ITM411 Final Project

Application Documentation

A project comprised of Graphical User Interface (GUI) components from the Swing and AWT APIs using Net Beans

Tejal Gajare (A20287489)
5/10/2013

INDEX

1. Project Abstract

Aim: To Develop a Net Beans project comprised of Graphical User Interface (GUI) components from the *Swing* and *AWT* APIs using **MP2** as the back-end code base

Objective: The objective of this final project is to help us understand the fundamental JAVA Graphical User Interface (GUI) components and their connectivity to the database in order to perform the basic CRUD operations on the table present in a My SQL database

Basic Understanding:

JAVA Swing Components:

JFrame: A frame, implemented as an instance of the JFrame class, is a window that has decorations such as a border, a title, and supports button components that close or iconify the window.

JTabbedPane: With the JTabbedPane class, you can have several components, such as panels, share the same space. The user chooses which component to view by selecting the tab corresponding to the desired component.
To create a tabbed pane, instantiate JTabbedPane, create the components you wish it to display, and then add the components to the tabbed pane using the addTab method.

JButton: To create a button, you can instantiate one of the many classes that descend from the AbstractButton class.

JLabel: With the JLabel class, you can display unselectable text and images. If you need to create a component that displays a string, an image, or both, you can do so by using or extending JLabel.

JTextField: Also known simply as text fields, text controls can display only one line of editable text. Like buttons, they generate action events. Use them to get a small amount of textual information from the user and perform an action after the text entry is complete.

JTable: With the JTable class you can display tables of data, optionally allowing the user to edit the data. JTable does not contain or cache data; it is simply a view of your data.

JDBC Introduction:

The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database.

JDBC helps us to write Java applications that manage these three programming activities:

1. Connect to a data source, like a database
2. Send queries and update statements to the database
3. Retrieve and process the results received from the database in answer to the query

Relational Databases:

A database is a means of storing information in such a way that information can be retrieved from it. In simplest terms, a relational database is one that presents information in tables with rows and columns. A table is referred to as a relation in the sense that it is a collection of objects of the same type (rows). Data in a table can be related according to common keys or concepts, and the ability to retrieve related data from a table is the basis for the term relational database.

Establish a Connection:

First, we need to establish a connection with the data source we want to use. A data source can be a DBMS, a legacy file system, or some other source of data with a corresponding JDBC driver. Typically, a JDBC application connects to a target data source using one of two classes:

- DriverManager: This fully implemented class connects an application to a data source, which is specified by a database URL. When this class first attempts to establish a connection, it automatically loads any JDBC 4.0 drivers found within the class path.
- DataSource: This interface is preferred over DriverManager because it allows details about the underlying data source to be transparent to the application. A DataSource object's properties are set so that it represents a particular data source.

Establish a Database Connection URL:

A database connection URL is a string that the DBMS JDBC driver uses to connect to a database. It can contain information such as where to search for the database, the name of the database to connect to, and configuration properties. The exact syntax of a database connection URL is specified by the DBMS.

The following is the database connection URL syntax for Java DB:

```
jdbc:derby:[subsubprotocol:][databaseName][;attribute=value]
```

The following is the database connection URL syntax for MySQL Connector/J:

```
jdbc:mysql://[host][,failoverhost...]
             [:port]/[database]
             [?propertyName1][=propertyValue1]
             [&propertyName2][=propertyValue2]...
```

- `host:port` is the host name and port number of the computer hosting your database. If not specified, the default values of host and port are 127.0.0.1 and 3306, respectively.
- `database` is the name of the database to connect to. If not specified, a connection is made with no default database.
- `failover` is the name of a standby database (MySQL Connector/J supports failover).
- `propertyName=propertyValue` represents an optional, ampersand-separated list of properties. These attributes enable you to instruct MySQL Connector/J to perform various tasks.

ResultSet: A ResultSet object is a table of data representing a database result set, which is usually generated by executing a statement that queries the database.

*Serialization and De-Serialization***:** Serialization is the process of writing the state of the object to a byte stream. This is useful if we want to save the state of the program to a persistent storage area such as a file. At a later time, we may restore these objects by using the process of de-serialization.

*Exception Handling***:** JAVA exception handling is managed by 5 keywords:

1. **Try:** To guard against and handle a run-time error, we enclose the code to be monitored inside a try block

2. **Catch:** Immediately following the try block, we include a catch clause that specifies the exception type of the exception to be caught

3. **Throw:** This enables to throw an exception explicitly. The flow of execution stops immediately after the throw statement, and the nearest enclosing try and catch block are examined to find the matching type of exception

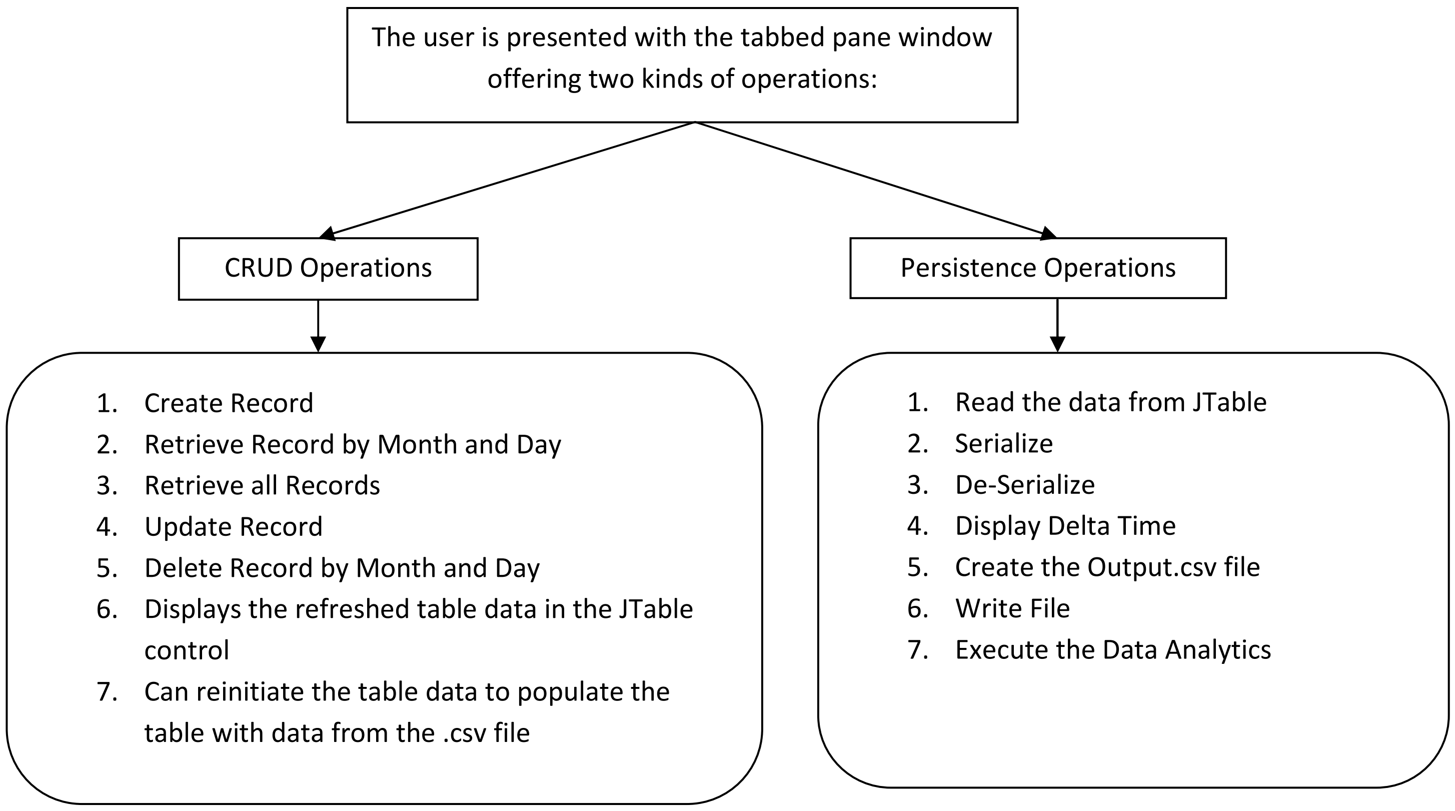
4. **Throws:** If a method is capable of causing an exception that it does not handle, it must specify this behavior so that the callers of the method can guard themselves against that exception. This can be done by using throws clause in the method declaration

5. **Finally:** This block contains the code that will be executed after a try/catch block has completed and before the following try/catch block. This block is executed whether or not an exception is thrown.

All exception types are subclasses of the built-in class Throwable.

2. Project Diagram

- Once the Project Execution begins, the flow of data and control happens as follows:


The user is presented with the tabbed pane window offering two kinds of operations:
CRUD Operations
Persistence Operations
1. Create Record
2. Retrieve Record by Month and Day
3. Retrieve all Records
4. Update Record
5. Delete Record by Month and Day
6. Displays the refreshed table data in the JTable control
7. Can reinitiate the table data to populate the table with data from the .csv file
1. Read the data from JTable
2. Serialize
3. De-Serialize
4. Display Delta Time
5. Create the Output.csv file
6. Write File
7. Execute the Data Analytics

3. Project Code Structure

Architecture or System Flow

3.1 Creation of Java Application:
Launch NetBeans and click on new Project to choose a new Java Application, name it as 'SolsticeEquinoxAnalysis' and click on Finish. Delete the Main driver class added by default. Add a JForm to the package 'finalproject'

3.2 Creation of packages:

We have in all 4 packages:

1. Default finalproject – This includes the SolsticeEquinoxAnalysis.java class (The main driver class)
2. Domain – This includes the Record.java, DaylightRecord.java, PersistentObject.java, Equinox.java, VernalEquinox.java, SumerSolstice.java sand WinterSolstice.java classes
3. Domain.util – This includes the MP2Utilities.java class
4. CRUD- This includes the UtilityFunctions,java class

3.3 Creation of classes/interfaces (Java files):

- SolsticeEquinoxAnalysis class- This class is the main driver class which has access to all the Swing Controls in order to design the required CRUD and Persistence Operations GUI

- Record class- This is the basic abstract class which contains the fields to store the month, date and their respective sunrise and sunset times. The fields month , date, sunrise and sunset are stored as integer value. We can have just 2 fields such as 'sunrisetime' and 'sunsettime' declared of type Date. While parsing the .csv file we then will have to use DateFormat, SimpleDateFormat and Date classes to send these values to the DaylightRecord class constructor.

- DaylightRecord class- This class extends the Record class. This class also implements the serializable interface. It has few functions to calculate the difference between the sunset/sunrise times for the particular date of the invoking object.

- PersistentObject class- This class encapsulates the Current Timestamp in the form of a date object and the read .csv input in the form of a list object.

- WinterSolstice class- This class implements the Comparator interface in order to return the date for the winter solstice. It implements the compare() method to compare the differences between the sunrise/sunset times of the invoking objects to find out the winter solstice date.

- SummerSolstice class- This class implements the Comparator interface in order to return the date for the summer solstice. It implements the compare() method to compare the differences between the sunrise/sunset times of the invoking objects to find out the summer solstice date.

- Equinox class- This class implements the Comparator interface in order to return the date for the autumnal equinox. It implements the compare() method to compare the differences between the sunrise/sunset times of the invoking objects to find out the autumnal equinox date.

- VernalEquinox class- This class implements the Comparator interface in order to return the date for the vernal equinox. It implements the compare() method to compare the differences between the sunrise/sunset times of the invoking objects to find out the vernal equinox date.

- MP2Utilities class - This class contains methods responsible for serialization , de-serialization and displaying the output results

- UtilityFunctions class- This class encapsulates the various methods in order to perform the tasks of creating,updating, retrieving and deleting of the records from the 'SunriseSunset' Table of itm411db database created in MySql. This table is populated using the sunset-sunrise.csv file

4. Project Read Me Details

- In order to run this project please launch Net Beans IDE

- In order to save the contents of the sunset-sunrise.csv file, I have created a table called 'SunriseSunset' in the itm411db using My SQL. So in order to execute this project without any details please make sure you have a itm411db created in your MySQL

- The Javadocs are located in the FinalProject Docs folder (which should be present in the zipped package)

5. Project Specifications

- The System Requirements are as follows:

  OS: Windows 7
  JDK: 1.7.0
  IDE: NetBeans IDE 7.2.1

- The Database Requirements are as follows:

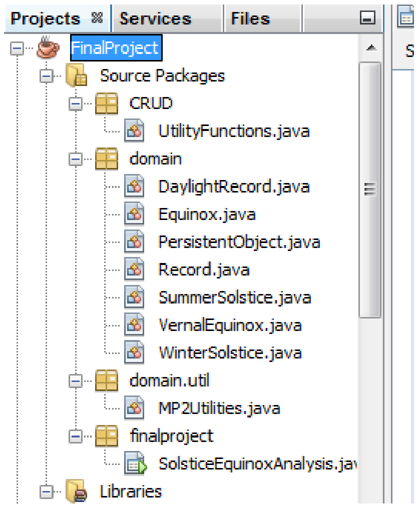
  Db: MySQL

6. Project Summary and Insights

- The Final Project is expected to use the Java Swing Components to implement the MP2 functionality of serializing and de-serializing using GUI controls
- It reads the data from the given .csv file, parses it and saves it in a table named as sunrisesunset in the itm411db of MYSQL using JDBC connectivity
- Once the data is read, it is saved in the DataModel created by using the JTable of Swing
- The user is able to perform the basic CRUD Operations on the table using various available swing controls
- Whatever modifications are made by the user, are reflected in the adjacent JTable
- The user is also able to perform the persistence operations used in the MP2 using various controls
- The file for serialization is not the given csv file which was used previously in MP2 but now, the table data is fetched into the daylightRecord List
- So, whatever changes that are made by the user during the CRUD Operations are now considered while serializing the data
- There is an option to reset the table data to the original given .csv file
- After the serilization/de-serialization, the user can also view the data analytics by clicking on the ‘Execute’ Button
- The Data Analytics for the Solstice and Equinox is displayed in the respective controls
- There is a status label at the bottom of the application which keeps the user notified about the current operation status
- This is how the Final Project is implemented successfully using the MP2 at the backend and GUI Swing Controls in the front end

7. Project Screen Captures

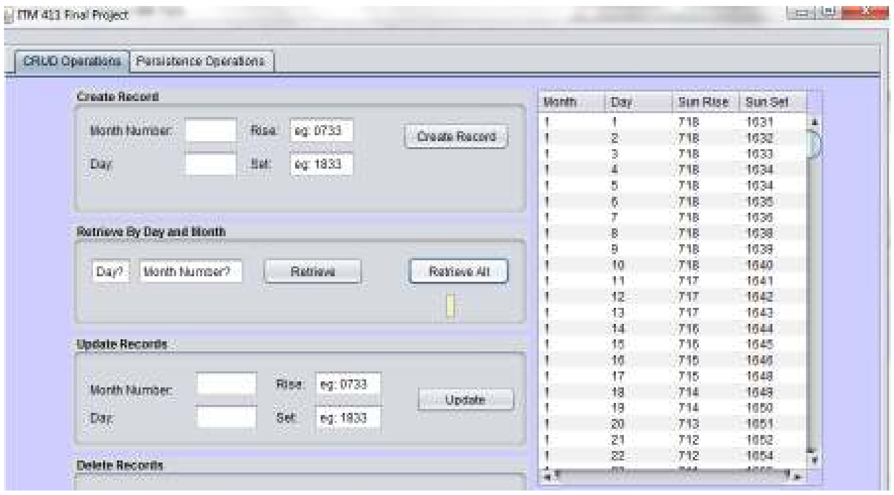
1. Folder Structure



```
FinalProject
  Source Packages
    CRUD
      UtilityFunctions.java
    domain
      DaylightRecord.java
      Equinox.java
      PersistentObject.java
      Record.java
      SummerSolstice.java
      VernalEquinox.java
      WinterSolstice.java
    domain.util
      MP2Utilities.java
    finalproject
      SolsticeEquinoxAnalysis.ja
  Libraries
```

2. CRUD Operations

- Retrieve All



Month Number:

Rise: eg: 0733

Day:

Set: eg: 1833

Day?

Month Number?

Month Number:

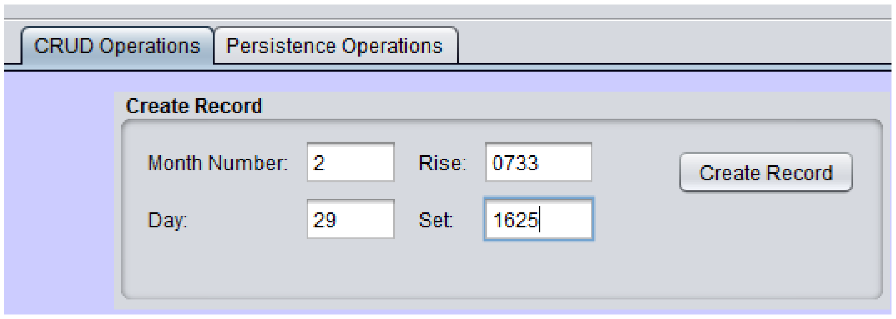
Rise: eg: 0733

Day:

Set: eg: 1833

Month	Day	Sun Rise	Sun Set
1	1	718	1631
1	2	718	1632
1	3	718	1633
1	4	718	1634
1	5	718	1634
1	6	718	1635
1	7	718	1636
1	8	718	1638
1	9	718	1639
1	10	718	1640
1	11	717	1641
1	12	717	1642
1	13	717	1643
1	14	716	1644
1	15	716	1645
1	16	715	1646
1	17	715	1648
1	18	714	1649
1	19	714	1650
1	20	713	1651
1	21	712	1652
1	22	712	1654

- Create Record



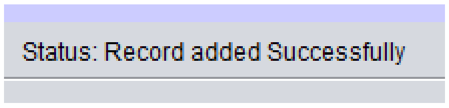
Month Number: 2

Rise: 0733

Day: 29

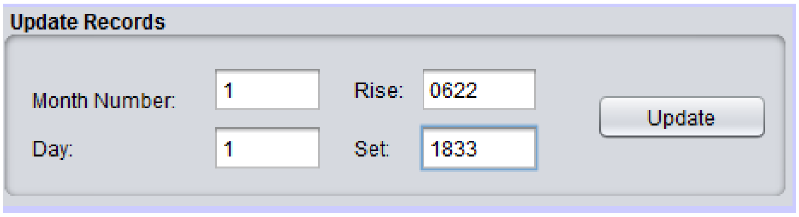
Set: 1625

- Status Label:



Status: Record added Successfully

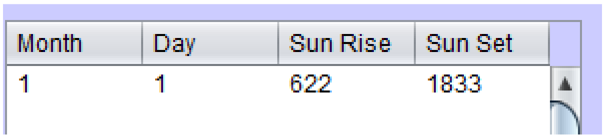
- Update



Update Records

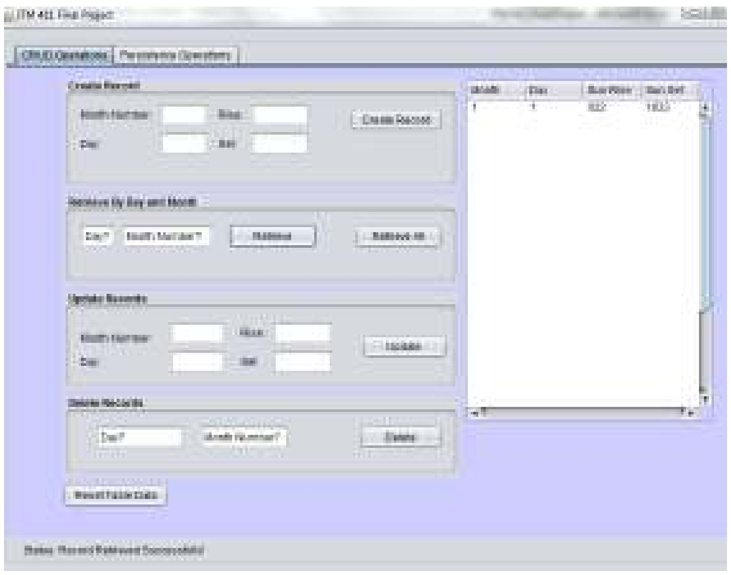
Month Number: 1   Rise: 0622
Day: 1   Set: 1833

- Retrieve by ID



Month	Day	Sun Rise	Sun Set
1	1	622	1833

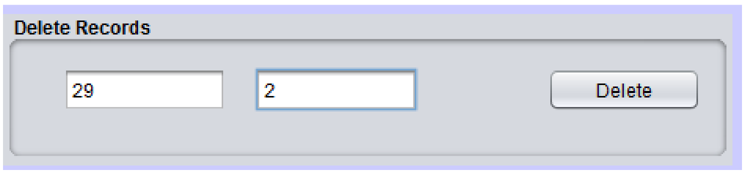
- Reset all controls after any operation



Month	Day	Sun Rise	Sun Set
1	1	622	1833

Status: Record Retrieved Successfully!

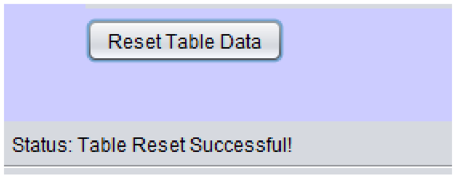
- Delete Record



Delete Records

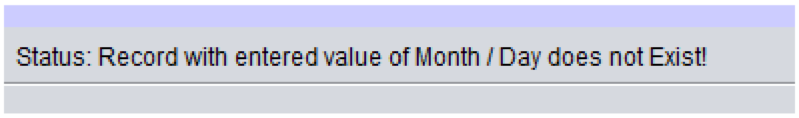
29   2

- Reset Table



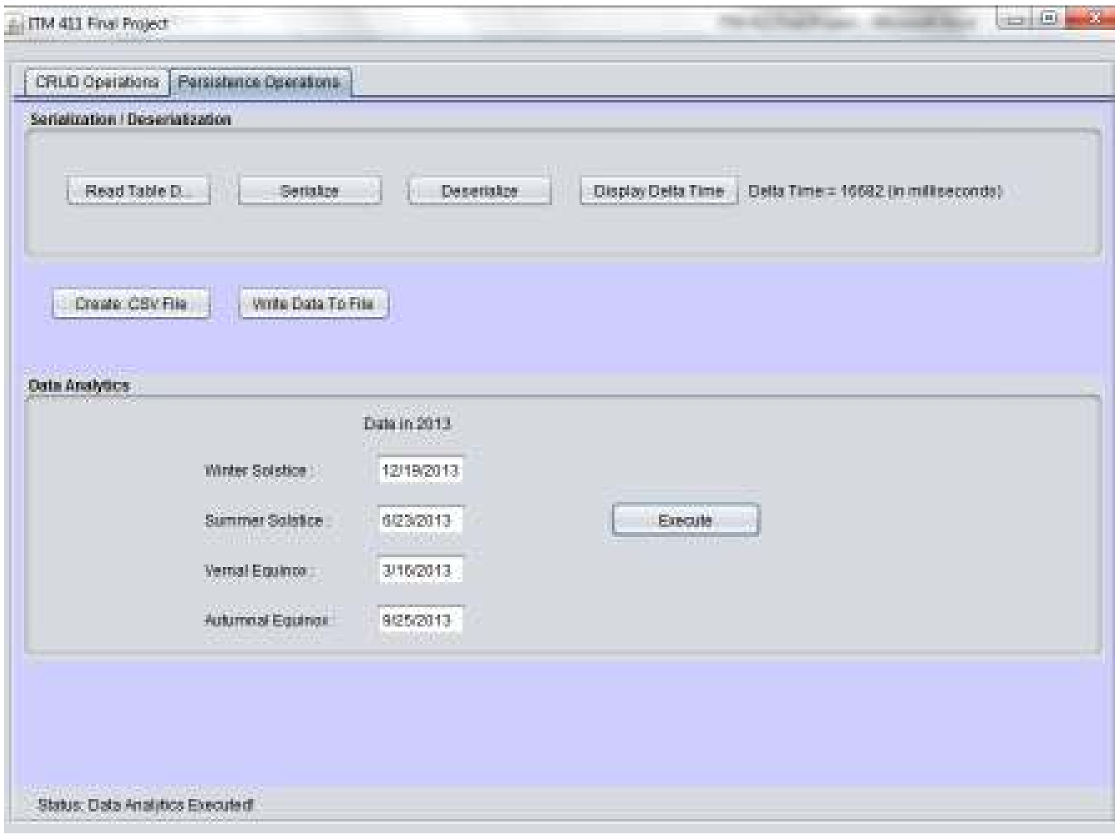
Status: Table Reset Successful!

- Provides Error Handling if data does not exist in the database in every CRUD Operation
  Eg: Tried to delete Feb 29 when it gave the following error prompt:



Status: Record with entered value of Month / Day does not Exist!

- Persistence Operations execute on every Button Click and then Data Analytics is displayed on clicking the Execute Button



Delta Time = 16682 (in milliseconds)

Data in 2013

Winter Solstice : 12/19/2013
Summer Solstice : 6/23/2013
Vernal Equinox : 3/16/2013
Autumnal Equinox : 9/25/2013

Status: Data Analytics Executed!