RIT | Golisano College of Computing and Information Sciences
School of Information

Name: _____

ISTE-120
Lab 06: Expressions & Scanner

Exercise 1 - Adding a tester and a method (3 points)

The exercise must be completed during the lab period.

Write a program that reads in two times in 24-hour (or military) format (e.g. 0900 representing 9:00 am or 1730 representing 5:30 pm) and prints the amount of time between the two times as number of hours and minutes. The times can be entered in either order and the result would be the same. Use the Scanner class for input. Assume both times are for the same day of the week.

Implement the class `TimeInterval` whose constructor takes in the two times as ints. The class should have three methods: `getHours`, `getMinutes` and `getDecimalTime`.

The constructor must test for illegal times (e.g., 2730 or -730) and print an error message. In the event of an error, terminate the program (`System.exit(0)`).

Create a class `TestTimeInterval` that uses the `Scanner` class to get the input (that is, the two times) and produces the following results:

```
Please enter the first time: 0400
Please enter the second time: 1530

Elapsed time in hrs/min:        11 hours 30 minutes
Elapsed Time in decimal:        11.5 hours
```

The reverse entry gives the same result:

```
Please enter the first time: 1530
Please enter the second time: 0400

Elapsed time in hrs/min:        11 hours 30 minutes
Elapsed Time in decimal:        11.5 hours
```

HINT: Use integer division to divide the entered time into hours and minutes.

©Peter Lutz, David Patric 2020

BUT - don't be fooled into believing that's all there is to it. Suppose the times are 1230 and 1309. These give 12 hours and 30 minutes and 13 hours and 9 minutes, respectively. But, if we try to subtract these directly, we get 1 hour and 21 minutes - but we know the difference is actually 39 minutes.

Times are 'sort' of numbers in base 60. But Java does arithmetic in base 10. So, we have to convert times to something in base 10 to use base 10 subtraction. To do this, convert BOTH times into the number of MINUTES from midnight to the given time, using the formula:

hours * 60 + minutes

So, 1230 becomes 12 * 60 + 30 = 750 minutes and 1309 becomes 13 * 60 + 9 = 789. The difference is 39 minutes, which is correct.

Also, 0400 becomes 4 * 60 + 0 = 240 minutes and 1530 becomes 15 * 60 + 30 = 930 minutes. The difference is 690 minutes. Now … 690 / 60 (int division) is 11 and 690 % 60 = 30 for 11 hours and 30 minutes which is also correct.

So the CORRECT approach is:

- Take the time, as given, and separate it into hours and minutes.
- Use the hours * 60 + minutes formula to convert it to just minutes (since midnight)
- Subtract the two times to get D (use Math.abs here to handle a negative outcome)
- Use the quotient of D and 60 to convert minutes back to full hours and use the remainder of D and 60 to convert minutes back to remaining minutes.

Your solution should work with 0400 and 1530 (as above) as well as with 1230 and 1309.

Signature: _____ Date: _____
Have your instructor or TA sign here when Exercise 1 works correctly.

©Peter Lutz, David Patric 2020

Exercise 2 – Implementing a class and tester (3 points)

The exercise must be completed during the lab period.

Develop a program that transforms the numbers 1, 2, 3, 4, …, 12 into the corresponding month names: January, February, March, April, …, December.

Implement the class `Month` whose constructor parameter is a month number (1…12). Create a method `getName()` that returns the month name (a `String`).

The solution must use a long string that contains all of the month names:

```
January February March    April      … etc.
```

Take care to **add spaces** between the month names so that each month name, along with added spaces is the same length (9 characters - which fits the longest month name `September`). Use the `substring()` method of the `String` class to extract the correct month (see the Java API for details).

Create a class `MonthTester` that uses `Scanner` to get the month number, create a `Month` object and produce an output as shown below:

```
Enter the month number (1 - 12): 4
The month is: April
```

or:

```
Enter the month number (1 - 12): 9
The month is: September
```

What happens if you enter a 0 or a 13 for the month number?

_____

Signature: _____ Date: _____

Have your instructor or TA sign here when Exercise 2 works correctly.

©Peter Lutz, David Patric 2020

Exercise 3 – Calculations (4 points)

If you do not complete this exercise during the lab period, you need to complete the work outside of the lab period and bring the completed work to the lab next week.

Write a class that will compute the date of Easter Sunday. Easter Sunday is the first Sunday after the first full moon of spring. To find Easter's date, use an algorithm developed by the mathematician Carl Friedrich Gauss in 1800.

1. Let **y** be the year; e.g. **1800**, **1975** or **2014**.
2. Divide **y** by **19** and call the remainder **a**. Ignore the quotient.
3. Divide **y** by **100** to get a quotient **b** and remainder **c**.
4. Divide **b** by **4** to get a quotient **d** and remainder **e**
5. Divide **(8 * b + 13)** by **25** to get quotient **g**. Ignore the remainder.
6. Divide **(19 * a + b – d – g + 15)** by **30** to get remainder **h**. Ignore the quotient.
7. Divide **c** by **4** to get the quotient **j** and remainder **k**.
8. Divide **(a + 11 * h)** by **319** to get quotient **m**. Ignore the remainder.
9. Divide **(2 * e + 2 * j – k – h + m + 32)** by **7** to get remainder **r**. Ignore the quotient.
10. Divide **(h – m + r + 90)** by **25** to get quotient **n**. Ignore the remainder.
11. Divide **(h – m + r + n + 19)** by **32** to get remainder **p**. Ignore the quotient.

The result is that Easter falls on day **p** of month **n** in the given year. For example, if **y** is **2001**:

```
a = 6                 g = 6                 m = 0
b = 20                h = 18                r = 6
c = 1                 j = 0,                n = 4
d = 5, e = 0          k = 1                 p = 15
```

Therefore, in **2001**, Easter Sunday falls on April 15 **(n = 4** and **p = 15**).

If you get the wrong answer, put a breakpoint on the last statement of your algorithm and check out the values of all the variables (a - p) and see where they go wrong. Then fix your code and try again.

Write a class `Easter` with methods `getEasterSundayMonth()` and `getEasterSundayDay()`. Write a class `EasterTester` that tests the `Easter` class, uses the `Scanner` to get the year and produces the following output:

```
Enter the year: 2001
Month: 4
Day: 15
```

Signature: _____ Date: _____

Have your instructor or TA sign here when Exercise 3 works correctly.

©Peter Lutz, David Patric 2020