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)

Facilitation Notes

Model 1: Remind students how to use JShell, especially tab completion and the up/down arrow keys. Encourage students to predict and discuss the results as a team while they run each line of code in JShell.

Model 2: Note that the provided files contain additional code than is in the model. Tell students to focus on the code in the activity, but they can use the source files to try other examples.

Key questions: #4, #10, #20

Source files: *Month.java*, *MonthHelp.java*

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;`	null
`m = JUN;`	cannot find symbol: variable JUN
`m = Month.JUN;`	JUN
`m.toString()`	"JUN"
`Month spring = Month.MAR;`	MAR
`Month summer = Month.JUN;`	JUN
`m == spring`	false
`m == summer`	true
`Month.JUL = summer;`	cannot assign a value to final variable JUL
`m.ordinal()`	5
`spring.ordinal()`	2
`Month.OCT.ordinal()`	9
`m.compareTo(spring)`	3
`m.compareTo(Month.OCT)`	-4
`m = Month.valueOf("Mar");`	IllegalArgumentException: No enum constant
`m = Month.valueOf("MAR");`	MAR
`m == spring`	true
`m = Month.valueOf(5);`	no suitable method found for valueOf(int)
`m = new Month("HEY");`	enum types may not be instantiated
`Month[] all = Month.values();`	Month[12] { JAN, FEB, MAR, ... }
`all[0]`	JAN
`all[11]`	DEC
`all[12]`	ArrayIndexOutOfBoundsException

Questions (25 min) Start time:

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

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

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?)

A reference type, because it can be `null`. It references one of the constants in `Month`.

3. What ability do classes have that enums do not have? (*Hint:* Review the error messages.)

Classes can be instantiated using the `new` operator.

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).

If two enums variables are equal, then they will be referencing the same constant. Only one of each constant will exist; there will not be multiple equivalent `Month` objects.

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

The "order" of the constant (i.e., its position in the enum definition). For `Month`, the possible values range from 0 to 11.

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

The difference between the ordinal values. Negative results mean "this" value comes before the "other" value; positive results mean "this" value comes after the "other" value.

7. What does the `valueOf` method return?

The `Month` object with the specified name, or it throws `IllegalArgumentException` if the name is not found.

8. What does the `values` method return?

An array of references to all constants in the enum. This method is useful for iterating the constants.

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

Enums have two static methods: `valueOf` and `values`. These methods are invoked on the enum itself, rather than an instance.

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`.)

```
Month birth = Month.valueOf(line);
```

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`.)

```
System.out.printf("You were born in month #%d\n", birth.ordinal() + 1);
```

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

```
for (Month m : Month.values()) {
    if (m.compareTo(birth) < 0) {
        System.out.println(m);
    }
}
```

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.

```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?

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

12. 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.

13. Based on what you observed in Model 1, 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.

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

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

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

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

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.

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.

17. 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.

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`.

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

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`.)

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

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)`.)

```
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());
```