# Project Title: World Timezone Converter

| | |
|---|---|
| **Project Description** | This project develops a User-Interfaced program which allows the user to convert a date and time from one time zone to another based on the selected locations. The user selects the "to" and "from" locations from an interactive drop-down menu, which are then displayed as pins on a world map. The user subsequently selects the date and time they wish to convert from any of the pins via a pop-up calendar and time spinner. The converted date and times are returned and displayed on all pins. |
| **Project Motivation** | In this age of globalization, collaborating with colleagues from other regions and countries is not only of common occurrence but also inevitable, especially in large organizations and professional communities. This project provides a useful tool to simplify the process of coordinating meetings and events involving collaborators who are geographically located in different time zones. |
| **Innovation and Value** | How often do you find yourself Googling "what time is this in [another city]?" followed by reading a long conversion table provided by TimeBie (or other similar websites)? Our project end product is specifically designed to streamline this process and offers a convenient solution to the everyday time zone-conversion problem that confronts all those who work with remote collaborators. Apart from saving time and effort, the sophisticated and aesthetically attractive user interface further enhances the user experience. |
| **Project Scope** | This project serves as a proof of concept for the Timezone Converter rather than its full implementation. All major time zones are represented, however the scope of the project is restricted to 39 cities/locations. |
| **Development Team** | MCIT 591 Team 14: Sai Merriam, John Caton, Chelsea Liu |

## Features

- Converts date and time from one city/location to other cities/locations.
- Allows user to select "To" and "From" locations from a drop-down menu of major cities.
- Displays pins representing cities/locations on a world map.
- Displays the correct converted date and time on the city pins across the world map.

## Getting Started

These instructions are designed to help you run a copy of the program on your local machine.

### Installing

- Install JavaFx on your local machine (see instructions).
- Connect JavaFx to Eclipse or other IDE (see instructions).
- Install controlsFX (see following GitHub Page, Wiki, .Jar Repository [8.40.15 or later only])

## Files

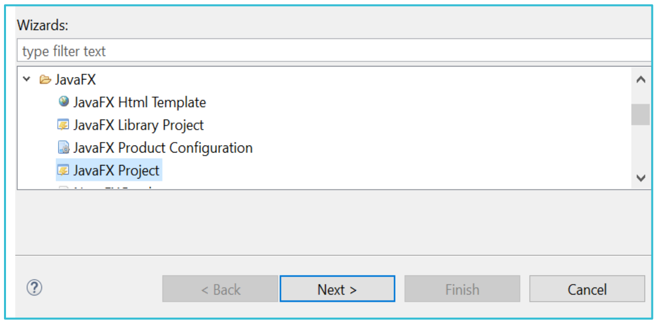
- AutoCompleteComboBoxListener.java
- DropDownMenu.java
- Main.java
- Pin.java
- TimeConverter.java
- TimeConverterTest.java
- TimeSpinner.java
- ZoomFunctions.java

## Supplementary Files

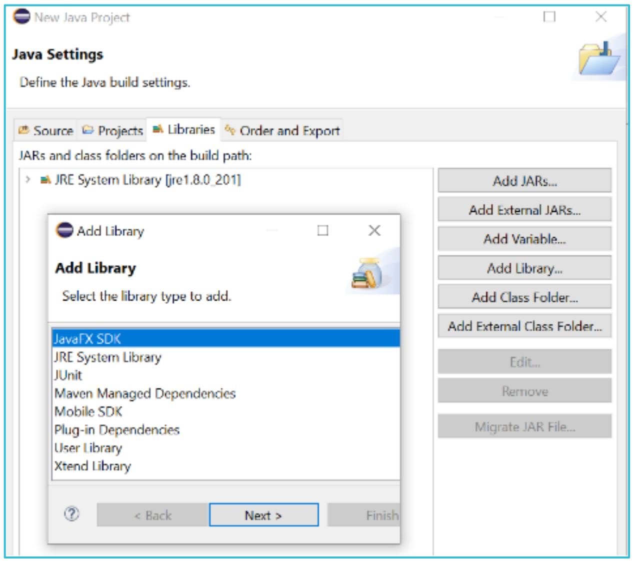
- Map_Export500.png
- GreenPinImage100.png
- PinkPinImage100.png
- WhitePinImage100.png
- Library: controlsfx-8.40.15.jar

## Step-by-Step Set-Up Instructions

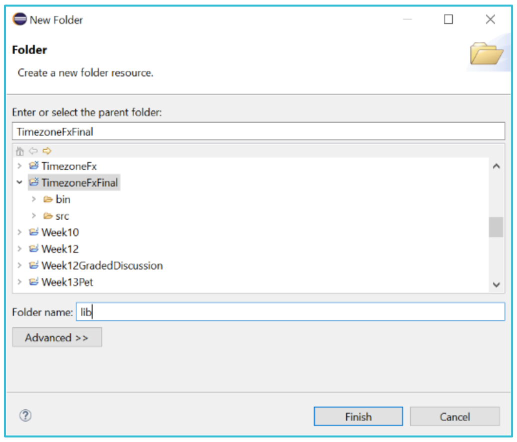
- Step 1: Create a New JavaFx Project in Eclipse (or other IDE).

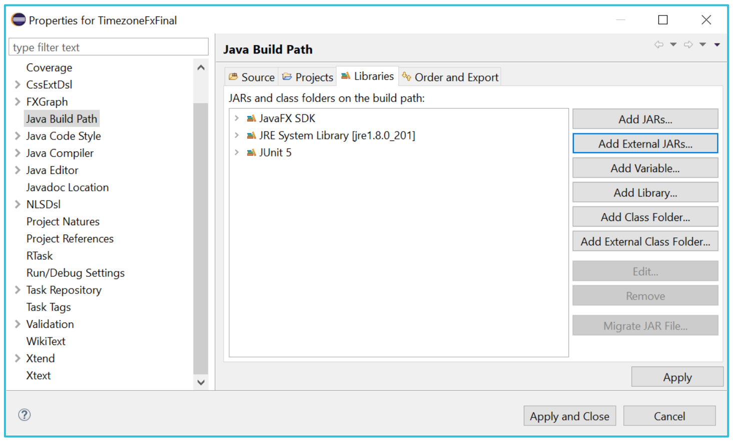


- Step 2: Ensure that the JavaFX SDK Library is added to the list of available Libraries: click “Add Library”, click JavaFx SDK in the pop-up window (see figure: right).

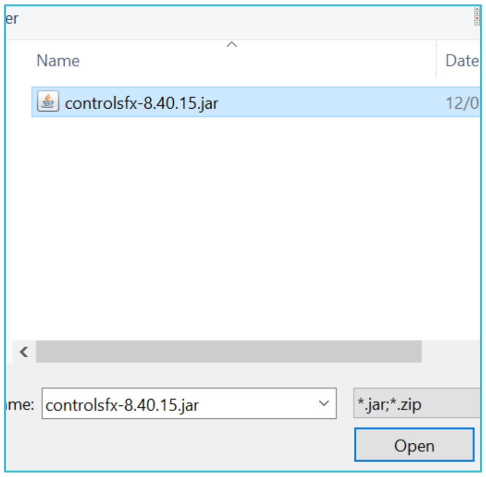


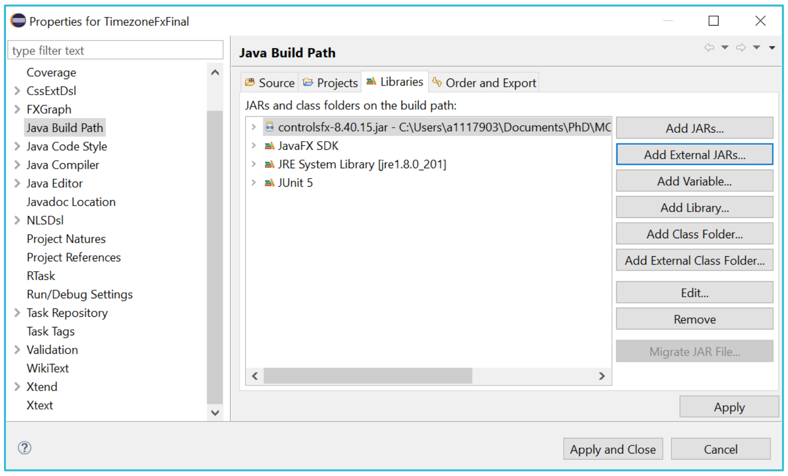
- Step 3: Add additional library:
  Create a folder named “lib” within the project (see figure below: left).
  Copy-and-paste supplementary file “controlsfx-8.40.15.jar” into this folder.





- Step 4: Link Additional Library to this JafaFx Project:
  Right-click project title in Eclipse. Select "Properties". Click Java Build Path (see figure above: right). Click "Add External JARs" – Select ""controlsfx-8.40.15.jar" placed in the lib folder (Step 3), as illustrated in figure below (left). Click open. This library file should now appear in the "Libraries Panel" (see figure below: right). Click "Apply and Close".





- Step 5: Place all Files in the "application" folder of the new JavaFx Project.

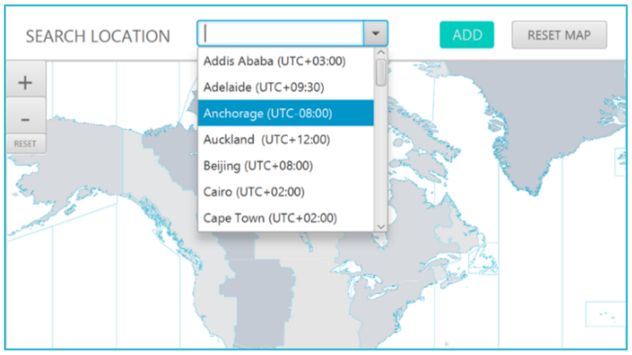
- Step 6: Place all Supplementary Files into the "src" folder of the new JavaFx Project – at the same level as where the "application" folder is located.

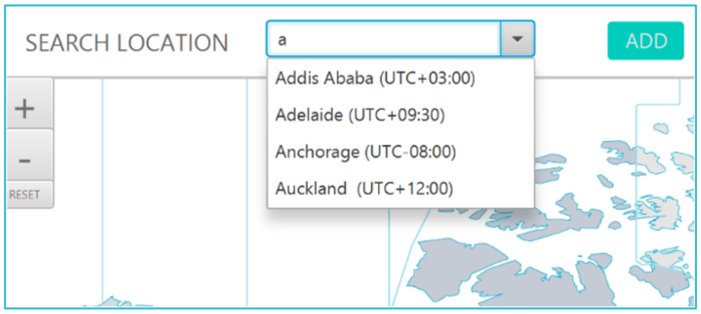
- Step 7: Run "main.java" file.

## How To Use?

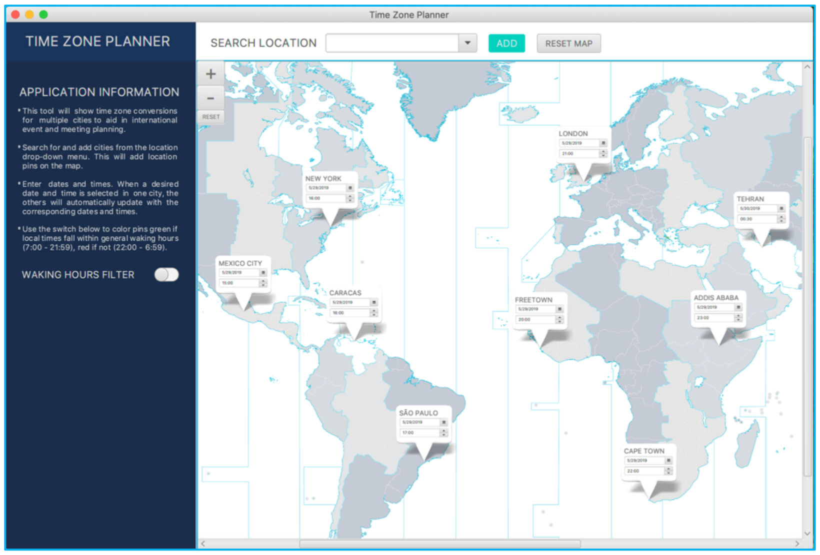
### Refer to Demo Video enclosed on GitHub.

Step 1: Use drop-down menu at the top of the screen to select the source and destination cities/locations that you would like to be displayed on the world map. You can also type in the name of the city that you are looking for. Then click the "ADD" button.





All selected cities appear on the world map.



TIME ZONE PLANNER

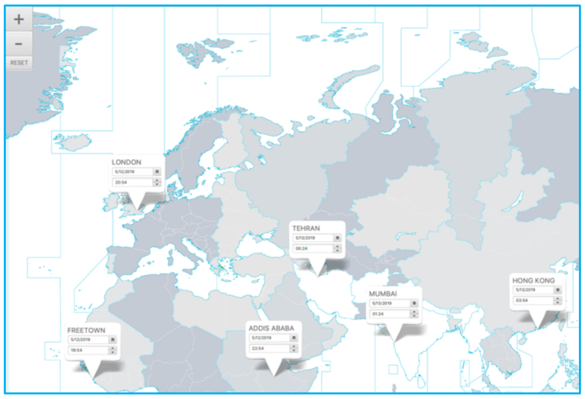
APPLICATION INFORMATION

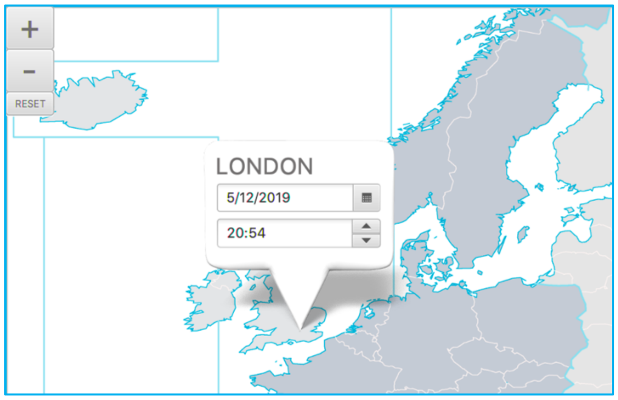
- This tool will show time zone conversions for multiple cities to aid in international event and meeting planning.
- Search for and add cities from the location drop-down menu. This will add location pins on the map.
- Enter dates and times. When a desired date and time is selected in one city, the others will automatically update with the corresponding dates and times.
- Use the switch below to color pins green if local times fall within general waking hours (7:00 - 21:59), red if not (22:00 - 6:59).

WAKING HOURS FILTER

Step 2: Zoom in and out on the world map using the + and - buttons, or use mouse scroll. Click and drag to shift the map, or use accompanying scroll bars.

Reset the scale of the world map using the RESET button.

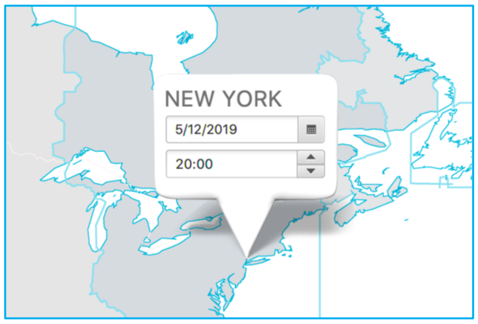


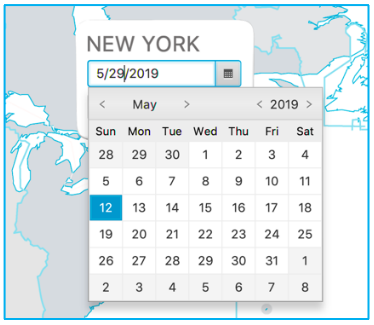


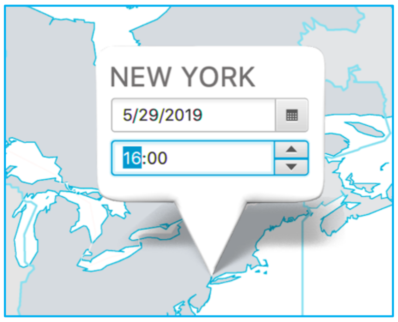
Step 3: Locate the city from which you wish to convert the date/time.

Date Selection: use the Calendar to pick the local date to be converted.

Time Selection: use the Spinner to input the local time to be converted.

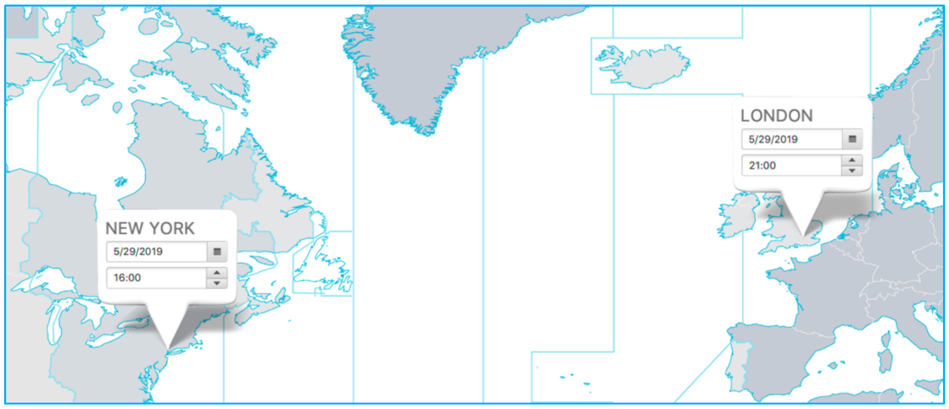






Step 4: Press Enter to see converted time and date displayed at the destination location(s).

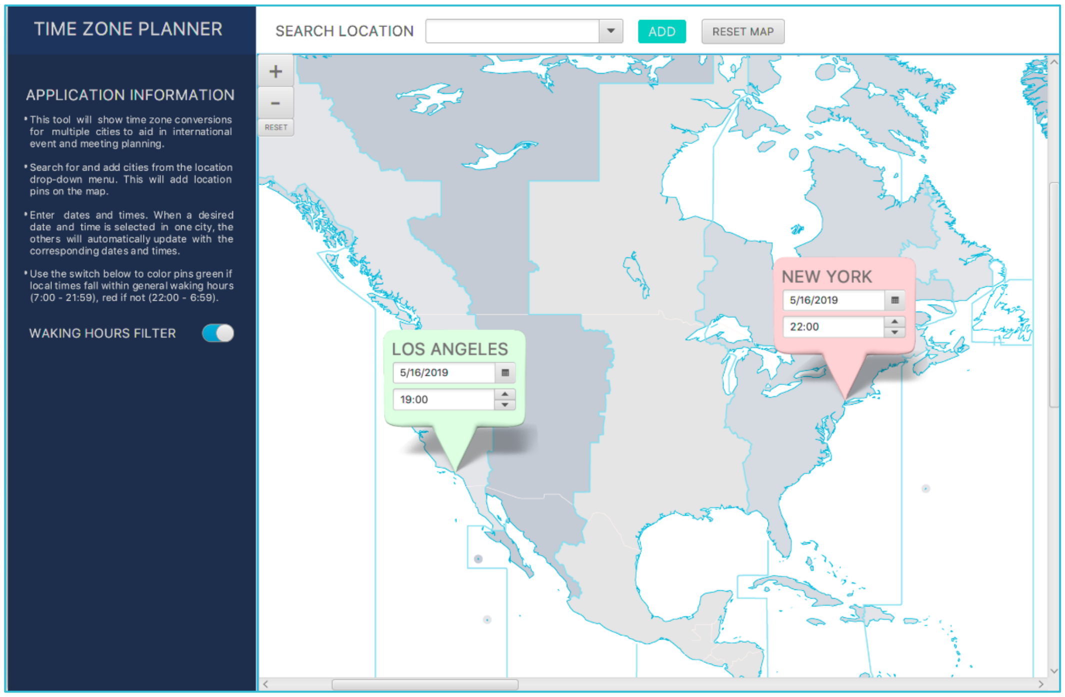
All cities displayed on the world map automatically update to display the converted time and date in their local timezones.



Step 5: Reset Cities on the map by clicking the RESET MAP button at the top of the screen before restarting a new conversion.

Step 6: Optional Feature: Waking Hour Filter

Toggle on the "Waking Hour Filter" on the left-hand-side panel to distinguish between daytime cities and nighttime cities according to the converted date/time. Cities which are in general waking hours (7:00 – 21:59) are colored green; in contrast, cities which are not in general waking hours (22:00 – 6:59) are colored pink.



APPLICATION INFORMATION

- This tool will show time zone conversions for multiple cities to aid in international event and meeting planning.
- Search for and add cities from the location drop-down menu. This will add location pins on the map.
- Enter dates and times. When a desired date and time is selected in one city, the others will automatically update with the corresponding dates and times.
- Use the switch below to color pins green if local times fall within general waking hours (7:00 - 21:59), red if not (22:00 - 6:59).

## Testing

### JUnit Tests

Run TimeConverterTest.java as a JUnit Test. The JUnit Test file provides multiple tests of all 3 methods in TimeConverter.java.

Method 1: Convert()

This method is designed to take in a local date & time at a source city, as well as the timezone IDs for both the source and destination cities, and return the converted date & time at the destination city (as a formatted string). A total of 5 tests (as illustrated in the example below) are written to ensure that the date *and* time are converted and displayed correctly.

```
class TimeConverterTest {
	TimeConverter tc1 = new TimeConverter();
@Test
	void testConvertSanFrancisco() {
		LocalDate date = LocalDate.of(2019, 1, 1);
		LocalTime time = LocalTime.of(0, 0);
		assertEquals(tc1.Convert(date, time, "US/Eastern", "US/Pacific"),
		"2018-12-31 21:00");
	}
}
```

Method 2: OutputFormatterDatePicker1()

This method is designed to take in the string output from the Convert method above and convert it into a LocalDate variable, which will then feed into the JavaFx DatePicker for display on the UI. Two tests are written (an example provided below) to ensure that the method functions correctly to convert the String input into the desired output.

```
@Test
void testOutputFormatterDatePicker1() {
	String input = "2019-05-31 18:30";
	tc1.outputFormatterDatePicker(input);
	assertEquals(tc1.getNewYear(), 2019);
	assertEquals(tc1.getNewMonth(), 5);
	assertEquals(tc1.getNewDay(), 31);
}
```

Method 3: OutputFormatterTimeSpinner1()

This method is designed to take in the string output from the Convert method above and convert it into a LocalTime variable, which will then feed into the JavaFx TimeSpinner for display on the UI. Two tests are written (an example provided below) to ensure that the method functions correctly to convert the String input into the desired output.

```
@Test
void testOutputFormatterTimeSpinner1() {
	String input = "2019-01-01 09:12";
	LocalTime time = LocalTime.of(9, 12);
	assertEquals(tc1.outputFormatterTimeSpinner(input), time);
}
```

### Test Coverage

All methods in the TimeConverter.java class are covered by the JUnit Tests.

Note: In this project we restrict our testing to only Java classes (TimeConverter.java) and not JavaFx classes (e.g., Pin.java, Main.java, and ZoomFunction.java), because the testing of JavaFx language requires entirely different methods and syntax. Upon consultation with Dr Arvind Bhusnurmath, the testing of JavaFx classes is deemed to be beyond the scope of this course and project.

## Technical Framework

### Built With

- JavaFx - Graphical User Interface (GUI)
- Java Date Time (newest in-built function to replace Joda Time)

## Design

For detailed notes on program design brainstorming (including Class, Responsibility, Collaborator (CRC) cards), please refer to document "Design Notes and CRC.docx" enclosed in the submission.

## Authors

**Sai Merriam** - *Original Developer*
**John Caton** - *Original Developer*
**Chelsea Liu** - *Original Developer*

## License

This project is licensed under the Course Policy of MCIT 591 in the School of Engineering and Applied Science, University of Pennsylvania.

## Further Development and Contributions

Teams and/or individuals who are interested to further develop or contribute to this project should contact the course administration team for MCIT 591 at the School of Engineering and Applied Science, University of Pennsylvania at mcitonline@seas.upenn.edu.

## Acknowledgments

We are grateful to Dr Arvind Bhusnurmath and Ms Anvi Dalal for their valuable guidance and feedback during the development phase of this project.

We thank James-D for providing guidance and example codes in relation to various JavaFx functions (Time Spinner and Zoom/Pan) and Mateus Viccari for providing example codes in relation to the Auto Complete ComboBox Listener function.