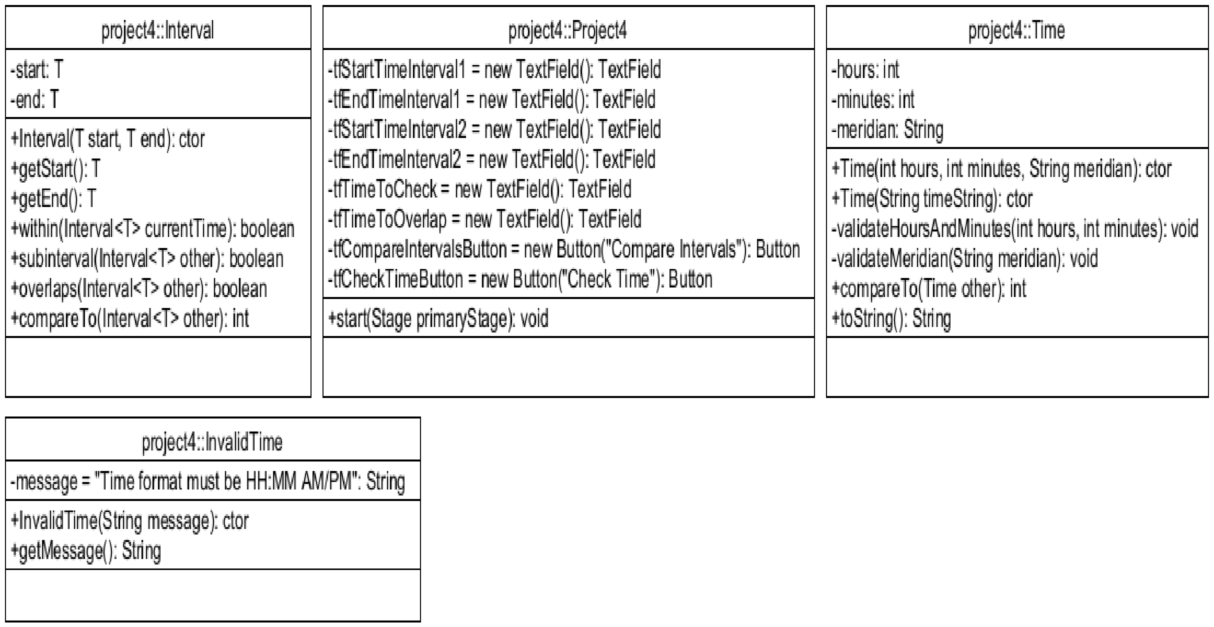
Victoria Lee
CMSC 215
7/9/2024

Project 4 Documentation

UML Class Diagrams and Package:



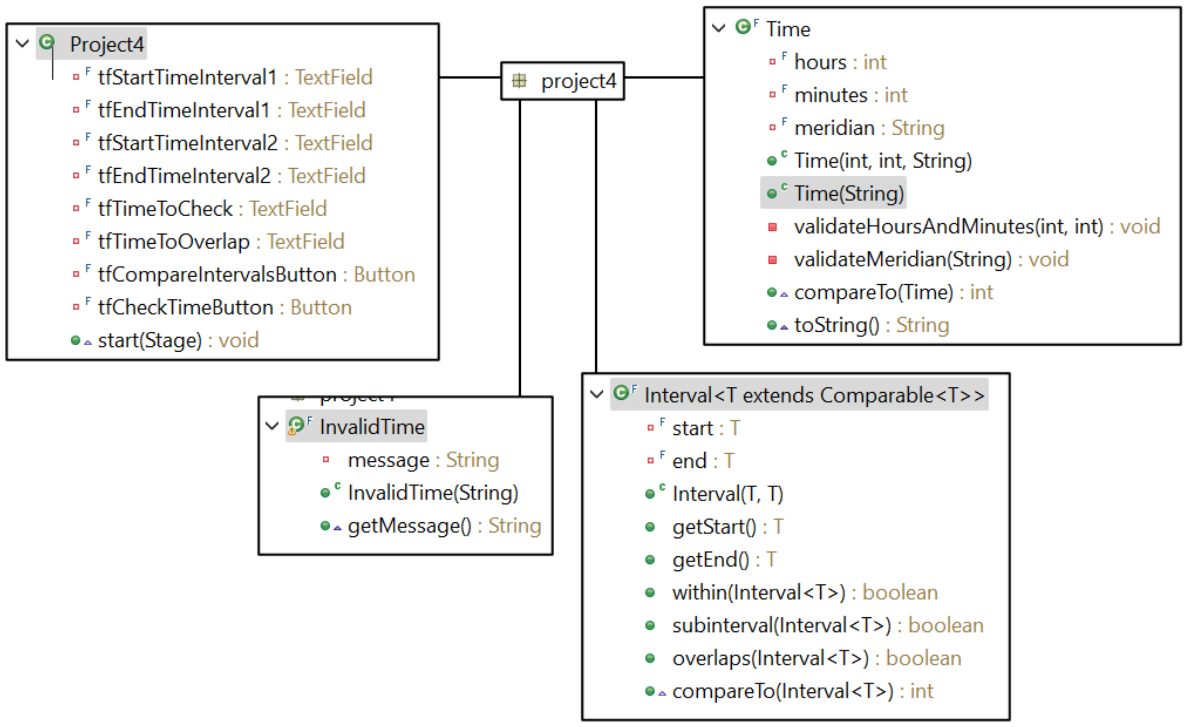
```
project4::Interval
-start: T
-end: T
+Interval(T start, T end): ctor
+getStart(): T
+getEnd(): T
+within(Interval<T> currentTime): boolean
+subinterval(Interval<T> other): boolean
+overlaps(Interval<T> other): boolean
+compareTo(Interval<T> other): int
```

```
project4::Project4
-tfStartTimeInterval1 = new TextField(): TextField
-tfEndTimeInterval1 = new TextField(): TextField
-tfStartTimeInterval2 = new TextField(): TextField
-tfEndTimeInterval2 = new TextField(): TextField
-tfTimeToCheck = new TextField(): TextField
-tfTimeToOverlap = new TextField(): TextField
-tfCompareIntervalsButton = new Button("Compare Intervals"): Button
-tfCheckTimeButton = new Button("Check Time"): Button
+start(Stage primaryStage): void
```

```
project4::Time
-hours: int
-minutes: int
-meridian: String
+Time(int hours, int minutes, String meridian): ctor
+Time(String timeString): ctor
-validateHoursAndMinutes(int hours, int minutes): void
-validateMeridian(String meridian): void
+compareTo(Time other): int
+toString(): String
```

```
project4::InvalidTime
-message = "Time format must be HH:MM AM/PM": String
+InvalidTime(String message): ctor
+getMessage(): String
```

Created UML Class Diagrams and Package:



```
project4

Project4
  tfStartTimeInterval1 : TextField
  tfEndTimeInterval1 : TextField
  tfStartTimeInterval2 : TextField
  tfEndTimeInterval2 : TextField
  tfTimeToCheck : TextField
  tfTimeToOverlap : TextField
  tfCompareIntervalsButton : Button
  tfCheckTimeButton : Button
  start(Stage) : void

Time
  hours : int
  minutes : int
  meridian : String
  Time(int, int, String)
  Time(String)
  validateHoursAndMinutes(int, int) : void
  validateMeridian(String) : void
  compareTo(Time) : int
  toString() : String

InvalidTime
  message : String
  InvalidTime(String)
  getMessage() : String

Interval<T extends Comparable<T>>
  start : T
  end : T
  Interval(T, T)
  getStart() : T
  getEnd() : T
  within(Interval<T>) : boolean
  subinterval(Interval<T>) : boolean
  overlaps(Interval<T>) : boolean
  compareTo(Interval<T>) : int
```

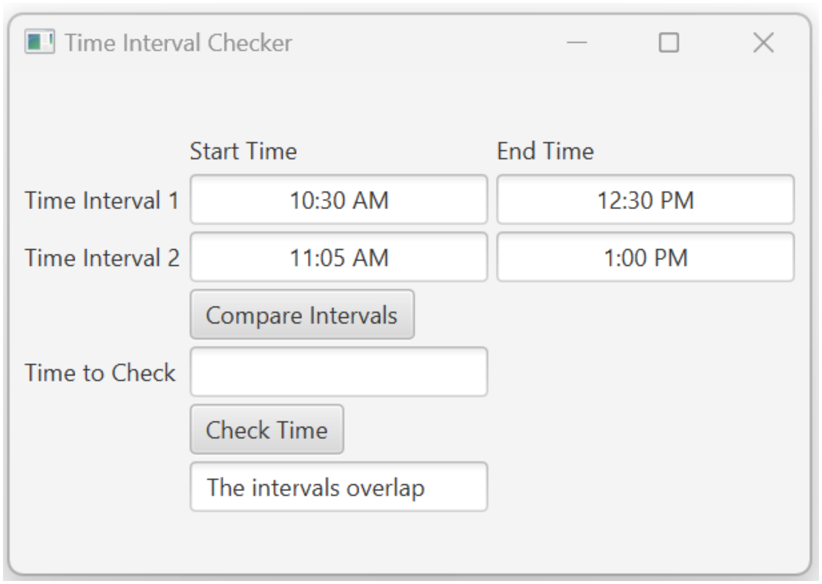
All Test Plans/Cases:

Test 1 (Given): Testing if the time zones from 10:30 AM-12:30 PM and 11:05 AM-1:00 PM overlaps.

Input:
Time Interval 1:
    Start Time: 10:30 AM
    End Time: 12:30 PM
Time Interval 2:
    Start Time: 11:05 AM
    End Time: 1:00 PM
Time to Check: Blank

Output:
Compare Intervals: The intervals overlap
Time to Check: Blank

Demo test below:


Time Interval Checker

	Start Time	End Time
Time Interval 1	10:30 AM	12:30 PM
Time Interval 2	11:05 AM	1:00 PM

Time to Check

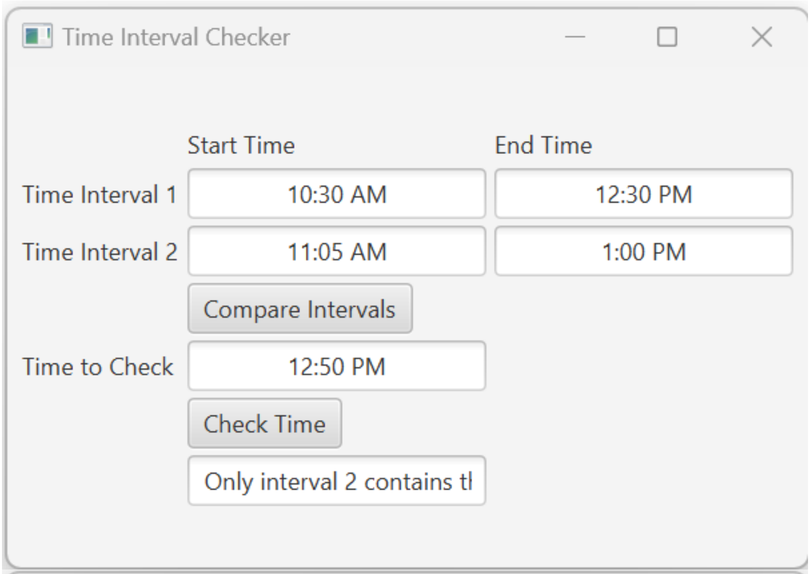
The intervals overlap

Test 2 (Given): Testing Time to Check if 12:50 PM is in one of the intervals which is interval 2.

Input:
Time Interval 1:
    Start Time: 10:30 AM
    End Time: 12:30 PM
Time Interval 2:
    Start Time: 11:05 AM
    End Time: 1:00 PM
Time to Check: 12:50 PM

Output:
Compare Intervals: Blank (Note: you can check, I did not test this but it should work)
Time to Check: Only interval 2 contains the time 12:50 PM

Demo test below:



Time Interval Checker

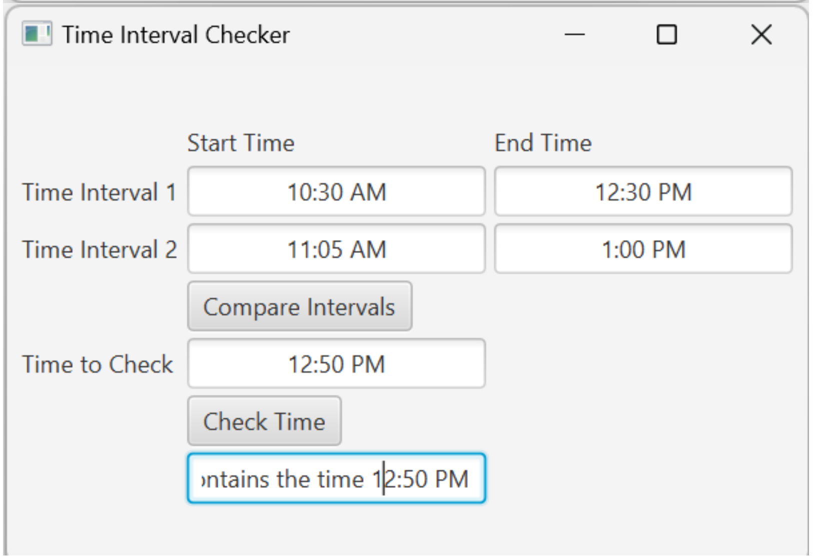
	Start Time	End Time
Time Interval 1	10:30 AM	12:30 PM
Time Interval 2	11:05 AM	1:00 PM

Compare Intervals

Time to Check: 12:50 PM

Check Time

Only interval 2 contains tl



Time Interval Checker

	Start Time	End Time
Time Interval 1	10:30 AM	12:30 PM
Time Interval 2	11:05 AM	1:00 PM

Compare Intervals

Time to Check: 12:50 PM

Check Time

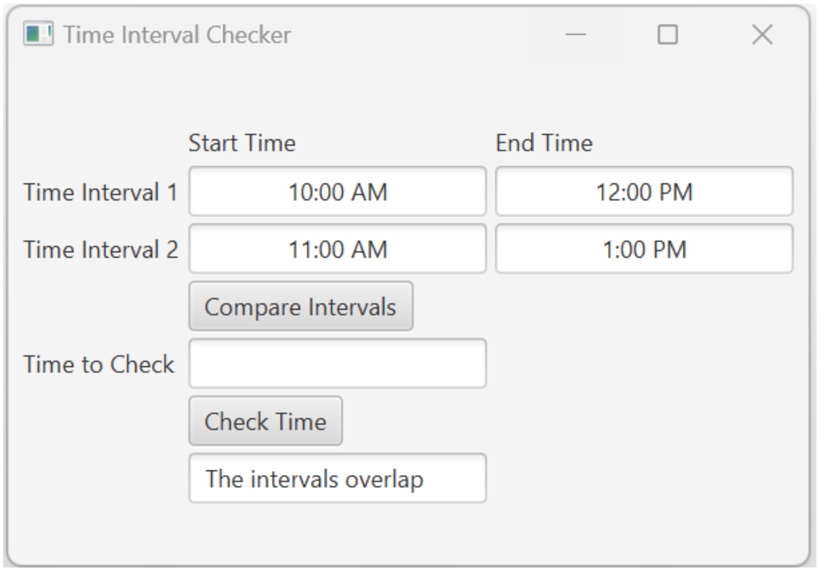
ıntains the time 12:50 PM

Test 3 (Given): Testing again if Time overlaps with a different time set.

Input:
Time Interval 1:
    Start Time: 10:00 AM
    End Time: 12:00 PM
Time Interval 2:
    Start Time: 11:00 AM
    End Time: 1:00 PM
Time to Check: Blank

Output:
Compare Intervals: The intervals overlap
Time to Check: Blank

Demo test below:



Time Interval 1	10:00 AM	12:00 PM
Time Interval 2	11:00 AM	1:00 PM
Time to Check		

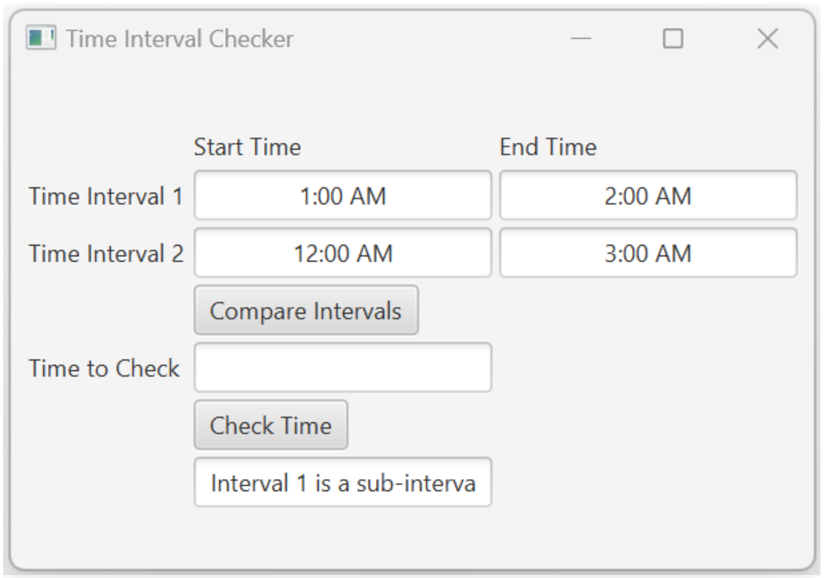
The intervals overlap

Test 4 (Given): Testing if the time intervals 1 is a subinterval of interval 2 with different times.

Input:
Time Interval 1:
    Start Time: 1:00 AM
    End Time: 2:00 AM
Time Interval 2:
    Start Time: 12:00 AM
    End Time: 3:00 AM
Time to Check: Blank

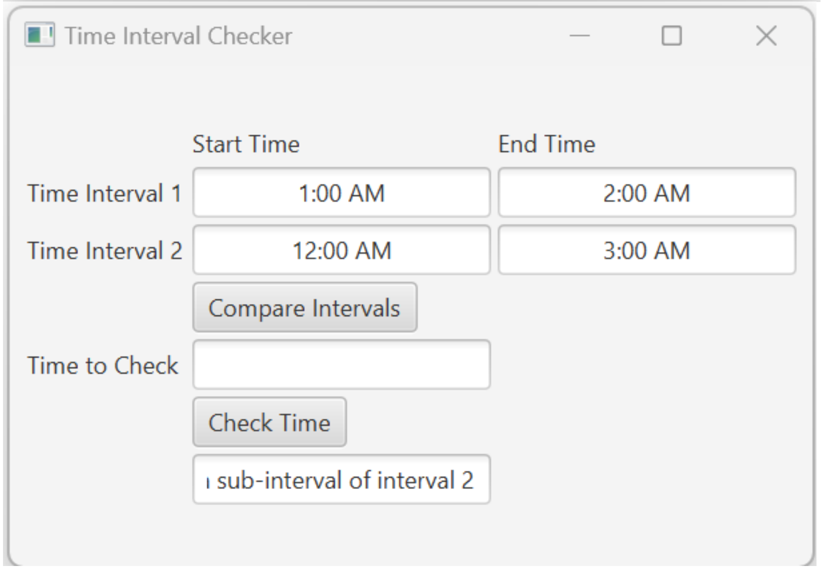
Output:
Compare Intervals: Interval 1 is a sub-interval of interval 2
Time to Check: Blank

Demo test below:



	Start Time	End Time
Time Interval 1	1:00 AM	2:00 AM
Time Interval 2	12:00 AM	3:00 AM
Time to Check		

Interval 1 is a sub-interva



	Start Time	End Time
Time Interval 1	1:00 AM	2:00 AM
Time Interval 2	12:00 AM	3:00 AM
Time to Check		

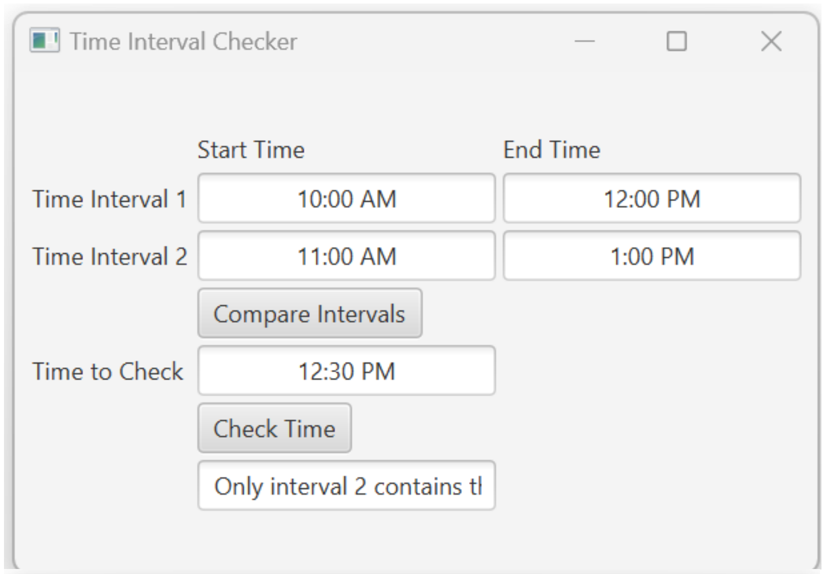
ı sub-interval of interval 2

Test 5 (Given): Testing if the Time to Check that one of the intervals contains 12:30 PM.

Input:
Time Interval 1:
    Start Time: 10:00 AM
    End Time: 12:00 PM
Time Interval 2:
    Start Time: 11:00 AM
    End Time: 1:00 PM
Time to Check: 12:30 PM

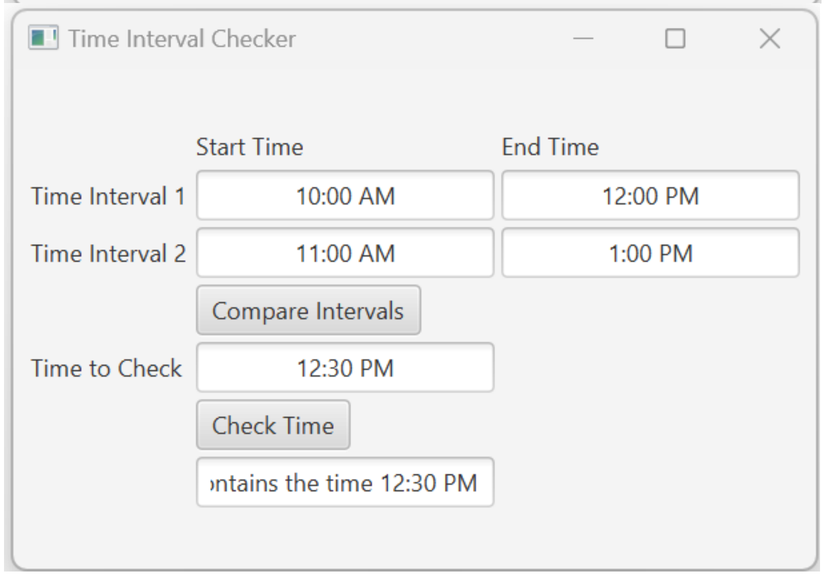
Output:
Compare Intervals: Blank (Note: you can check, I did not test this but it should work)
Time to Check: Only interval 2 contains the time 12:30 PM

Demo test below:



	Start Time	End Time
Time Interval 1	10:00 AM	12:00 PM
Time Interval 2	11:00 AM	1:00 PM
Time to Check	12:30 PM	

Only interval 2 contains tl



	Start Time	End Time
Time Interval 1	10:00 AM	12:00 PM
Time Interval 2	11:00 AM	1:00 PM
Time to Check	12:30 PM	

›ntains the time 12:30 PM

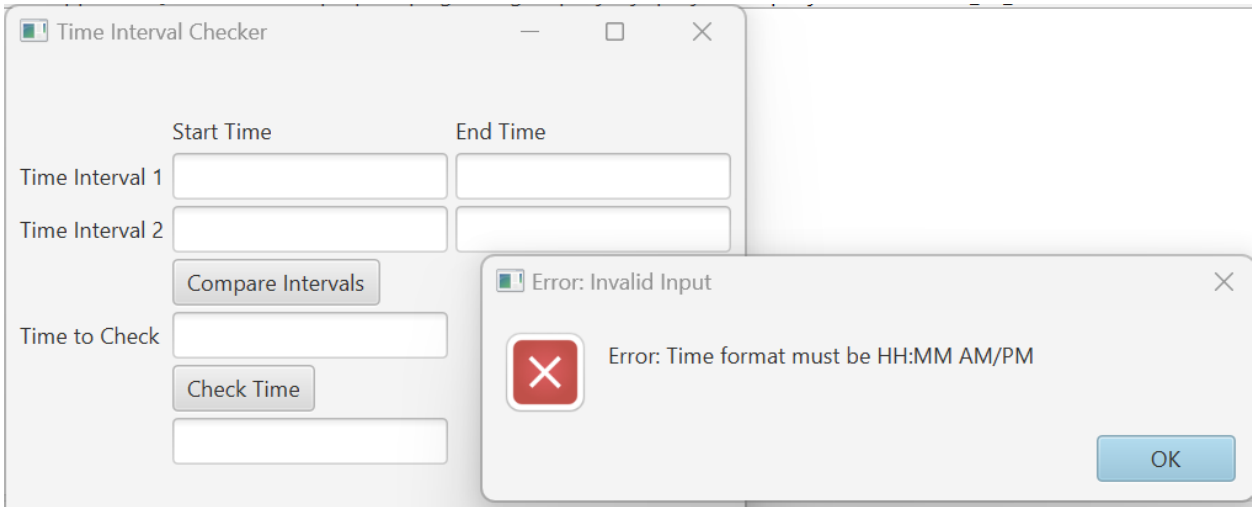
Test 6 (Created): Testing for no/blank user inputs. It should pop up an error alert message.

Input:
Time Interval 1:
    Start Time: Blank
    End Time: Blank
Time Interval 2:
    Start Time: Blank
    End Time: Blank
Time to Check: Blank

Output:
Compare Intervals: Blank
Time to Check: Blank

Alert:
Error: Invalid Input
Error: Time format must be HH:MM AM/PM

Demo test below:



Error: Time format must be HH:MM AM/PM

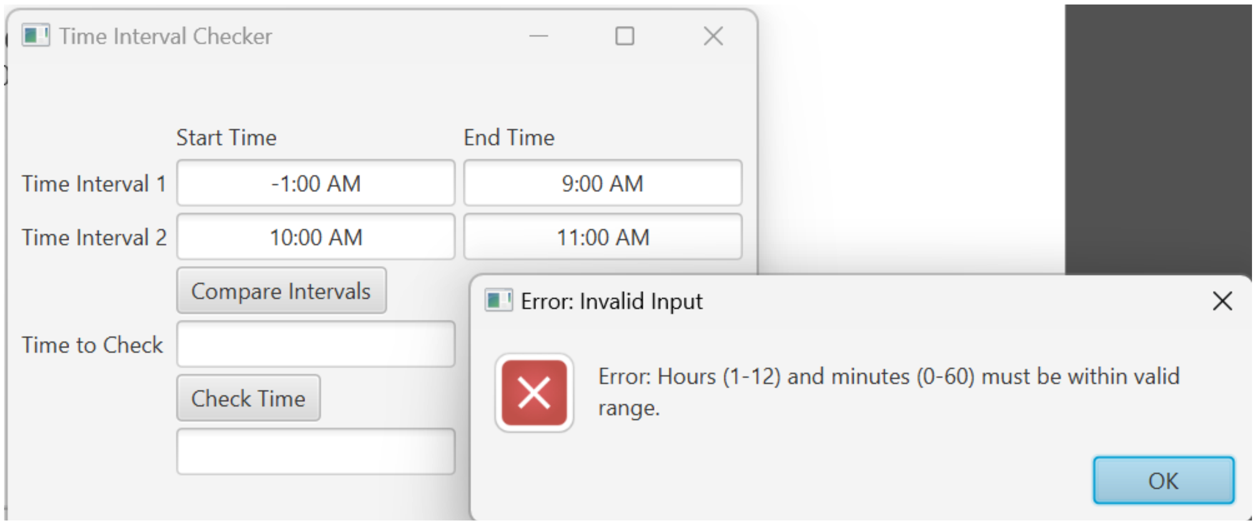
Test 7 (Created): Testing for negative time user inputs. It should pop up an error alert message.

Input:
Time Interval 1:
Start Time: -1:00 AM
End Time: 9:00 AM
Time Interval 2:
Start Time: 10:00 AM
End Time: 11:00 AM
Time to Check: Blank

Output:
Compare Intervals: The intervals overlap
Time to Check: Blank

Alert:
Error: Invalid Input
Error: Hours (1-12) and minutes (0-60) must be within valid range.

Demo test below:



	Start Time	End Time
Time Interval 1	-1:00 AM	9:00 AM
Time Interval 2	10:00 AM	11:00 AM

Time to Check

Error: Invalid Input

Error: Hours (1-12) and minutes (0-60) must be within valid range.

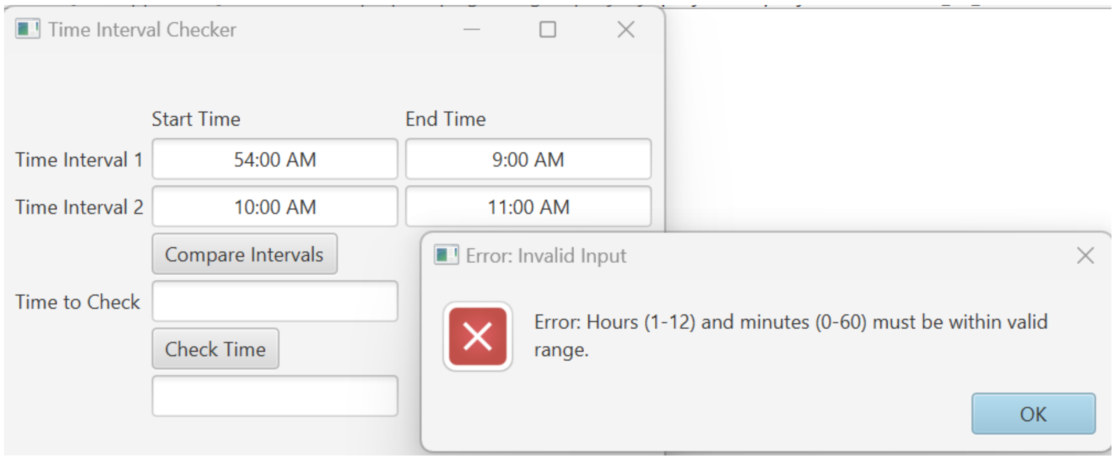
Test 8 (Created): Testing for hours greater than 12 from user inputs and minutes equal to or greater than 60. It should pop up an error alert message.

Input:
Time Interval 1:
    Start Time: 54:00 AM
    End Time: 9:00 AM
Time Interval 2:
    Start Time: 10:00 AM
    End Time: 11:00 AM
Time to Check: Blank

Output:
Compare Intervals: Blank
Time to Check: Blank

Alert:
Error: Invalid Input
Error: Hours (1-12) and minutes (0-60) must be within valid range.

Demo test below:

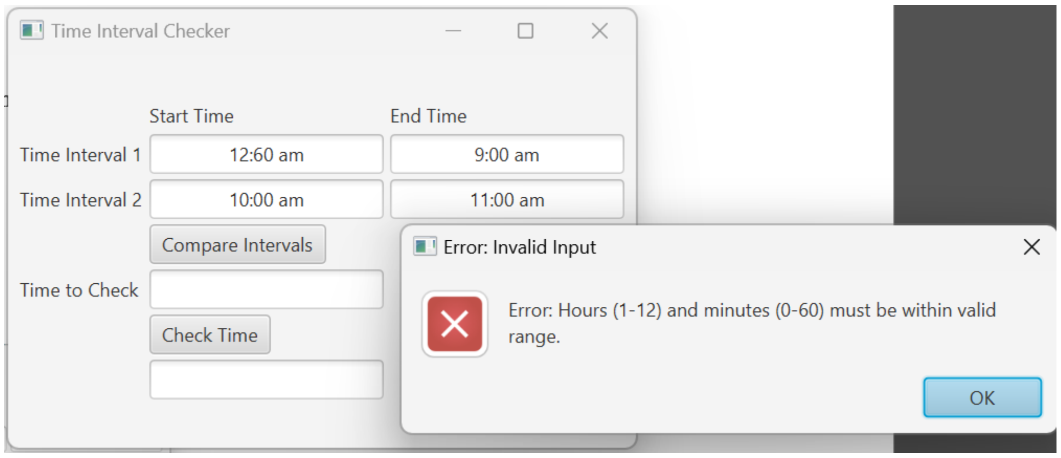


Time Interval Checker

	Start Time	End Time
Time Interval 1	54:00 AM	9:00 AM
Time Interval 2	10:00 AM	11:00 AM

Error: Invalid Input

Error: Hours (1-12) and minutes (0-60) must be within valid range.



Time Interval Checker

	Start Time	End Time
Time Interval 1	12:60 am	9:00 am
Time Interval 2	10:00 am	11:00 am

Error: Invalid Input

Error: Hours (1-12) and minutes (0-60) must be within valid range.

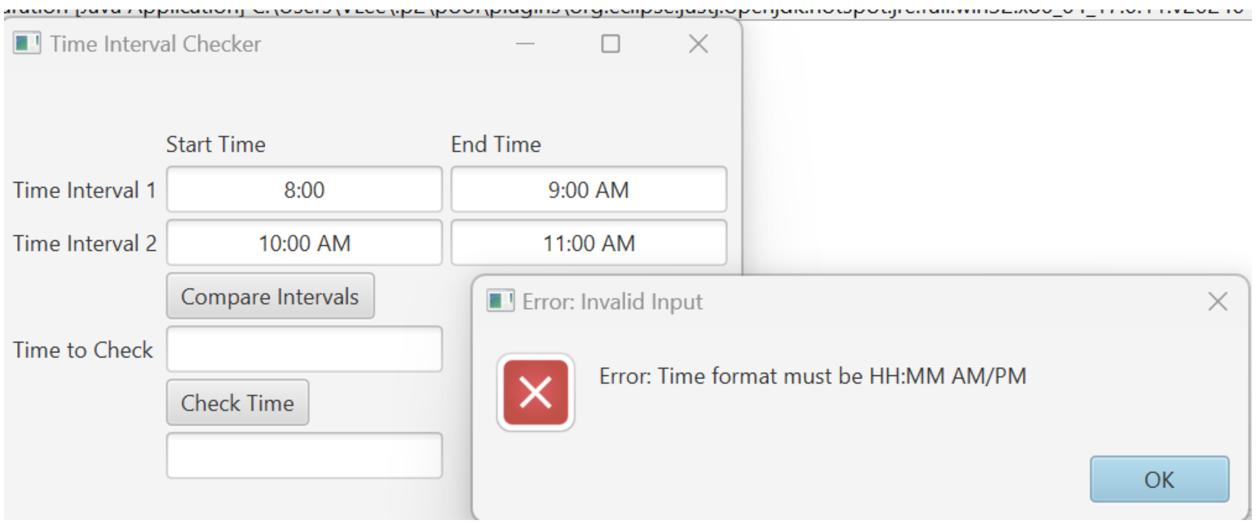
Test 9 (Created): Testing for if the user does not input properly with the format HH:MM AM/PM. If it is missing something it should pop up an error or alert message.

Input:
Time Interval 1:
Start Time: 8:00
End Time: 9:00 AM
Time Interval 2:
Start Time: 10:00 AM
End Time: 11:00 AM
Time to Check: Blank

Output:
Compare Intervals: Blank
Time to Check: Blank

Alert:
Error: Invalid Input
Error: Time format must be HH:MM AM/PM

Demo test below:



Time Interval Checker

	Start Time	End Time
Time Interval 1	8:00	9:00 AM
Time Interval 2	10:00 AM	11:00 AM

Time to Check

Error: Invalid Input

Error: Time format must be HH:MM AM/PM

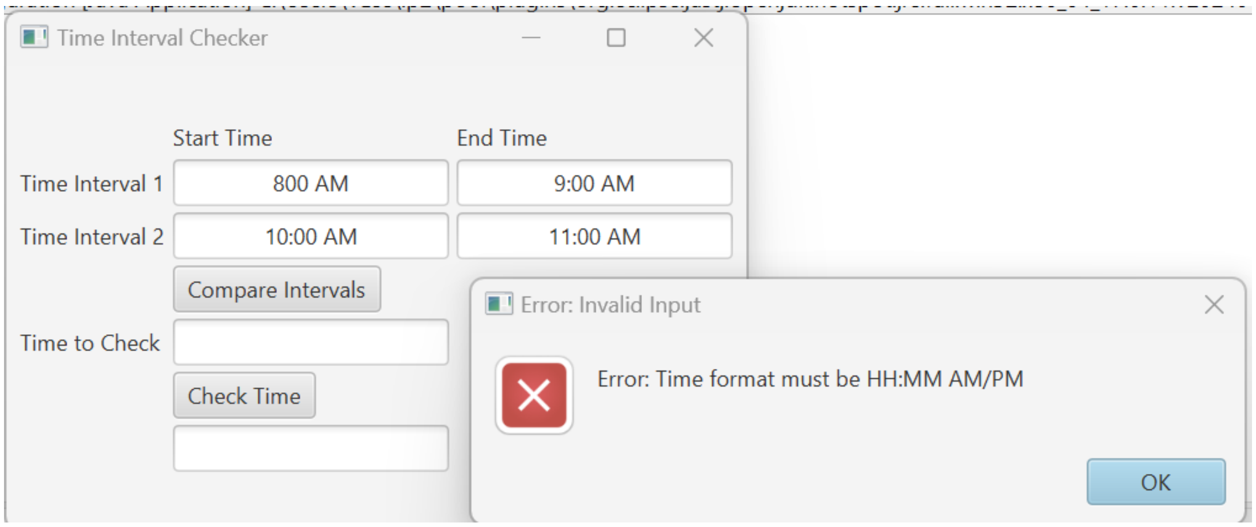
Test 10 (Created): Testing for invalid input of the format HH:MM AM/PM with 800 AM. It should pop up an error alert message.

Input:
Time Interval 1:
    Start Time: 800 AM
    End Time: 9:00 AM
Time Interval 2:
    Start Time: 10:00 AM
    End Time: 11:00 AM
Time to Check: Blank

Output:
Compare Intervals: Blank
Time to Check: Blank

Alert:
Error: Invalid Input
Error: Time format must be HH:MM AM/PM

Demo test below:


Time Interval Checker

	Start Time	End Time
Time Interval 1	800 AM	9:00 AM
Time Interval 2	10:00 AM	11:00 AM

Error: Invalid Input

Error: Time format must be HH:MM AM/PM

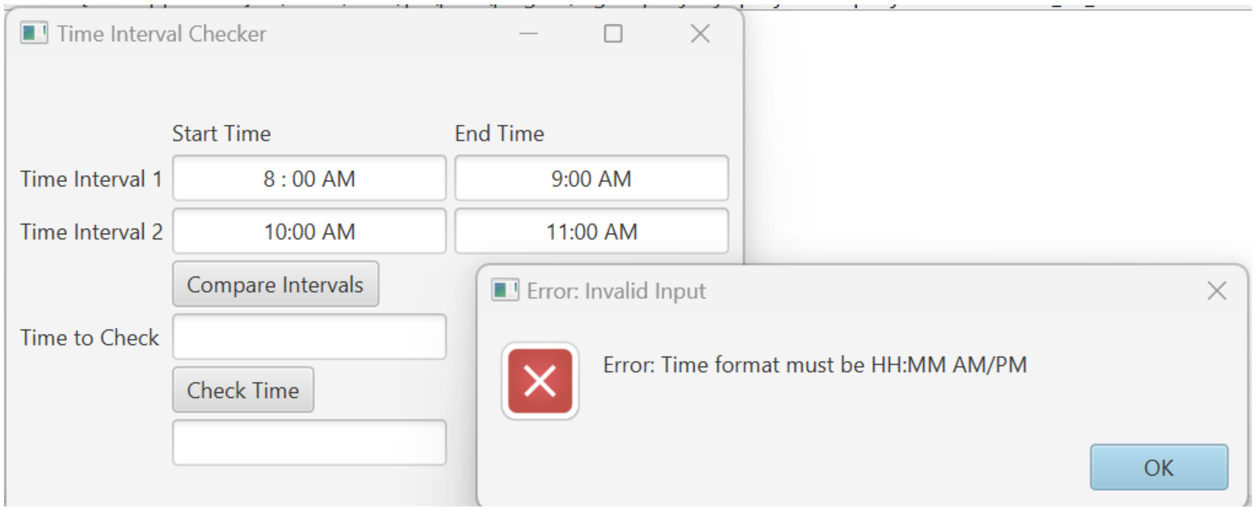
Test 11 (Created): Testing for invalid input of the format HH:MM AM/PM with 8 : 00 AM which is adding extra space in the time. It should pop up an error alert message.

Input:
Time Interval 1:
    Start Time: 8 : 00 AM
    End Time: 9:00 AM
Time Interval 2:
    Start Time: 10:00 AM
    End Time: 1:00 AM
Time to Check: Blank

Output:
Compare Intervals: Blank
Time to Check: Blank

Alert:
Error: Invalid Input
Error: Time format must be HH:MM AM/PM

Demo test below:



Time Interval Checker

	Start Time	End Time
Time Interval 1	8 : 00 AM	9:00 AM
Time Interval 2	10:00 AM	11:00 AM

Time to Check:

Error: Invalid Input

Error: Time format must be HH:MM AM/PM

Test 12 (Created): Testing for invalid input of the format HH:MM AM/PM if the user inputs any symbols. It should pop up an error alert message.

Input:

Time Interval 1:

Start Time: ~!@#$%^&*()?

End Time: 9:00 AM

Time Interval 2:

Start Time: 10:00 AM

End Time: 11:00 AM

Time to Check: Blank

Output:

Compare Intervals: Blank

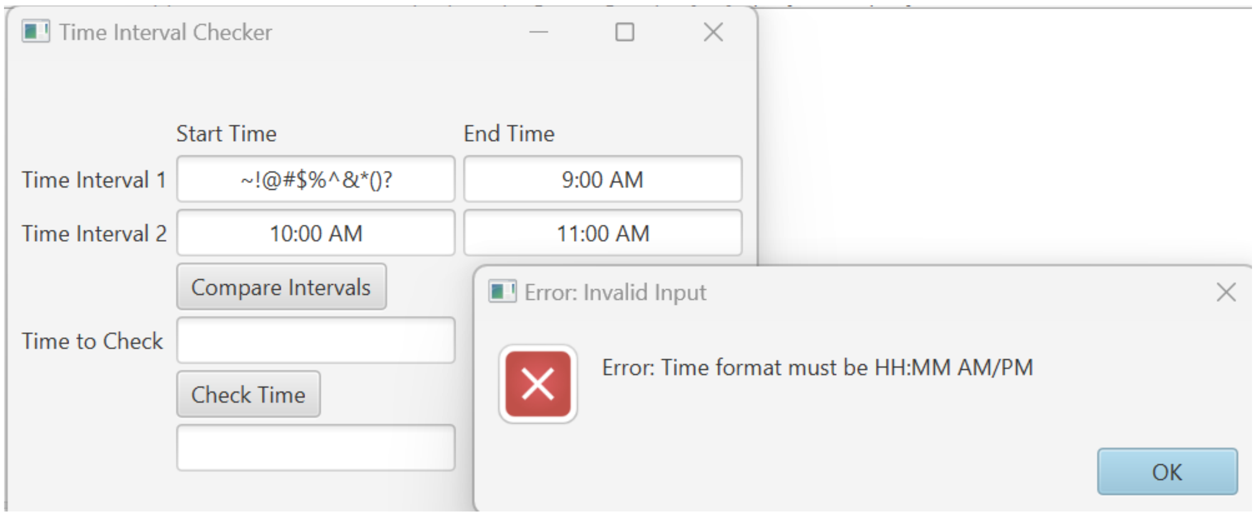
Time to Check: Blank

Alert:

Error: Invalid Input

Error: Time format must be HH:MM AM/PM

Demo test below:



~!@#$%^&*()?

9:00 AM

10:00 AM

11:00 AM

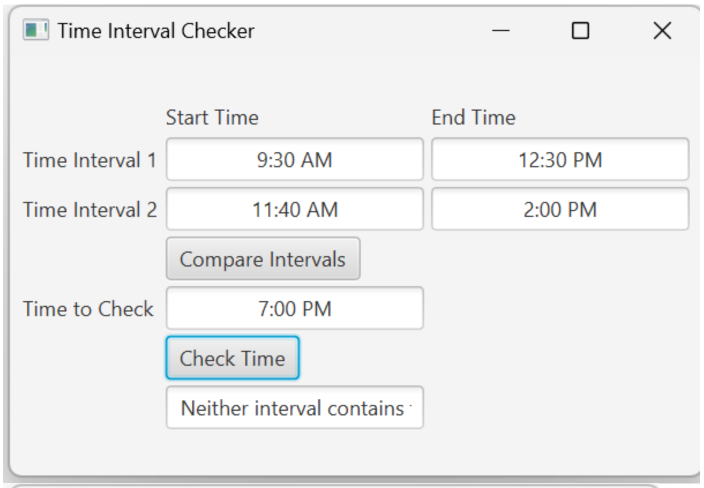
Error: Time format must be HH:MM AM/PM

Test 13 (Created): Testing for the Time to Check that is out of the boundaries for the max intervals. For instance, min is 9:30 am and max is 2pm. It will check if 7:00 pm is in both intervals.

Input:
Time Interval 1:
    Start Time: 9:30 AM
    End Time: 12:30 PM
Time Interval 2:
    Start Time: 11:40 AM
    End Time: 2:00 PM
Time to Check: 7:00 PM

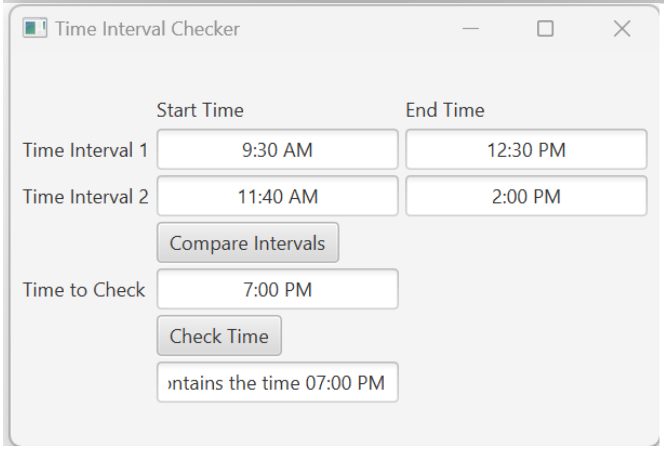
Output:
Compare Intervals: Blank (Note: you can check, I did not test this but it should work)
Time to Check: Neither interval contains the time 07:00 PM

Demo test below:



	Start Time	End Time
Time Interval 1	9:30 AM	12:30 PM
Time Interval 2	11:40 AM	2:00 PM
Time to Check	7:00 PM	

Neither interval contains ·



	Start Time	End Time
Time Interval 1	9:30 AM	12:30 PM
Time Interval 2	11:40 AM	2:00 PM
Time to Check	7:00 PM	

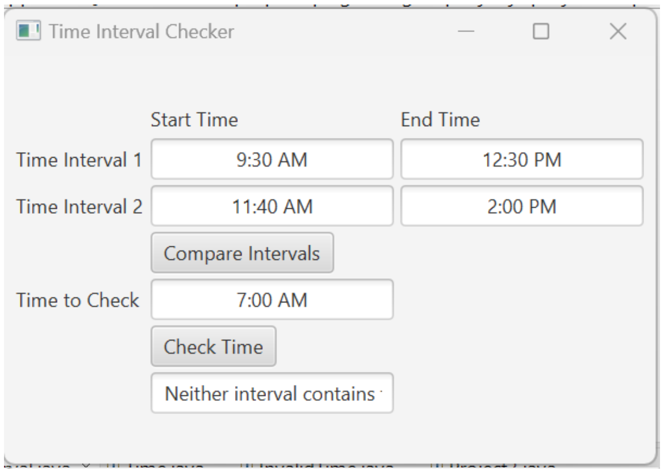
›ntains the time 07:00 PM

Test 14 (Created): Testing for the Time to Check that is out of the boundaries for the min intervals. For instance, min is 9:30 am and max is 2pm. It will check if 7:00 am is in both intervals.

Input:
Time Interval 1:
	Start Time: 9:30 AM
	End Time: 12:30 PM
Time Interval 2:
	Start Time: 11:40 AM
	End Time: 2:00 PM
Time to Check: 7:00 AM

Output:
Compare Intervals: Blank (Note: you can check, I did not test this but it should work)
Time to Check: Neither interval contains the time 07:00 AM

Demo test below:



Time Interval Checker

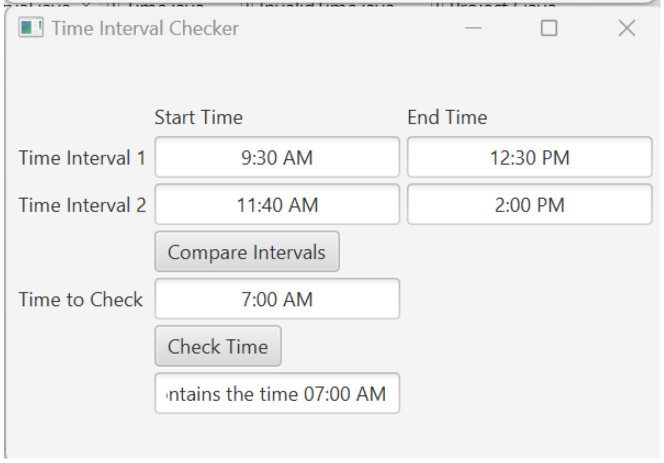
	Start Time	End Time
Time Interval 1	9:30 AM	12:30 PM
Time Interval 2	11:40 AM	2:00 PM

Compare Intervals

Time to Check: 7:00 AM

Check Time

Neither interval contains



Time Interval Checker

	Start Time	End Time
Time Interval 1	9:30 AM	12:30 PM
Time Interval 2	11:40 AM	2:00 PM

Compare Intervals

Time to Check: 7:00 AM

Check Time

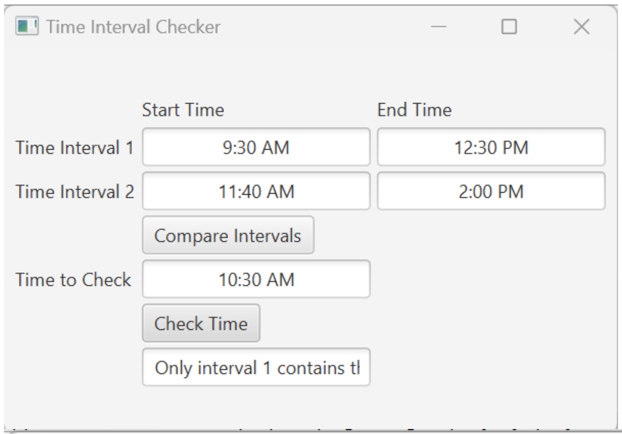
ntains the time 07:00 AM

Test 15 (Created): Testing for the Time to Check that has a time in one of the intervals which is interval 1.

Input:
Time Interval 1:
    Start Time: 9:30 AM
    End Time: 12:30 PM
Time Interval 2:
    Start Time: 11:40 AM
    End Time: 2:00 PM
Time to Check: 10:30 AM

Output:
Compare Intervals: Blank (Note: you can check, I did not test this but it should work)
Time to Check: Only interval 1 contains the time 10:30 AM

Demo test below:


Time Interval Checker

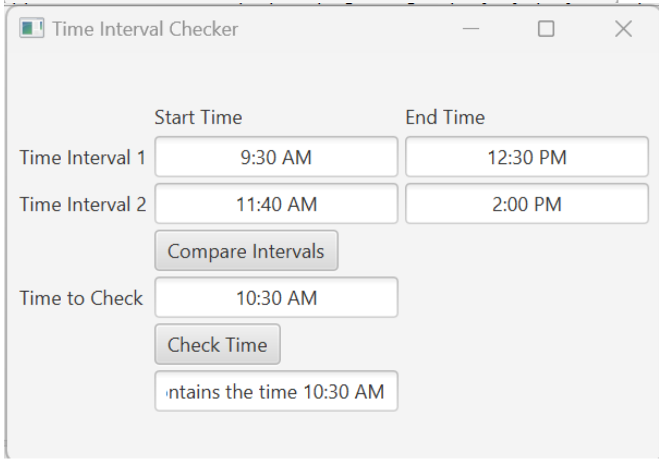
	Start Time	End Time
Time Interval 1	9:30 AM	12:30 PM
Time Interval 2	11:40 AM	2:00 PM

Compare Intervals

Time to Check: 10:30 AM

Check Time

Only interval 1 contains tl


Time Interval Checker

	Start Time	End Time
Time Interval 1	9:30 AM	12:30 PM
Time Interval 2	11:40 AM	2:00 PM

Compare Intervals

Time to Check: 10:30 AM

Check Time

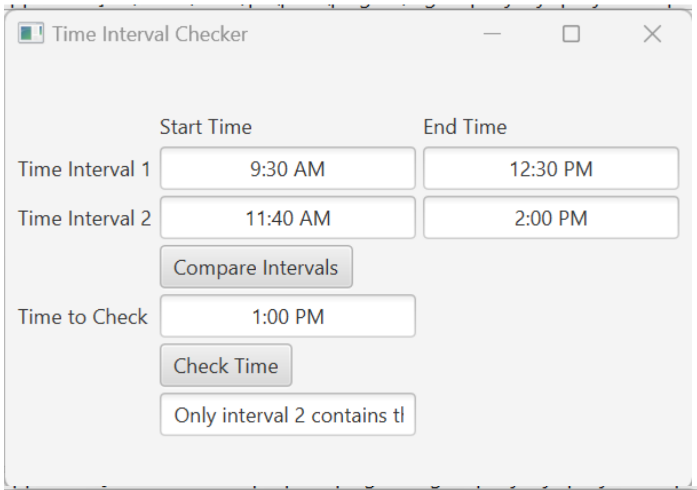
ntains the time 10:30 AM

Test 16 (Created): Testing for the Time to Check that has a time in one of the intervals which is interval 2.

Input:
Time Interval 1:
    Start Time: 9:30 AM
    End Time: 12:30 PM
Time Interval 2:
    Start Time: 11:40 AM
    End Time: 2:00 PM
Time to Check: 1:00 PM

Output:
Compare Intervals: Blank (Note: you can check, I did not test this but it should work)
Time to Check: Only interval 2 contains the time 01:00 PM

Demo test below:



Time Interval Checker

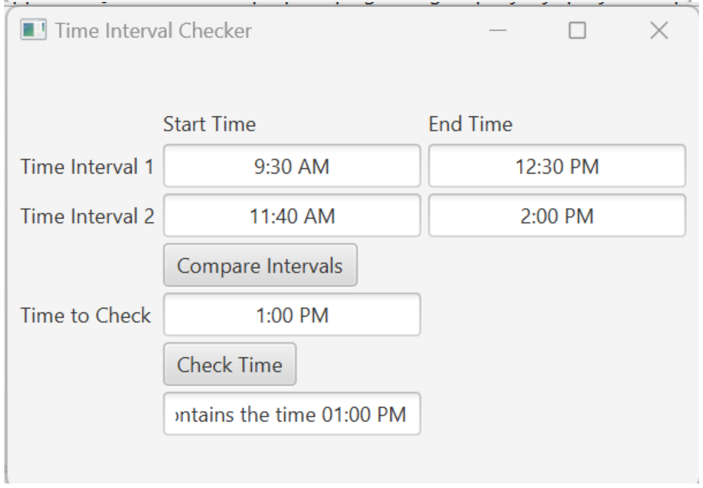
	Start Time	End Time
Time Interval 1	9:30 AM	12:30 PM
Time Interval 2	11:40 AM	2:00 PM

Compare Intervals

Time to Check: 1:00 PM

Check Time

Only interval 2 contains tl



Time Interval Checker

	Start Time	End Time
Time Interval 1	9:30 AM	12:30 PM
Time Interval 2	11:40 AM	2:00 PM

Compare Intervals

Time to Check: 1:00 PM

Check Time

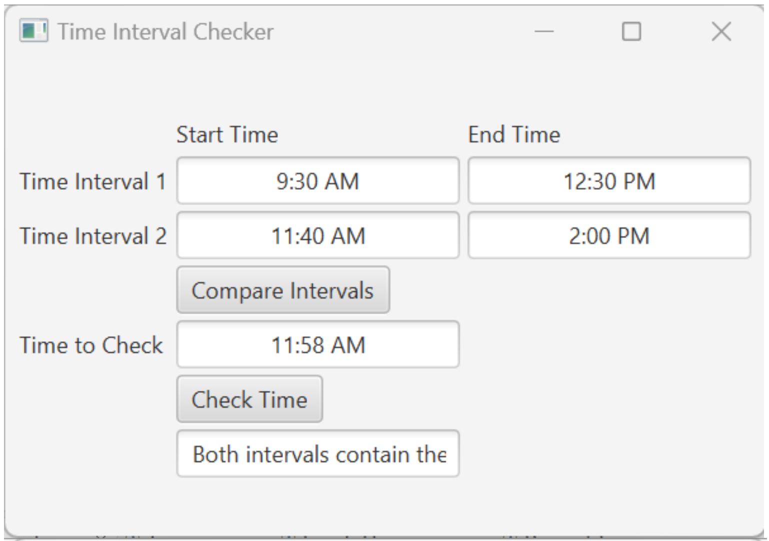
ıntains the time 01:00 PM

Test 17 (Created): Testing metric units with my own choice of values.

Input:
Time Interval 1:
    Start Time: 9:30 AM
    End Time: 12:30 PM
Time Interval 2:
    Start Time: 11:40 AM
    End Time: 2:00 PM
Time to Check: 11:58 AM

Output:
Compare Intervals: Blank (you can check, I did not test this but it should work)
Time to Check: Both intervals contain the time 11:58 AM

Demo test below:

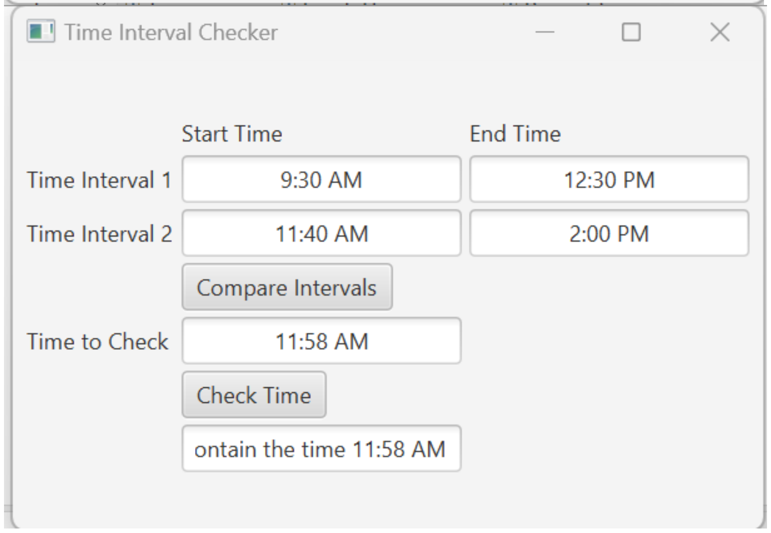


Time Interval Checker

	Start Time	End Time
Time Interval 1	9:30 AM	12:30 PM
Time Interval 2	11:40 AM	2:00 PM

Time to Check: 11:58 AM

Both intervals contain the



Time Interval Checker

	Start Time	End Time
Time Interval 1	9:30 AM	12:30 PM
Time Interval 2	11:40 AM	2:00 PM

Time to Check: 11:58 AM

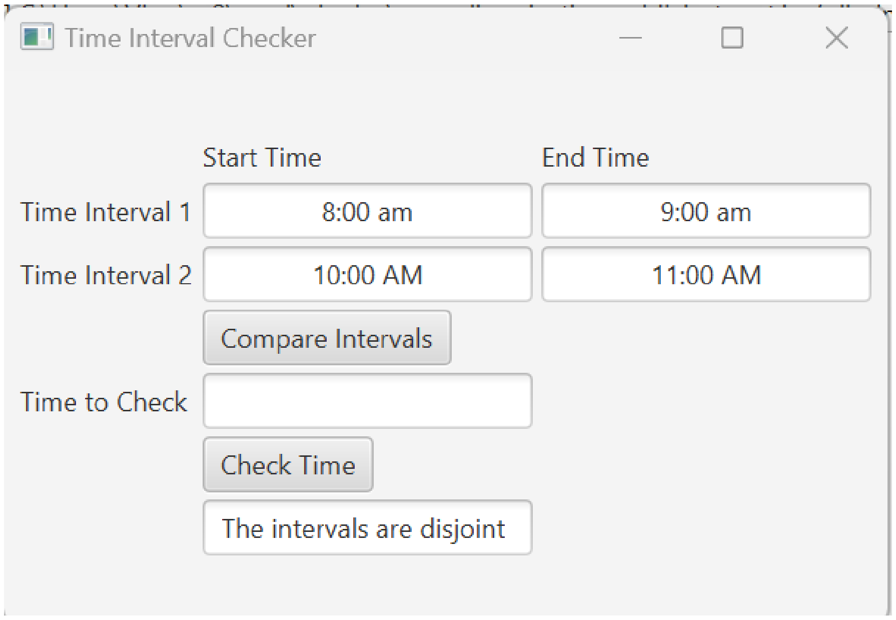
ontain the time 11:58 AM

Test 18 (Created): Testing if the intervals in different times are disjoint for interval 1 with all AM and interval 2 with all AM. I am also testing that it converts lowercase and uppercase properly with AM/PM.

Input:
Time Interval 1:
    Start Time: 8:00 am
    End Time: 9:00 am
Time Interval 2:
    Start Time: 10:00 AM
    End Time: 11:00 AM
Time to Check: Blank

Output:
Compare Intervals: The intervals are disjoint
Time to Check: Blank

Demo test below:



	Start Time	End Time
Time Interval 1	8:00 am	9:00 am
Time Interval 2	10:00 AM	11:00 AM

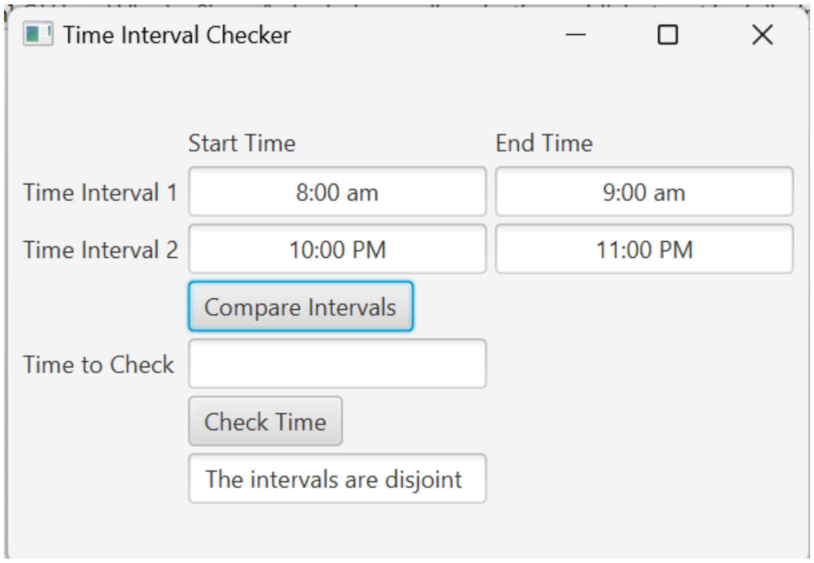
The intervals are disjoint

Test 19 (Created): Testing if the intervals in different times are disjoint for interval 1 with all AM and interval 2 with all PM. I am also testing that it converts lowercase and uppercase properly with AM/PM.

Input:
Time Interval 1:
 Start Time: 8:00 am
 End Time: 9:00 am
Time Interval 2:
 Start Time: 10:00 PM
 End Time: 11:00 PM
Time to Check: Blank

Output:
Compare Intervals: The intervals are disjoint
Time to Check: Blank

Demo test below:



	Start Time	End Time
Time Interval 1	8:00 am	9:00 am
Time Interval 2	10:00 PM	11:00 PM

The intervals are disjoint

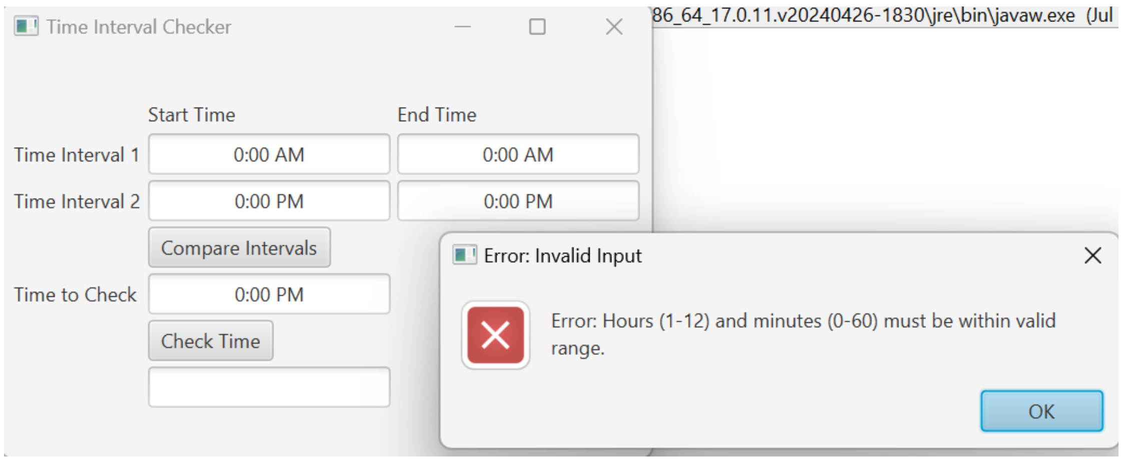
Test 20 (Created): Testing for invalid input of the format HH:MM AM/PM if the user inputs all 0's with AM/PM sign. It should pop up an error alert message. This will prevent the user from inputting any values and NaN values and it will pop up an error.

Input:
Time Interval 1:
Start Time: 0:00 AM
End Time: 0:00 AM
Time Interval 2:
Start Time: 0:00 PM
End Time: 0:00 PM
Time to Check: 0:00 AM

Output:
Compare Intervals: Blank
Time to Check: Blank

Alert:
Error: Invalid Input
Error: Hours (1-12) and minutes (0-60) must be within valid range.

Demo test below:



	Start Time	End Time
Time Interval 1	0:00 AM	0:00 AM
Time Interval 2	0:00 PM	0:00 PM
Time to Check	0:00 PM	

Error: Hours (1-12) and minutes (0-60) must be within valid range.

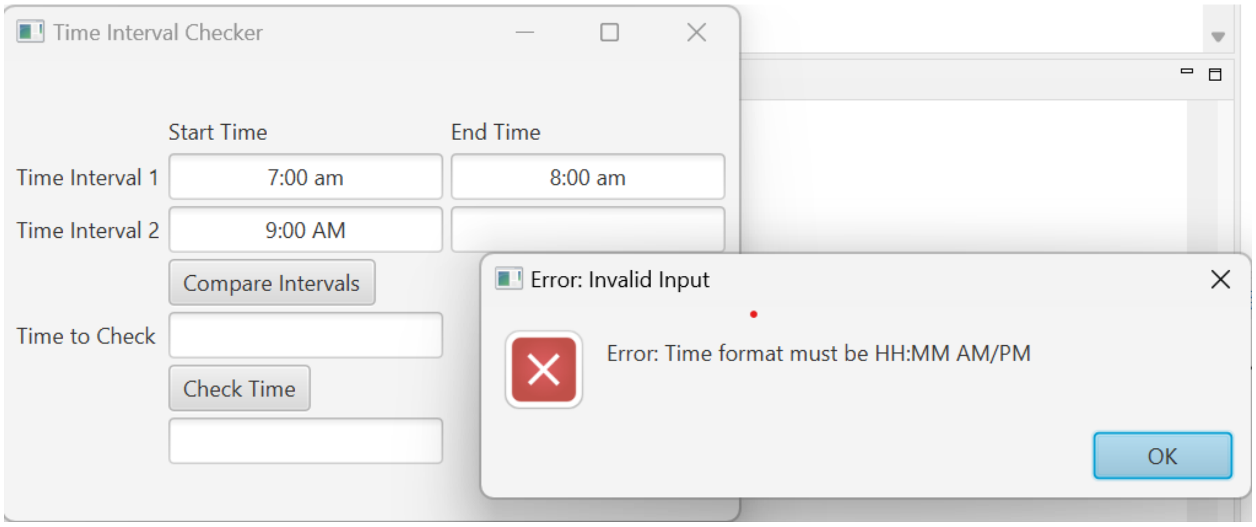
Test 21 (Created): Testing for invalid input of one blank while the rest has inputs. It is in the format HH:MM AM/PM. It should pop up an error alert message. This will prevent the user from inputting any values and NaN values and it will pop up an error.

Input:
Time Interval 1:
Start Time: 7:00 AM
End Time: 8:00 AM
Time Interval 2:
Start Time: 9:00 AM
End Time:
Time to Check:

Output:
Compare Intervals: Blank
Time to Check: Blank

Alert:
Error: Invalid Input
Error: Time format must be HH:MM AM/PM.

Demo test below:



Time Interval Checker

	Start Time	End Time
Time Interval 1	7:00 am	8:00 am
Time Interval 2	9:00 AM	

Error: Invalid Input

Error: Time format must be HH:MM AM/PM

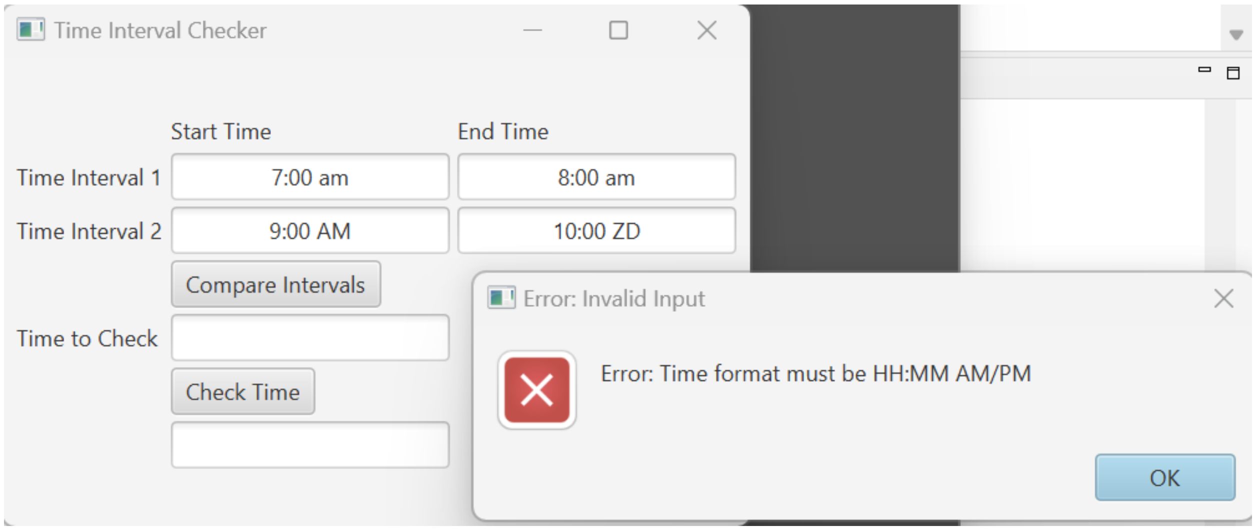
Test 23 (Created): Testing for invalid input if the AM/PM is not inputted and it is some other letter. It should pop up an error alert message. This will prevent the user from inputting any values and NaN values and it will pop up an error.

Input:
Time Interval 1:
  Start Time: 7:00 AM
  End Time: 8:00 AM
Time Interval 2:
  Start Time: 9:00 AM
  End Time: 10:00 ZD
Time to Check:

Output:
Compare Intervals: Blank
Time to Check: Blank

Alert:
Error: Invalid Input
Error: Time format must be HH:MM AM/PM.

Demo test below:



	Start Time	End Time
Time Interval 1	7:00 am	8:00 am
Time Interval 2	9:00 AM	10:00 ZD

Error: Time format must be HH:MM AM/PM

Lessons Learned (brief paragraphs):

To achieve my project goals, I learned how to create a project, classes, immutable classes, generics, try/catch, throw exceptions, methods, functions, and GUI generics/interfaces with JavaFX. I learned that the Time Interval Checker program with a GUI interface allowed me to comprehend the significant principles of object-oriented programming and software development. A class defines all the attributes an object can have and methods that define the object's functionality. A subclass inherits the properties and behaviors of another class. Immutable classes in Java mean that once an object is created, we cannot change its content. Generics allow you to parameterize types to enable errors to be detected at compile time rather than at runtime. When defining a class or a method with generic types, the compiler replaces them with concrete types. Utilizing Generics allows the programmer to specify allowable types of objects that the class or method can work with. The Labeled class is the base class for Label, Button, etc. The ButtonBase class defines the onAction property for specifying a handler for action events. The TextInputControl class is for TextField which fires an action event if you code it. And so, the main lesson learned from the Project4 class is that it is responsible for creating the user interface GUI. This interface includes input fields for time intervals 1 and 2, a time checker, and buttons to compare if the time overlaps, subintervals, intervals, and if the time checker is in any of the intervals. Moreover, these immutable classes are Interval, Time, and InvalidTime. Interval is responsible for passing objects of this type to have a start and end of the generic type of the parameter of the intervals. It compares if it is within, subinterval, or overlapping with each other. Time is responsible for containing the string input and splitting it into two integer instance variables for the hours and minutes and one additional variable for the meridian, AM/PM. This compares proper time and prints the format through toString() properly. InvalidTime is responsible for checking the valid object time and it prints out a message if invalid. The main goals in this class's design include its constructor, which initializes the object, and its methods so that the objects can compare the input time intervals. These lessons helped me understand good modular design for developing applications with JavaFX or GUI with generics. In real life, users can utilize this application to Compare Times. Project 4 is about comprehending the generics, and implementing the comparisons was the trickiest. Overall, I learned to apply it to Project 4 with the lessons about try/catch, exceptions/throws, classes, subclasses, generics, packages, importing libraries, constructors, GUI/JavaFX, and immutable classes.

My design approach was to create the Interval class, Time, and InvalidTime before creating the Project4 class. I started with a Bottom-Up Design when building the code, but then debugged the code through a Top-Down Design. I followed the instructions on what is asked for the Project4, Time, Interval, and InvalidTime classes. I utilized the lessons to apply them to the Project4 and TripCost classes. Once it was finished, I went back into the Project 4 class to create the GUI using the JavaFX Application, and then the user inputs would be passed and the results back through the Project 4 class. Project4 class creates the text fields, labels, buttons, pane, scenes, and other parts of the JavaFX to create the application. If it is an invalid input or 0 calculation/NaN, it must throw an exception or a message from InvalidTime. To debug Project 4, I looked at the samples, lessons, my old codes, and different perspectives of generics concepts online. I then modified the classes. I also adjusted the classes so that the constructor would pass the values from the text fields. After that, I had to confirm that the comparison would work with the buttons. Then, I checked back to see if the output was correct through the Project4 class.

Note: I did not include a Javadoc here as the Javadoc compiler wanted the file path of the JDK for the GUI/JavaFX (including imported classes) which made it complicated. Due to this, I did not include a Javadoc. The Javadoc comments should be similar to the coding comments in the Javadoc HTML.