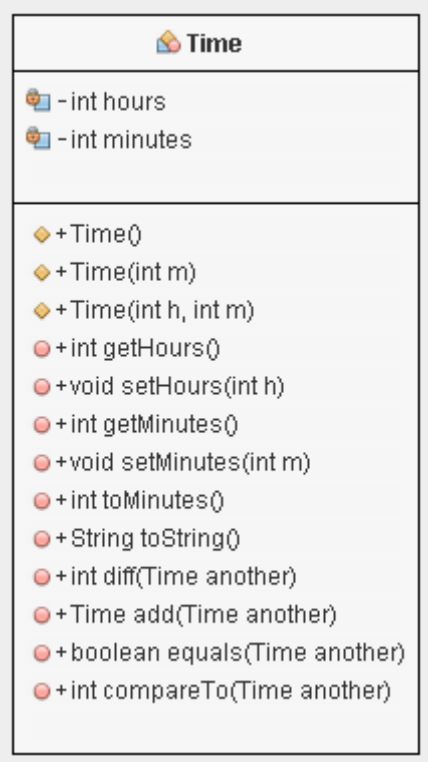
Practical IB Computer Science OOP Test

Name: _____ Date: 8/10/2020

Time Class

Write a Java Class to represent and handle times in 24 hour format. Your class should be cleanly written, encapsulated, override the `equals(Time another)` and `toString()` methods, use simple validation (e.g. no negative values), include a `toMinutes()` and `compareTo(Time another)` methods, and calculate the difference between two times. Refer to the UML class diagram below for additional guidance. Note that the `add(Time another)` method is optional (not assessed).

Use the file attached to the online homework `TimeTest.java` to expedite testing. You may want to comment and uncomment the appropriate lines of code in the main method of `TimeTest.java` as you implement each objective of this test.



Work through the test from the beginning. Your Time class should build and grow –do not start a new program for each point. During this test, you may use any resources that you have created, but you may **not** use Internet.

<<< Please Turn Over >>>

Practical IB Computer Science OOP Test

Objectives

1. Successfully implement an empty constructor
2. Successfully implement a constructor that accepts minutes, and another that accepts hours and minutes
3. Successfully implement the appropriate accessors (“getters”) and mutators (“setters”)
4. Successfully implement a **toMinutes()** method that turns a Time object into minutes
5. Successfully implement an **equals(Time another)** and a **compareTo(Time another**) methods that turns a Time object into minutes
6. Successfully implement a **toString()** method to format the output of a Time object with preceding zeroes if the hours or minutes are only one digit long
7. Add validation checks to the **setters** to handle possible overflow of hours – rollover of hours > 23, e.g. 24 hours = 0 hour; 30 hours = 6 hours; 64 hours = 16 hours
8. Add validation checks to the **setters** to handle minutes > 60 – rollover minutes, update hours if needed, e.g. 72 minutes = 1 hour 12 minutes
9. Successfully implement an **diff()** method to calculate the difference in minutes between two Time objects.
10. Write an **Event** class to keep track of an *eventTitle* (String), *eventLocation* (String), *eventDate* (Date) and *eventTime* (Time). Code at least a *constructor* and *toString()* methods.

Time Class Testing	OUTPUT	OBJECTIVES
No argument constructor	[00:00] = 00:00	1, 7
setHour(3); setMinute(360);	[09:00] = 09:00	3, 7, 8
Constructor with (15, 45)	[15:45] = 15:45	2
Minutes only constructor w/(-533)	[08:53] = 08:53	2, 7, 8
Constructor w/ (2, 97)	[03:37] = 03:37	2, 8
Constructor w/ (-80,-90), getters	[09:30] = 09:30	2, 4, 7, 8
toMinutes() method with 15:45	[945] = 945 m	4
toMinutes() method with 08:53	[533] = 533 m	4
equals method with 15:45 and 08:53	[false] = false	5
compareTo method w/ 08:53 and 15:45	[-412] = -412	5
diff method w/ 15:45 and 08:53	[1028] = 1028 minutes	9
diff method w/ 03:37 and 09:30	[353] = 353 minutes	9
Event [eventTitle=Xmas, eventLocation=home, eventDate=25/12/2020, eventTime=19:00]		10

[[[Objective #6 is used in all output, except when testing getters]]]