

Team Practice 3

In this problem, you will be defining class **Time** and simultaneously testing it using another class, **UseTime**, which will consist of method **main** only. Be sure to have your **Time** and **UseTime** classes defined in the same project. When running your program in Eclipse, pick **UseTime** as the main class.

Define a class called **Time** to represent 24-hour clock time values represented with the hour and minute components. The class should define the following private instance variables:

1. *hour* (integer), and
2. *minute* (integer).

Define the following public instance methods for the class

- a) Mutator method *setTime()*: accepts 2 input parameters representing an hour and a minute values. The method should verify that the passed values are valid, i.e. the hour is between 0 and 23, and the minute is between 0 and 59. If the values are valid, they should be assigned to the corresponding instance variables of the calling object and the method should return *true*. Otherwise, the method should print a message about the invalid parameter and return *false*, leaving the instance variables unchanged.
- b) Accessor method *getHour()*: returns the value of the hour instance variable of the calling object.
- c) Accessor method *getMinute()*: returns the value of the minute instance variable of the calling object.
- d) *toMinutes()*: returns an int, representing the time value expressed in minutes.
- e) *getAMPMTIME()*: returns a String, representing the time value stored in a calling object in the AM/PM format. For example, if the calling object's hour is 14 and minute is 45, the method should return "2:45 PM". If the calling object's hour is 10 and minute is 14, the method should return "10:14 AM".
- f) *timeAfter()*: accepts 1 input parameter, let's call it *min*, designating number of minutes. The method should create and return a new object of type **Time**, which represents the time which is *min* minutes after the calling object.

As you are developing class **Time**, create a separate class called **UseTime** that will be used for testing the functionality of the **Time** class. The **UseTime** class will consist of a single method - **main()**. The main method must

1. Prompt the user to enter an hour and a minute values.
2. Create a new object of class **Time** and call the *setTime()* method to set its instance variables to the hour and minute values obtained from the user. If the value returned by the *setTime()* method is false, display an appropriate message and continue asking the user (step 1) to enter time strings, until *setTime()* returns true.
3. Call methods *getHour()* and *getMinute()* and print their return values, as shown in the sample interaction below.
4. Call method *toMinutes()* and *getAMPMTIME()* and print the return values, as shown in the first underlined line of the sample interaction below.
5. Ask a user to enter a number of minutes, and call methods *timeAfter()* and *getAMPMTIME()* to obtain new values of time as shown in the last line of the interaction below.

Here is a sample interaction. The numbers on the left are not part of the output – they correspond to the item numbers above. User input appears in boldface:

```
1,2: Testing the Time class definition:
    Please enter hour and minute value: 17 70
    Invalid value of minute: 70
    Please enter hour and minute value: 71 20
    Invalid value of hour: 71
    Please enter hour and minute value: 17 20
3: Hour is 17, minute is 20.
4: That's 1040 minutes, or 5:20 PM.
5: How many minutes would you like to add to that time? 75
    That's 6:35 PM.
```