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);
    }
}
```