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.

```
public enum Month {

    JAN("January", 31),
    FEB("February", 28),
    MAR("March", 31),
    APR("April", 30),
    MAY("May", 31),
    JUN("June", 30),
    JUL("July", 31),
    AUG("August", 31),
    SEP("September", 30),
    OCT("October", 31),
    NOV("November", 30),
    DEC("December", 31);

    private final String name;
    private final int days;

    private Month(String name, int days) {
        this.name = name;
        this.days = days;
    }

    public int length() {
        return days;
    }

    public int number() {
        return ordinal() + 1;
    }

    public static Month parseMonth(String name) {
        String abbr = name.substring(0, 3);
        return valueOf(abbr.toUpperCase());
    }

    public String toString() {
        return name;
    }

}
```

Questions (20 min) **Start time:**

1. What are the attributes of a `Month` object?

The `name` of a month, and the number of `days` in a month.

2. Open the provided *Month.java* file. Try changing the constructor to `public`. What compiler error results?

Illegal modifier for the enum constructor; only private is permitted.

3. Based on what you observed in **??**, why do you think an `enum` constructor must be declared `private`?

Enum types may not be instantiated in other classes using the `new` operator. However, they may be instantiated within the enum definition itself.

4. On which lines is the `Month` constructor called in Model 1?

Lines 3–14, where the `Month` constants are defined.

5. Other than `substring` and `toUpperCase`, what methods are called in Model 1 that are not explicitly defined in *Month.java*?

The `ordinal` method (on Line 29) and the `valueOf` method (on Line 34).

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.

The `ordinal` method returns the month's position in the enum declaration, which is a number from 0 to 11. Adding one to this number yields the desired result.

7. The `parseMonth` method returns the `Month` that corresponds to the provided name. Explain how the implementation works.

It gets the first three letters of the given name and converts them to uppercase. Then it uses the `valueOf` method to get the corresponding `Month` constant.

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`.

```
Month m = Month.parseMonth(line);
System.out.printf("%s has %d days\n", m, m.length());
```

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`.)

```
public static Month parseMonth(int number) {
    return values()[number - 1];
}
```

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)`.)

```
Month m;
if (line.length() > 2) {
    m = Month.parseMonth(line);
} else {
    int number = Integer.parseInt(line);
    m = Month.parseMonth(number);
}
System.out.printf("%s has %d days\n", m, m.length());
```