scrum, which was involving a Scrum Master and five program developers.

**Required Programming Tool**

1. The program required NetBeans Compiler 8.2 to compile and execute most of the codes from the project.
2. An account of the GitHub was requiring for committing the changes from each member.
3. An availability of Command Prompt was requiring forking or to update the changes of locations for the team project.
4. MySQL and its following products were requiring storing the corresponding files or data to access the program.
5. If possible, an application of having Discord account can help to stabilize the communication between the Scrum Master and the Scrum developers. It’ll reduce the negative effect of sending the java files during the Sprint periods, besides using the phone to communicate with other team members.

**Java Classes in the Program**

1. Cs310TeamProject.java
2. Badge.java
3. Punch.java
4. Shift.java
5. TASDatabase.java
6. TASLogic.java
7. Absenteeism.java

**Classes Documentation**

**Class: Cs310TeamProject**

This class did not play much of significant role, but it was created for us to test the timestamps based on the method.

**Java Imports:**

Import java.text.SimpleDateFormat

Import java.util.GregorianCalendar

|  |  |
| --- | --- |
| **Method** | **Description** |
| GregorianCalendar expected | This object was creating to store the expected time stamps. |
| GregorianCalendar actual | This object was creating to store the actual time stamps. |

**Class: Badge**

This class was creating to return the value of the badge from the Database, which was mainly involving the data of its id and its description.

**Java Imports: N/A**

**Attribute**

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| Private String description | Description represents as a String |
| Private String id | Id number represents as a String |

**Constructor**

|  |  |
| --- | --- |
| Public Badge(String id, String description) | Assigning the parameters to the corresponding attributes. |

**Methods**

|  |  |
| --- | --- |
| **Methods** | **Description or Return Type** |
| getDescription() | Return as String |
| getId() | Return as String |
| setDescription(String description) | Assign the corresponding String attributes |
| setId(String Id) | Assign the corresponding String attributes |
| toString() | Set the return as assigned String |

**Class: Punch**

This class was mainly adjusting with the clock in and clock out status based on the punchtype, badgeId, punchId, and original and adjusted timestamp.

**Java Imports:**

|  |
| --- |
| import java.util.\*; |
|  |

|  |
| --- |
| import java.text.SimpleDateFormat; |
|  |

import java.time.LocalTime;

**Attributes**

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| Public static final int CLOCK\_IN | Constant value |
| Public static final int CLOCK\_OUT | Constant value |
| Public static final int TIME\_OUT | Constant value |
| Private Badge badge | Create an object from the Badge class |
| Private int terminalId | terminalId represents as integer |
| Private int punchId | punchId represents as integer |
| Private int shiftId | shiftId represents as integer |
| Private int punchType | punchType represents as integer |
| Private GregorianCalendar originalTS | OriginalTS represents as Gregorian Calendar |
| Private GregorianCalendar adjustedTS | adjustedTS represents as Gregorian Calendar |
| Private long Timestamp | Timestamp represents as long integer |
| Private String trigger | Trigger initially assigned as empty string |
| Private int day | Day represents as integer |

**Constructors:**

|  |  |
| --- | --- |
| Public Punch(Badge badge, int terminalId, int punchId) | It accepts and assign the corresponding values of badge, terinalId and punchId, and it also stores the value of adjusted timestamps from the original timestamp. |
| Public Punch(Badge badge, int terminalId, int punchId, String BadgeId, long Timestamp, int punchType) | It is quite similar to the one above, but it additionally accepts and assign the corresponding values of BadgeId, Timestamp, and PunchType; nevertheless, it stores the value of adjusted timestamp as well. |

**Methods**

|  |  |
| --- | --- |
| **Methods** | **Description or Return** |
| setBadge(Badge badge) | Assign the badge |
| setTerminalId(int terminalId) | Assign the terminalId |
| setPunchId(int punchId) | Assign the punchId |
| setPunchType(int punchType) | Assign the punchType |
| setShiftId(int shiftId) | Assign the shiftId |
| setOriginalTS(GregorianCalendar originalTS) | Assign the original timestamp |
| setAdjustedlTS(GregorianCalendar adjustedTS) | Assign the adjusted timestamp |
| getPunchId() | Return punchId |
| getBadgeId() | Return badgeId |
| getPunchType() | Return punchType |
| getAdjustedTS() | Return adjusted timestamp in millisecond |
| getOriginalTS() | Return original timestamp in millisecond |
| getBadge() | Return badge |
| getTerminalId() | Return terminalId |
| getShiftId() | Return shiftId |
| getOriginalTimeStamp() | Return original timestamp in millisecond |
| getTrigger() | Return trigger |
| printOriginalTimestamp() | Print the corresponding timestamp based on the punchtype in the Date form of “EEE MM/DD/YYYY” |
| adjust(Shift s) | Print the adjusted timestamp from the corresponding clock in and clock out value in millisecond |
| printAdjustedTimestamp() | Print the corresponding timestamp based on the punchtype in the Date form of “EEE MM/DD/YYYY” |
| getDayOfWeek() | Return the day of the week from the Gregorian Calendar |

**Class: Shift**

This class was mainly adjusting with different shifts for each employee, which it is involving the interference of grace period and lunch time database.

**Java Imports:**

|  |
| --- |
| import java.time.LocalTime; |
|  |

|  |
| --- |
| import java.time.temporal.ChronoUnit; |
|  |

import static java.time.temporal.ChronoUnit.MINUTES;

**Attributes**

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| Private int id | Id represents as integer |
| Private String description | Description as String |
| Private LocalTime start | Start represents as Local Time |
| Private LocalTime stop | Stop represents as Local Time |
| Private int interval | Interval represents as integer |
| Private int graceperiod | Graceperiod represents as integer |
| Private int dock | Dock represents as integer |
| Private int numOfDayInShift | Number of day in shift represents as integer |
| Private LocalTime lunchstart | Lunch start represents as Local Time |
| Private LocalTime lunchstop | Lunch stop represents as Local Time |
| Private int lunchdeduct | Lunch deduct represents as integer |

**Constructors**

|  |  |
| --- | --- |
| Public Shift() | Assigning all the attributes to 0 or empty string, and it also assigned the start and stop time to the current local time. |

**Methods**

|  |  |
| --- | --- |
| **Methods** | **Description or return** |
| getStart() | Return start |
| getStop() | Return stop |
| getStart(int day) | Return start based on the day |
| getStop(int day) | Return stop based on the day |
| getShiftstarthour(int day) | Return shift start of hour |
| getShiftstartminute(int day) | Return shift start of minute |
| getShiftstophour(int day) | Return shift stop of hour |
| getShiftstopminute(int day) | Return shift stop of minute |
| getLunchstarthour(int day) | Return lunch start of hour |
| getLunchstartminute(int day) | Return lunch start of minute |
| getLunchstophour(int day) | Return lunch stop of hour |
| getLunchstopminute(int day) | Return lunch stop of minute |
| getLunchstop() | Return lunch stop |
| getLunchDuration() | Return lunch duration |
| setShiftStart(int hour, int minute) | Return local start time |
| getShiftLength() | Return shift length in minutes |
| getTotalScheduledHours() | Return approximal schedule hour |
| getShiftStartHour() | Return start hour |
| getShiftStartMinute() | Return start minute |
| setShiftStop(int hour, int minute) | Return local stop time |
| getShiftStopHour() | Return local stop time in hour |
| getShiftStopMinute() | Return local stop time in minute |
| setShiftLunchStart(int hour, int minute) | Return local lunch start |
| getShiftLunchStartHour() | Return local lunch start in hour |
| getShiftLunchStartMinute() | Return local lunch start in minute |
| setShiftLunchStop(int hour, int minute) | Return local lunch stop |
| getShiftLunchStopHour() | Return local lunch stop in hour |
| getShiftLunchStopMinute() | Return local lunch stop in minute |
| getId() | Return id |
| setId(int id) | Assign id |
| getDescription() | Assign description |
| setDescription(String description) | Assign description |
| getInterval() | Return interval |
| setInterval(int interval) | Assign interval |
| getGraceperiod() | Return grace period |
| setGraceperiod(int graceperiod) | Assign grace period |
| getDock() | Return dock |
| setDock(int dock) | Assign dock |
| getLunchdeduct() | Return lunch deduct |
| setLunchdeduct(int lunchdeduct) | Assign lunch deduct |
| toString() | Return the corresponding output |
| getNumOfDaysInShift() | Return number of days in shift |

**Class: TASDatabase**

This class was the main class that control and update the database throughout the entire project, which it would set up the connection to jQuery and follow certain commands to control the database.

**Java imports:**

|  |
| --- |
| import java.sql.\*; |
|  |

|  |
| --- |
| import java.util.GregorianCalendar; |
|  |

|  |
| --- |
| import java.util.\*; |
|  |

import java.text.SimpleDateFormat;

**Constructor:**

|  |  |
| --- | --- |
| Public TASDatabase() | Set up the connection to the database |

**Attribute**

|  |  |
| --- | --- |
| Private Connection conn | Main attribute to set up the connection |

**Methods**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| getBadge(String badgeID) | Iterate through the database, get the badge id and the description |
| getPunch(int punchID) | Iterate through the database, get the punch data by the punch Id |
| getShift(int shiftID) | Iterate through the database, get the shift data by the shift id |
| getShift(Badge badge) | Iterate through the database, get the shift data by the badge class |
| insertPunch(Punch p) | Iterate through the database, insert the punch data |
| ArrayList getDailyPunchList(Badge b, long ts) | Iterate through the database, getting badge and the timestamp data into the daily punch array list |
| ArrayList getPayPeriodPunchList(Badge b, long ts) | Iterate through the database, getting badge and the timestamp dtata into the get pay period punch list |
| getAbsenteeism(String id, long ts) | Iterate through the database, getting absenteeism data by using the id and the timestamp |
| insertAbsenteeism(Absenteeism absentee) | Iterate through the database, insert the corresponding absenteeism data to the database |

**Class: TASLogic**

This class is mainly generating the data and providing the array list as needed in the program.

**Java Imports:**

|  |
| --- |
| import java.text.SimpleDateFormat; |
|  |

|  |
| --- |
| import java.util.\*; |
|  |

import org.json.simple.\*;

**Constructor: N/A**

**Methods**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| calculateTotalMinutes(ArrayList<Punch>dailypunchlist, Shift shift) | Iterate through the data, return total minute |
| getPunchListAsJSON(ArrayList<Punch> dailypunchlist) | Iterate through the punch list data, return as the JSON data |
| calculateAbsenteeism(ArrayList<Punch> punchlist, Shift shift) | Iterate through the data, calculate the absenteeism, return as absenteeism |
| getPunchListPlusTotalsAsJSON(ArrayList<Punch> punchlist, Shift s) | Iterate through the data, getting the punch list plus total and return as JSON String |

**Class: Absenteeism**

This class is mainly dealing with the absentee’s percentages by returning the values.

**Java Imports:**

|  |
| --- |
| import java.text.DecimalFormat; |
|  |

|  |
| --- |
| import java.text.Format; |
|  |

|  |
| --- |
| import java.text.SimpleDateFormat; |
|  |

|  |
| --- |
| import java.util.Calendar; |
|  |

import java.util.GregorianCalendar;

**Constructor**

|  |  |
| --- | --- |
| Absenteeism(String badgeId, long payTS, double percentage) | Assign the corresponding attributes |

**Attributes**

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Private String badgeId | badgeId represents as String |
| Private long payTS | PayTS represents as long integer |
| Private double percentage | Percentage represents as double |

**Methods**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| getBadgeId() | Return badgeid |
| getpayTS() | Return pay timestamp |
| getPercentage() | Return percentage |
| setBadgeId(String badgeId) | Assign badgeid |
| setPayTS(long payTS) | Assign pay timestamp |
| setPercentage(double percentage) | Assign percentage |
| toString() | Return the corresponding output |