

## Model 1 Attributes and Methods

Here is a new and improved version of the `enum` from ???. Read and discuss the following source code as a team.

```
1 public enum Month {
2
3     JAN("January", 31),
4     FEB("February", 28),
5     MAR("March", 31),
6     APR("April", 30),
7     MAY("May", 31),
8     JUN("June", 30),
9     JUL("July", 31),
10    AUG("August", 31),
11    SEP("September", 30),
12    OCT("October", 31),
13    NOV("November", 30),
14    DEC("December", 31);
15
16    private final String name;
17    private final int days;
18
19    private Month(String name, int days) {
20        this.name = name;
21        this.days = days;
22    }
23
24    public int length() {
25        return days;
26    }
27
28    public int number() {
29        return ordinal() + 1;
30    }
31
32    public static Month parseMonth(String name) {
33        String abbr = name.substring(0, 3);
34        return valueOf(abbr.toUpperCase());
35    }
36
37    public String toString() {
38        return name;
39    }
40
41 }
```

## Questions (20 min)

Start time:

1. What are the attributes of a `Month` object?
2. Open the provided *Month.java* file. Try changing the constructor to `public`. What compiler error results?
3. Based on what you observed in ??, why do you think an `enum` constructor must be declared `private`?
4. On which lines is the `Month` constructor called in Model 1?
5. Other than `substring` and `toUpperCase`, what methods are called in Model 1 that are not explicitly defined in *Month.java*?
6. The `number` method returns the numeric value of the month (i.e., 1 for January or 12 for December). Explain how the implementation works.
7. The `parseMonth` method returns the `Month` that corresponds to the provided name. Explain how the implementation works.

8. Open the provided *MonthHelp.java* file, and discuss the code as a team. Write additional code that displays the full name and number of days in the month input by the user. For example, if the user inputs `Sept.`, output the message `September has 30 days`.

9. Implement a new method named `parseMonth(int number)` that returns the month for the given integer. For example, `parseMonth(1)` would return `JAN`, `parseMonth(2)` would return `FEB`, and so forth. (*Hint: Use values.*)

10. Extend your code from #8 to use both versions of `parseMonth`. If the user inputs a month name or 3-letter abbreviation, call the string version. If the user inputs a month number, call the integer version. (*Hint: Use `line.length()` and `Integer.parseInt(line)`.*)