

## CMSC 215 Intermediate Programming

### Programming Project 4

The fourth programming project involves writing a program to test the relationships between time intervals. The program should contain four classes. The first class should be a generic class `Interval` defined for any type that implements the `Comparable` interface. Objects of this type have a start and end of the generic type parameter that defines the start and end of the interval. The class should be immutable, so it should have no setter methods. At a minimum it should contain the following public methods:

- A constructor that accepts the start and end of an interval and constructs an `Interval` object.
- A method `within` that is supplied an object of the generic type parameter and returns whether that object is inside the interval including the endpoints.
- A method `subinterval` that is passed an interval as a parameter and returns whether the interval parameter is a subinterval, completely within, the interval on which the method is invoked.
- A method `overlaps` that is passed an interval as a parameter and returns whether the interval parameter overlaps the interval on which the method is invoked.

The second class `Time` should contain two integer instance variables that represent the hours and minutes and one additional variable for the meridian, AM or PM. The class should implement the `Comparable` interface and should be immutable, so it should have no setter methods. At a minimum it should contain the following public methods:

- A constructor that accepts the hours and minutes as integers and the meridian as a string and constructs a `Time` object.
- A constructor that accepts a string representation of a time in the format HH:MM AM and constructs a `Time` object.
- A method `compareTo` that compares two times and returns what is required of all such methods needed to implement the `Comparable` interface.
- A method `toString` that returns the string representation of the time in the format time in the format HH:MM AM

When either constructor is called, several checks need to be made on the input. For both constructors, a check is needed to ensure that the hours and minutes are within range and that the meridian is AM or PM. For the constructor that accepts the string representation, additional checks are needed to ensure that the hours and minutes are numeric values. Should any check fail, an exception `InvalidTime` should be thrown that is provided the reason.

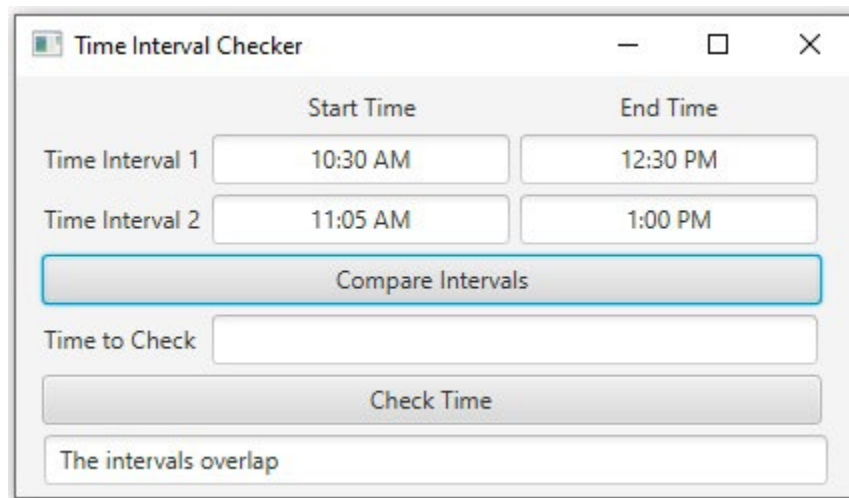
The third class is the exception class `InvalidTime` that implements a checked exception. It should have an instance variable of type `String` that saves the message and the following two methods;

- A constructor that accepts the message as a string and constructs an `InvalidTime` object
- A method `getMessage` that returns the message

The fourth class `Project4` should implement a GUI interface that contains two buttons. The first button *CompareIntervals* should compare the two intervals and output one of the following messages depending upon how the intervals compare:

- Interval 1 is a sub-interval of interval 2
- Interval 2 is a sub-interval of interval 1
- The intervals overlap
- The intervals are disjoint

Shown below is an example of the output when the *CompareIntervals* button is clicked:



The screenshot shows a window titled "Time Interval Checker". It has two columns: "Start Time" and "End Time". Under "Start Time", "Time Interval 1" is 10:30 AM and "Time Interval 2" is 11:05 AM. Under "End Time", "Time Interval 1" is 12:30 PM and "Time Interval 2" is 1:00 PM. A button labeled "Compare Intervals" is highlighted with a blue border. Below it is a text input field labeled "Time to Check". A button labeled "Check Time" is below the input field. At the bottom, a text box displays the message "The intervals overlap".

The second button *CheckTime* should check whether the time is within the intervals and output one of the following messages depending upon which intervals it is within:

- Both intervals contain the time HH:MM AM
- Only interval 1 contains the time HH:MM AM
- Only interval 2 contains the time HH:MM AM
- Neither interval contains the time HH:MM AM

Shown below is an example of the output when the *CheckTime* button is clicked:

### Documentation Requirements:

Follow the naming conventions previously provided in the course announcements. Please follow these requirements:

*Make sure your Java program is using the recommended style such as:*

- **Javadoc comment with your name as author, date, and brief purpose of the program**
- *Comments for variables and blocks of code to describe major functionality*
- *Meaningful variable names and prompts*
- Class names are written in upper CamelCase
- Constants are written in All Capitals
- Use proper spacing and empty lines to make your source code human readable

### Deliverables:

You are to submit two files.

1. The first is a .zip file that contains all the source code for the project. The .zip file should contain only source code and nothing else, which means only the .java files. If you elect to use a package the .java files should be in a folder whose name is the package name. Every outer class should be in a separate .java file with the same name as the class name. Each file should include a comment block at the top containing your name, the project name, the date, and a short description of the class contained in that file.
2. The second is a Word document (PDF or RTF is also acceptable) that contains the documentation for the project, which should include the following:
  - a. A UML class diagram that includes all classes you wrote. Do not include predefined classes.
  - b. A test plan that includes test cases that you have created indicating what aspects of the program each one is testing. Include the **results of your testing** with screen captures clearly showing the output for each test case.

- c. A short paragraph on lessons learned from the project.