Enum Types

An *enum type* is a special data type that enables for a variable to be a set of predefined constants. The variable must be equal to one of the values that have been predefined for it. Common examples include compass directions (values of NORTH, SOUTH, EAST, and WEST) and the days of the week.

Because they are constants, the names of an enum type's fields are in uppercase letters.

In the Java programming language, you define an enum type by using the `enum` keyword. For example, you would specify a days-of-the-week enum type as:

```
public enum Day {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
    THURSDAY, FRIDAY, SATURDAY
}
```

You should use enum types any time you need to represent a fixed set of constants. That includes natural enum types such as the planets in our solar system and data sets where you know all possible values at compile time—for example, the choices on a menu, command line flags, and so on.

Here is some code that shows you how to use the `Day` enum defined above:

```
public class EnumTest {
    Day day;

    public EnumTest(Day day) {
        this.day = day;
    }

    public void tellItLikeItIs() {
        switch (day) {
            case MONDAY:
                System.out.println("Mondays are bad.");
                break;

            case FRIDAY:
                System.out.println("Fridays are better.");
                break;

            case SATURDAY: case SUNDAY:
                System.out.println("Weekends are best.");
                break;

            default:
                System.out.println("Midweek days are so-so.");
                break;
        }
    }

    public static void main(String[] args) {
        EnumTest firstDay = new EnumTest(Day.MONDAY);
        firstDay.tellItLikeItIs();
        EnumTest thirdDay = new EnumTest(Day.WEDNESDAY);
        thirdDay.tellItLikeItIs();
        EnumTest fifthDay = new EnumTest(Day.FRIDAY);
        fifthDay.tellItLikeItIs();
        EnumTest sixthDay = new EnumTest(Day.SATURDAY);
        sixthDay.tellItLikeItIs();
        EnumTest seventhDay = new EnumTest(Day.SUNDAY);
```

```
            seventhDay.tellItLikeItIs();
        }
    }
```

The output is:

```
    Mondays are bad.
    Midweek days are so-so.
    Fridays are better.
    Weekends are best.
    Weekends are best.
```

Java programming language enum types are much more powerful than their counterparts in other languages. The `enum` declaration defines a *class* (called an *enum type*). The enum class body can include methods and other fields. The compiler automatically adds some special methods when it creates an enum. For example, they have a static `values` method that returns an array containing all of the values of the enum in the order they are declared. This method is commonly used in combination with the for-each construct to iterate over the values of an enum type. For example, this code from the `Planet` class example below iterates over all the planets in the solar system.

```
    for (Planet p : Planet.values()) {
        System.out.printf("Your weight on %s is %f%n",
                          p, p.surfaceWeight(mass));
    }
```

Note: *All* enums implicitly extend `java.lang.Enum`. Since Java does not support multiple inheritance, an enum cannot extend anything else.

In the following example, `Planet` is an enum type that represents the planets in the solar system. They are defined with constant mass and radius properties.

Each enum constant is declared with values for the mass and radius parameters. These values are passed to the constructor when the constant is created. Java requires that the constants be defined first, prior to any fields or methods. Also, when there are fields and methods, the list of enum constants must end with a semicolon.

Note: The constructor for an enum type must be package-private or private access. It automatically creates the constants that are defined at the beginning of the enum body. You cannot invoke an enum constructor yourself.

In addition to its properties and constructor, `Planet` has methods that allow you to retrieve the surface gravity and weight of an object on each planet. Here is a sample program that takes your weight on earth (in any unit) and calculates and prints your weight on all of the planets (in the same unit):

```
public enum Planet {
    MERCURY (3.303e+23, 2.4397e6),
    VENUS   (4.869e+24, 6.0518e6),
    EARTH   (5.976e+24, 6.37814e6),
    MARS    (6.421e+23, 3.3972e6),
    JUPITER (1.9e+27,   7.1492e7),
    SATURN  (5.688e+26, 6.0268e7),
    URANUS  (8.686e+25, 2.5559e7),
    NEPTUNE (1.024e+26, 2.4746e7);

    private final double mass;   // in kilograms
    private final double radius; // in meters
    Planet(double mass, double radius) {
        this.mass = mass;
        this.radius = radius;
    }
    private double mass() { return mass; }
```

```
        private double radius() { return radius; }

        // universal gravitational constant  (m3 kg-1 s-2)
        public static final double G = 6.67300E-11;

        double surfaceGravity() {
            return G * mass / (radius * radius);
        }
        double surfaceWeight(double otherMass) {
            return otherMass * surfaceGravity();
        }
        public static void main(String[] args) {
            if (args.length != 1) {
                System.err.println("Usage: java Planet <earth_weight>");
                System.exit(-1);
            }
            double earthWeight = Double.parseDouble(args[0]);
            double mass = earthWeight/EARTH.surfaceGravity();
            for (Planet p : Planet.values())
               System.out.printf("Your weight on %s is %f%n",
                                 p, p.surfaceWeight(mass));
        }
    }
```

If you run `Planet.class` from the command line with an argument of 175, you get this output:

```
$ java Planet 175
Your weight on MERCURY is 66.107583
Your weight on VENUS is 158.374842
Your weight on EARTH is 175.000000
Your weight on MARS is 66.279007
Your weight on JUPITER is 442.847567
Your weight on SATURN is 186.552719
Your weight on URANUS is 158.397260
Your weight on NEPTUNE is 199.207413
```

« Previous • Trail • Next »