COURSEWARE

Enums

Contents

Overview

Enumerated types (enums) are used as a clean way of storing data that has a finite number of valid values.
Enums should be defined in their own file to allow for re-use across the entire application.

Tutorial

Creating an Enum

We can create an enum class like the following.

```
public enum Planet {
    MERCURY, VENUS, EARTH, MARS, JUPITER, SATURN, URANUS, NEPTUNE
}
```

We would store the planets in our solar system in an enum as there are only eight possible values that are valid.
Note that each value is in all capital letters, this is because the values are all constants so standard naming convention in Java suggests that they should be all capitals.

Enum Fields

We can add to our enum to provide more information on each value, by adding fields and a constructor.

```
public enum Planet {
    MERCURY(3.03e+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;
    private final double radius;

    Planet(double mass, double radius) {
        this.mass = mass;
        this.radius = radius;
    }
}
```

In the above the enum now has two fields, one named mass and one named radius, both of type double.
The enum also has a constructor, which is implicitly private, if you try to create a public constructor you will get a compilation error.
Finally each value now has two parameters, according to the parameters taken by the enums constructor.

Enum Methods

```
public enum Planet {
    MERCURY(3.03e+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;
    private final double radius;

    Planet(double mass, double radius) {
        this.mass = mass;
        this.radius = radius;
    }

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

    public double surfaceGravity() {
        return GRAVITATIONAL_CONSTANT * mass / (radius * radius);
    }

    public double surfaceWeight(double otherMass) {
        return otherMass * surfaceGravity();
    }
}
```

As you can see in the above enum, we now have two methods at the bottom, one calculates surface gravity and the other calculates surface weight.
Both of these methods can now be called from outside of the enum.

Using Methods Within The Enum

We can access the methods within an enum, whilst specifying which value in the enum we want to run the method against.

```
public class Runner {

    public static void main(String[] args) {
        double earthWeight = Double.parseDouble("175");
        double mass = earthWeight/Planet.EARTH.surfaceGravity();
    }
}
```

In the above main method, we are initialising a double variable called earthWeight with a Double value of 175.
Then we are declaring a double variable called mass and setting it to earthWeight divided by the double returned from the method surfaceGravity within the Planet enum.

Iterating Through Enum Values

We can also iterate through the values within the enum, and even call the methods within the enum whilst iterating through.

```
public class Runner {

    public static void main(String[] args) {
        double earthWeight = Double.parseDouble("175");
        double mass = earthWeight/Planet.EARTH.surfaceGravity();
        for(Planet p : Planet.values()) {
            System.out.printf("Your weight on %s is %f%n", p,
p.surfaceWeight(mass));
        }
    }
}
```

In the above main method, we are initialising a double variable called earthWeight with a Double value of 175.
Then we are declaring a double variable called mass and setting it to earthWeight divided by the double returned from the method surfaceGravity within the Planet enum.
Then we are iterating over the values in the enum using a for each loop, and inside the loop we are executing a printf statement to print the mass of the person on each planet within the solar system.
The printf statement is another way to print stuff and allows us to format what we are printing using flags.
The flags used in our printf statement are %s which formats strings, %f which formats floating point numbers, and %n which adds a new line.

Limitations

One limitation of enums is that they extend java.lang.Enum implicitly so our enum types cannot extend from another class.
They are also implicitly final and therefore cannot be extended, or implement interfaces.

If an enum contains fields and methods, the definition of those fields and methods **must** come after the list of constants in the enum, and the list of enum constants **must** be terminated with a semicolon.

Exercises

There are no exercises for this module.