Static data types

Enums

Enum types represent a data type with a finite or concrete set of possible values. For instance, the seasons of the year, of the days of the week are good examples of enums.

1. Basic enum management

If we want to declare an *enum* type, we use the word `enum` followed by a name. This will be the name that we will use to refer to this data type when we declare variables. Then, between curly braces, we specify the values that this enum can have.

For instance, this way we can declare an enum for the days of the week:

```
enum DayOfWeek { MONDAY, TUESDAY, WEDNESDAY, THURSDAY,
    FRIDAY, SATURDAY, SUNDAY};
```

This *enum* should be declared out of the *class*, as an external data type. Then, we can declare variables of this enum in our program, this way:

```
DayOfWeek aDay = DayOfWeek.FRIDAY;
```

Every value inside an *enum* as a numeric index associated, starting at 0. This way, we can either work with the enum value or with its inner index, if we want to, using `ordinal()` instruction.

```
DayOfWeek aDay = DayOfWeek.FRIDAY;
System.out.println(aDay);           // Prints FRIDAY
System.out.println(aDay.ordinal()); // Prints 4
```

If we want to read an enum value from the keyboard, this value can be either the enum value (for instance, *FRIDAY*), or the ordinal number for this value (4, if we want to refer to FRIDAY). We can use `valueOf` or `value` instructions, respectively, to match the value typed by the user with the value stored in the enum:

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter a day of the week:");
DayOfWeek aDay1 = DayOfWeek.valueOf(sc.nextLine());  // If user types FRIDAY
DayOfWeek aDay2 = DayOfWeek.values()[sc.nextInt()];  // If user types 4
```

> **Exercise 1**:
>
> Create a program called **MonthEnum.java** that defines an enum to store the months of the year. Then, ask the user to type a month and tell him/her the number of days of this month (we assume that February has 28 days).