Kirk Forthman

October 3, 2023

CMSC 215

Prof. Helms

**Project 4**

**Design Approach:**

The design of this project is largely defined in the instructions of the assignment. It involves the generation of a class to represent a time object that is comparable, using that time object class to define a generic class that uses a wild card to extend the time class that creates objects representing an interval of time in the same calendar day. Each of these classes can throw a number of exceptions. All checked exceptions will cause a self defined “Invalid Time” exception with a meaningful message to warn user of broken expectations while not crashing the program.

These classes and their instance objects and methods are used to operate the buttons in the GUI and take input via the text fields. When the “compare intervals” button is used, the text fields are used to create time objects that are used to create interval objects. The interval objects are then compared using a methods defined in the intervals class which utilize methods defined in the time class. These methods provide a specific answer to whether either of the intervals is a subinterval of the other, if the two intervals overlap or if the intervals are disjoint. When the “check time” button is used further methods defined in the interval class is used to determine which, if any interval contains the time requested.

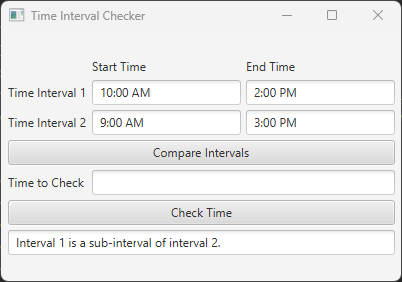
**UML Diagrams:**

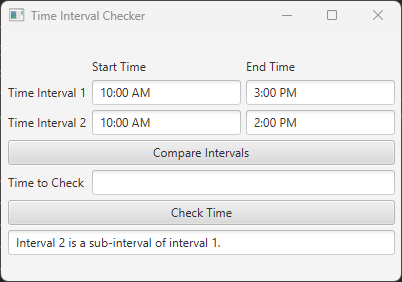
|  |  |  |
| --- | --- | --- |
| **Project4** |  | **Interval <E extends Time>** |
| +tfinterval1Start: TextField  +tfinterval1End: TextField  +tfinterval2Start: TextField  +tfinterval2End: TextField  +tfTimeToCheck: Textfield  +tfCheckTimeResult: TextField  +btnCompareIntervals: Button  +btnCheckTime: Button  +gridPane: GridPane |  | -start: <E>  -end: <E> |
| +compareIntervalText(Time: int1Start, Time: int1End, Time: int2Start, Time: int2End): String  +checkTimeText(Time: int1Start, Time: int1End, Time: int2Start, Time: int2End): String |  | +<<create>>Interval(E:start, E: end): void  +compareTo(): int  +within(E: object): boolean  +subinterval(Interval<Time>: testInterval): boolean  +overlaps(Interval<Time>: testInterval): boolean |
|  |  |  |
| **Time** |  | **InvalidTime (Exception)** |
| -hours: int  -minutes: int  -meridian: String |  | -errorMessage: String |
| +<<create>>Time(int: hours, int: minutes, String: meridian): void  +<<create>>Time(String: input): void  +getHours(): int  +getMinutes(): int  +getMeridian(): String  +compareTo(Time: o): int  +toString(): String |  | +<<create>>InvalidTime(): void |

**Test Cases:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Case#** | **Subject** | **Input** | **Expected Output** | **Actual Output** | **Pass/Fail** |
| 1 | 1 sub-int of 2 | Int1Start: 10:00 AM  Int1End: 2:00 PM  Int2Start: 9:00 AM  Int2End: 3:00 PM  CompareButton | Interval 1 is a subinterval of interval 2. | Interval 1 is a subinterval of interval 2. | Pass |
| 2 | 2 sub-int of 1 | Int1Start: 9:00 AM  Int1End: 3:00 PM  Int2Start: 10:00 AM  Int2End: 2:00 PM  CompareButton | Interval 2 is a subinterval of interval 1. | Interval 2 is a subinterval of interval 1. | Pass |
| 3 | overlap | Int1Start: 9:00 AM  Int1End: 2:00 PM  Int2Start: 1:00 PM  Int2End: 5:00 PM  CompareButton | The intervals overlap. | The intervals overlap. | Pass |
| 4 | Disjoint | Int1Start: 9:00 AM  Int1End: 10:00 AM  Int2Start: 11:00 AM  Int2End: 12:00 PM  CompareButton | The intervals are disjoint. | The intervals are disjoint. | Pass |
| 5 | Time in int 1 | Int1Start: 12:00 PM  Int1End: 3:00 PM  Int2Start: 2:00 PM  Int2End: 5:00 PM  Time: 1:15 PM  CheckTimeButton | Only interval 1 contains the time 1:15 PM. | Only interval 1 contains the time 1:15 PM. | Pass |
| 6 | Time in int 2 | Int1Start: 12:00 PM  Int1End: 3:00 PM  Int2Start: 2:00 PM  Int2End: 5:00 PM  Time: 4:15 PM  CheckTimeButton | Only interval 2 contains the time 4:15 PM. | Only interval 2 contains the time 4:15 PM. | Pass |
| 7 | Time in both ints | Int1Start: 12:00 PM  Int1End: 5:00 PM  Int2Start: 2:00 PM  Int2End: 5:00 PM  Time: 4:15 PM  CheckTimeButton | Both intervals contain the time 4:15 PM. | Both intervals contain the time 4:15 PM. | Pass |
| 8 | Time in neither int | Int1Start: 12:00 PM  Int1End: 3:00 PM  Int2Start: 2:00 PM  Int2End: 4:00 PM  Time: 4:15 PM  CheckTimeButton | Neither interval contains the time 4:15 PM. | Neither interval contains the time 4:15 PM. | Pass |
| 9 | Invalid Time Non-numeric entry | Int1Start: 12:0h PM  Int1End: 3:00 PM  Int2Start: 2:00 PM  Int2End: 4:00 PM  Time: 4:15 PM  CheckTimeButton | Invalid Time Format Entered (HH:MM AM/PM) | Invalid Time Format Entered (HH:MM AM/PM) | Pass |
| 10 | Invalid Time (semicolon instead of colon) | Int1Start: 12:00 PM  Int1End: 3;00 PM  Int2Start: 2:00 PM  Int2End: 4:00 PM  Time: 4:15 PM  CheckTimeButton | Invalid Time Format Entered (HH:MM AM/PM) | Invalid Time Format Entered (HH:MM AM/PM) | Pass |
| 11 | Invalid Time (too many minutes) | Int1Start: 12:00 PM  Int1End: 3:00 PM  Int2Start: 2:60 PM  Int2End: 4:00 PM  Time: 4:15 PM  CompareButton | Invalid Numeric Entry | Invalid Numeric Entry | Pass |
| 12 | Invalid Time (too many hours) | Int1Start: 12:00 PM  Int1End: 3:00 PM  Int2Start: 2:00 PM  Int2End: 14:00 PM  Time: 4:15 PM  CheckTimeButton | Invalid Numeric Entry | Invalid Numeric Entry | Pass |
| 13 | Invalid Time (too many digits) | Int1Start: 12:00 PM  Int1End: 3:00 PM  Int2Start: 2:00 PM  Int2End: 4:00 PM  Time: 4:150 PM  CheckTimeButton | Invalid Numeric Entry | Invalid Numeric Entry | Pass |
| 14 | Invalid Time (invalid meridian) | Int1Start: 12:00 PM  Int1End: 3:00 PM  Int2Start: 2:00 mM  Int2End: 4:00 PM  Time: 4:15 PM  CompareButton | Invalid Meridian | Invalid Meridian | Pass |

**Test Case Screen Captures:**

**1:**

**2:**

**3:A screenshot of a computer

Description automatically generated**

**4: A screenshot of a computer

Description automatically generated**

**5: A screenshot of a computer

Description automatically generated**

**6: A screenshot of a computer

Description automatically generated**

**7: A screenshot of a computer

Description automatically generated**

**8: A screenshot of a computer

Description automatically generated**

**9: A screenshot of a computer

Description automatically generated**

**10: A screenshot of a computer

Description automatically generated**

**11: A screenshot of a computer

Description automatically generated**

**12: A screenshot of a computer

Description automatically generated**

**13: A screenshot of a computer

Description automatically generated**

**14: A screenshot of a computer

Description automatically generated**

**Lessons Learned:**

In the implementation of this project I learned and practiced several lessons from this course. The most obvious lesson is the practice of generics and wildcards in the Interval class. In addition to the use of generics, I got to practice utilizing JavaFX for creating a GUI and event handling (including the use of lambda expressions), classes that extend classes, override methods and implement interfaces. I also used inner classes as well as anonymous inner classes, defined exceptions and handling them by throwing exceptions, using try/catch statements and passing the error message to the user.

This project allowed me to demonstrate many of the lessons learned in this course.