Enum Types

An `enum` is a special data type that defines a fixed set of constants. Enums are a good choice when you can *enumerate* all possible values at compile time.

Manager:	Recorder:
Presenter:	Reflector:

Content Learning Objectives

After completing this activity, students should be able to:

- Explain and apply the methods provided by an enum type.
- Summarize the main differences between classes and enums.
- Implement an enum that includes attributes and methods.

Process Skill Goals

During the activity, students should make progress toward:

- Discussing results while running code interactively. (Oral Communication)

Model 1 Months of the Year

Open *JShell* on your computer. Type (or copy and paste) the following enum definition:

```
public enum Month {
    JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC;
}
```

Then type each line of code below in *JShell*, ***one at a time***, and record the results. You only need to record the output to the right of the "==>" symbol. For example, if *JShell* outputs `$8 ==> true`, then just write `true`. If an error occurs, summarize the error message.

Java code	Shell output

`Month m = null;`
`m = JUN;`
`m = Month.JUN;`
`m.toString()`	

`Month spring = Month.MAR;`
`Month summer = Month.JUN;`
`m == spring`
`m == summer`
`Month.JUL = summer;`	

`m.ordinal()`
`spring.ordinal()`
`Month.OCT.ordinal()`
`m.compareTo(spring)`
`m.compareTo(Month.OCT)`	

`m = Month.valueOf("Mar");`
`m = Month.valueOf("MAR");`
`m == spring`
`m = Month.valueOf(5);`
`m = new Month("HEY");`	

`Month[] all = Month.values();`
`all[0]`
`all[11]`
`all[12]`	

Questions (25 min) Start time:

1. Consider the variables `JAN`, `FEB`, `MAR`, etc. Based on how they were used:

a) Are they `public`? b) Are they `static`? c) Are they `final`?

2. Is the variable "`m`" a primitive type or a reference type? Justify your answer. (If primitive, what is its value? If not, what does it reference?)

3. What ability do classes have that enums do not? (*Hint:* Review the error message for `"HEY"`.)

4. Based on your previous answers, explain why it's okay to compare enum variables using the == operator (as opposed to calling the `equals` method).

5. What does the `ordinal` method return? Explain the range of possible values.

6. What does the `compareTo` method return? Explain how to interpret the results.

7. What does the `valueOf` method return?

8. What does the `values` method return?

9. Which of the aforementioned methods are `static`? Explain how you can tell.

10. The following code snippet prompts the user to input their birth month:

```
Scanner in = new Scanner(System.in);
System.out.print("What month were you born? ");
String line = in.nextLine();
```

a) Declare a variable named `birth` and initialize it to the `Month` object that corresponds to the user input. (*Hint:* Use `valueOf`.)

b) Output a message that tells the user the number of their birth month. For example, if the user inputs `MAY`, output the message `You were born in month #5`. (*Hint:* Use `ordinal`.)

c) Write an enhanced `for` loop that outputs each of the `Month` names that come before `birth`. (*Hint:* Use `values` and `compareTo`.)

Model 2 Attributes and Methods

Here is a new and improved version of the `enum` from Model 1. Read and discuss the following source code as a team. Notice in particular how the constants (`JAN`, `FEB`, `MAR`, etc.) are declared. As before, the constants are separated by commas and end with a semicolon.

```java
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:

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

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

13. Based on what you observed in Model 1, why do you think an `enum` constructor must be declared `private`?

14. On which lines is the `Month` constructor called in Model 2?

15. Other than `substring` and `toUpperCase`, what methods are called in Model 2 that are not explicitly defined in *Month.java*?

16. The `number` method returns the numeric value of the month (i.e., `1` for January or `12` for December). Explain how the implementation works.

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

18. 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`.

19. 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`.)

20. Extend your code from #18 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)`.)