# COMP 1600 Fall 2025

# Project 2: Going through a Phase

## Due by: Friday, October 3, 2025 at 11:59 p.m.

Photo credit: **John5199**

Werewolves walk the earth! Above ground, your friends and family are being devoured, but you are safely hidden in an underground bunker. Your trusty digital wristwatch keeps track of the date, but your food is running out. When will it be safe to go up to the surface? Well, lycanthropes only take their dreaded wolf-shape near a full moon. And if it's Sunday, then there's bound to be holy water available to throw on them as well.

What to do? Of course! You can use computer science to save the day! Your mission is to write a Java program that will accomplish the following tasks.

1. Read in a date in normal format: December 7, 1941
2. Determine the Julian day for the date and output it.
3. Determine the day of the week that the date occurred on and output it.
4. Determine the phase of the moon on the date and output that.

## Specification

### Input

Your program must read in a month name, a day, and a year. The month name will be a `String` while the day and year will be `int` values.

Normal use of `Scanner` will serve you well, except that there is a comma after the day of the month. Consequently, you should read in the day-comma combination as a `String`. You should use the `substring()` method and the `Integer` wrapper class to extract the `int` data from this `String` made up of a number followed by a comma.

**Hint:** You need to use `substring()` to get rid of the last character, the comma.

Next, to do the math in the next section, you will need to know the number of the month (1-12), not just the name. To discover the month number, you will need to examine the `String` holding the month's name. Recall that the `equals()` method can be used to compare two `String`s. It will return `true` if the two strings are identical. Consider the following example code.

```
String text = "walnuts";
boolean same = text.equals("walnuts");
```

The variable `same` should contain `true` after this code has executed, because `text` contains `"walnuts"`. I have given this example just to remind you how `equals()` works. If is very likely that you will use it in an `if` statement, unlike this example. Note: `equals()` is case sensitive. The strings `"january"` and `"January"` are different. In your program, you may assume that the user enters the month with correct capitalization. Do **not** use `==` to compare two `String` values.

Alternatively, in Java 7 and higher, you can use a `switch` statement to determine the number of the month.

## Julian Day

For some situations (like this one), it is useful to know what day a date falls on based on some arbitrary starting point. One such system, often used by astronomers, is the **Julian day**. The Julian calendar starts on what would be November 24, 4714 BC in the Gregorian calendar that we generally use. That is Julian day 0. The Julian day for any later date is however many days after that point that it falls. For example, July 4, 1776 has a Julian day of 2,369,916.

To compute the Julian day, use the following formulas, where *month* is the number of the month (1-12), *year* is the year, and *day* is the day of the month.

$a = (14 - month) / 12$
$y = year + 4800 - a$
$m = month + 12a - 3$
$Julian = day + (153m + 2)/5 + 365y + y/4 - y/100 + y/400 - 32045$

Note that all divisions shown above are integer divisions, so these formulas are very easy to implement in Java with `int` values.

## Day of the Week

Once you know the Julian day of a date, computing the day of the week is easy using the following formula.

$d = (Julian + 1) \bmod 7$

The result *d* will be a number 0-6. You can determine the day of the week from it with the following table.

| Number | Day |
|---|---|
| 0 | Sunday |
| 1 | Monday |
| 2 | Tuesday |
| 3 | Wednesday |
| 4 | Thursday |
| 5 | Friday |
| 6 | Saturday |

## Phase of the Moon

It takes the moon roughly 29.530588853 days to go through a complete lunar cycle from new moon to full moon and back to new moon. Astronomers further break down that cycle into eight phases: new moon, waxing crescent, first quarter, waxing gibbous, full moon, waning gibbous, last quarter, and waning crescent.

To compute the current phase of the moon, we want to find the difference in Julian day between the current date and some date we know had a new moon. Then, we can take the modulus 29.530588853 which will give us a number between 0 and 29.530588853 which says how many days into the lunar cycle we are. Finally, if we divide that day number by 29.530588853, multiply by 8, and round to the nearest integer, we should get an integer between 0 and 8. These values map to the lunar phases described above according to the following table.

| Value | Lunar Phase | Value | Lunar Phase |
|---|---|---|---|
| **0, 8** | New Moon | 4 | Full Moon |
| **1** | Waxing Crescent | 5 | Waning Gibbous |
| **2** | First Quarter | 6 | Last Quarter |
| **3** | Waxing Gibbous | 7 | Waning Crescent |

Because of rounding, both 0 and 8 correspond to the New Moon phase.

In order to do this calculation, you have to know a date when there was a new moon. As it turns out, the moon was new on January 1, 1900. Thus, you should use the Julian day calculation formula from above to determine the Julian day of January 1, 1900, subtract that result from the Julian day of the day in question, find its modulus 29.530588853, divide by 29.530588853, multiply by 8, and round to the nearest integer. From that number, you should be able to print out the lunar phase.

Note that you may get a negative number if the date comes before January 1, 1900. If the final phase value is a negative number, add 8 to it to get the correct result.

## Sample Execution

Below is sample output from the execution of the program. The user is prompted for a date in the form month name, day, comma, year.

After these values are read in, the date is repeated with its Julian day. Then, its day of the week is output. Finally, the corresponding phase of the moon is output.

Note: Your output must match **exactly** to the character to get full credit. `Green` is used below to indicate user input.

```
Enter the date (e.g. July 4, 1776): July 4, 1776
July 4, 1776 has a Julian day of 2369916.
Its day of the week is Thursday.
Its phase of the moon is Waning Gibbous.
```

Another example follows.

```
Enter the date (e.g. July 4, 1776): February 20, 1967
February 20, 1967 has a Julian day of 2439542.
Its day of the week is Monday.
Its phase of the moon is Waxing Gibbous.
```

Here's one more example.

```
Enter the date (e.g. July 4, 1776): October 3, 2014
October 3, 2014 has a Julian day of 2456934.
Its day of the week is Friday.
Its phase of the moon is First Quarter.
```

## Testing

Test heavily. Test today's date, your own birthday, and any date you care to look up on **Moon Phase for Any Date**. For example, the attack on Pearl Harbor was Sunday, December 7, 1941. It has a Julian day of 2430336, and the moon was Waning Gibbous.

## Turn In

Your IntelliJ project should be called `Project2`, but the class you create inside should be called `MoonPhase`. Upload the `MoonPhase.java` from the `Project2\src` folder wherever you created your project to **Brightspace**.

All work must be submitted before Friday, October 3, 2025 at 11:59 p.m. unless you are going to use a grace day.

All work must be done individually. You may discuss general concepts with your classmates, but it is never acceptable for you to look at someone else's code. Please refer to the course policies if you have any questions about academic integrity. If you have trouble with the assignment, I am always available for assistance.

## Grading

Your grade will be determined by the following categories:

| Category | Weight |
|---|---|
| Correctly reading and interpreting input | 25% |
| Computing the Julian day and outputting it properly | 25% |
| Computing the day of the week and outputting it properly | 15% |
| Determining the phase of the moon and outputting it properly | 25% |
| Style and comments | 10% |

**Under no circumstances should any student look at the code written by another student. Tools will be used to detect code similarity automatically.**

**Code that does not compile will automatically score zero points.**