Правда про enum-ы

Юрий Литвинов
yurii.litvinov@gmail.com

27.02.2019г

Enum-ы

- Типобезопасны
- Автоматически получают пространство имен
- Возможны конструкторы и методы

```
public enum Apple { FUJI, PIPPIN, GRANNY_SMITH }
public enum Orange { NAVEL, TEMPLE, BLOOD }

enum Season {
    WINTER,
    SPRING,
    SUMMER,
    AUTUMN
}
```

Методы

- Можно сравнивать с помощью ==
- name(), ordinal(), toString()

```
var season = Season.WINTER;
System.out.println(
    season.name() + ", " +
    season.toString() + ", " +
    season.ordinal()
);
```

WINTER, WINTER, 0

Статические методы

- valueOf()

```
String name = "WINTER";
Season season = Season.valueOf(name);

Season.valueOf(null); // NullPointerException
Season.valueOf("HOLIDAYS"); // IllegalArgumentException
```

- values()

```
System.out.println(Arrays.toString(Season.values()));

[WINTER, SPRING, SUMMER, AUTUMN]
```

- Автоматически добавляются компилятором

Поля

```
enum Type {
    INT(true),
    INTEGER(false),
    STRING(false);

    private final boolean primitive;

    Type(boolean primitive) { this.primitive = primitive; }

    public boolean isPrimitive() { return primitive; }
}
```

Методы

```
enum Direction {
  UP, DOWN;

  public Direction opposite() {
    switch (this) {
      case UP:
        return DOWN;
      case DOWN:
        return UP;
      throw new AssertionError("Unknown op: " + this);
    }
  }
}
```

Методы (constant-specific)

```
enum Direction {
  UP {
      public Direction opposite() { return DOWN; }
  },

  DOWN {
      public Direction opposite() { return UP; }
  };

  public abstract Direction opposite();
}
```

Пример

```
enum Type {
    INT(true) {
        public Object parse(String string) { return Integer.valueOf(string); }
    },
    INTEGER(false) {
        public Object parse(String string) { return Integer.valueOf(string); }
    },
    STRING(false) {
        public Object parse(String string) { return string; }
    };

    private final boolean primitive;

    Type(boolean primitive) { this.primitive = primitive; }

    public boolean isPrimitive() { return primitive; }
    public abstract Object parse(String string);
}
```

Ещё пример

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

    private static final int HOURS_PER_SHIFT = 8;

    double pay(double hoursWorked, double payRate) {
        double basePay = hoursWorked * payRate;
        double overtimePay;
        switch (this) {
            case SATURDAY: case SUNDAY:
                overtimePay = hoursWorked * payRate / 2;
                break;
            default:
                overtimePay = hoursWorked <= HOURS_PER_SHIFT
                        ? 0 : (hoursWorked - HOURS_PER_SHIFT) * payRate / 2;
                break;
        }
        return basePay + overtimePay;
    }
}
```

Паттерн “Стратегия” на enum-ах

```
enum PayrollDay {
    MONDAY(PayType.WEEKDAY), TUESDAY(PayType.WEEKDAY),
    WEDNESDAY(PayType.WEEKDAY), THURSDAY(PayType.WEEKDAY),
    FRIDAY(PayType.WEEKDAY),
    SATURDAY(PayType.WEEKEND), SUNDAY(PayType.WEEKEND);

    private final PayType payType;

    PayrollDay(PayType payType) { this.payType = payType; }

    double pay(double hoursWorked, double payRate) {
        return payType.pay(hoursWorked, payRate);
    }
    ...
}
```

PayType

```
private enum PayType {
    WEEKDAY {
        double overtimePay(double hours, double payRate) {
            return hours <= HOURS_PER_SHIFT ? 0 :
                    (hours - HOURS_PER_SHIFT) * payRate / 2;
        }
    },

    WEEKEND {
        double overtimePay(double hours, double payRate) {
            return hours * payRate / 2;
        }
    };

    private static final int HOURS_PER_SHIFT = 8;

    abstract double overtimePay(double hrs, double payRate);

    double pay(double hoursWorked, double payRate) {
        double basePay = hoursWorked * payRate;
        return basePay + overtimePay(hoursWorked, payRate);
    }
}
```

Битовые поля

Как делали без enum-ов:

```
public class Text {
    public static final int STYLE_BOLD = 1 << 0;
    public static final int STYLE_ITALIC = 1 << 1;
    public static final int STYLE_UNDERLINE = 1 << 2;

    public void applyStyles(int styles) {
        // styles — побитовое "или"
        // text.applyStyles(STYLE_BOLD | STYLE_ITALIC);
    }
}
```

EnumSet

- Все возможности Set
- Внутри — long или long[]
  - Производительность сравнима с битовыми масками

```
public class Text {
    public enum Style { BOLD, ITALIC, UNDERLINE }

    public void applyStyles(Set<Style> styles) {
        // text.applyStyles(EnumSet.of(Style.BOLD, Style.ITALIC));
    }
}
```

EnumMap, пример без

```
public class Herb {
    public enum Type { ANNUAL, PERENNIAL, BIENNIAL }

    private final String name;
    private final Type type;

    public Herb(String name, Type type) {
        this.name = name;
        this.type = type;
    }
}
```

EnumMap, пример без (2)

```
var herbsByType =
    (Set<Herb>[]) new Set[Herb.Type.values().length];
    // Indexed by Herb.Type.ordinal()

for (int i = 0; i < herbsByType.length; i++) {
    herbsByType[i] = new HashSet<Herb>();
}

for (Herb h : garden) {
    herbsByType[h.type.ordinal()].add(h);
}
```

EnumMap, пример с

```
var herbsByType =
    new EnumMap<Herb.Type, Set<Herb>>(Herb.Type.class);

for (Herb.Type t : Herb.Type.values()) {
    herbsByType.put(t, new HashSet<Herb>());
}

for (Herb h : garden) {
    herbsByType.get(h.type).add(h);
}
```

EnumMap EnumMap-ов, пример без

```
public enum Phase {
    SOLID, LIQUID, GAS;

    public enum Transition {
        MELT, FREEZE, BOIL, CONDENSE, SUBLIME, DEPOSIT;

        private static final Transition[][] TRANSITIONS = {
            { null, MELT, SUBLIME },
            { FREEZE, null, BOIL },
            { DEPOSIT, CONDENSE, null }
        };

        public static Transition from(Phase src, Phase dst) {
            return TRANSITIONS[src.ordinal()][dst.ordinal()];
        }
    }
}
```

EnumMap EnumMap-ов, пример с

```
public enum Phase {
    SOLID, LIQUID, GAS;

    public enum Transition {
        MELT(SOLID, LIQUID), FREEZE(LIQUID, SOLID),
        BOIL(LIQUID, GAS), CONDENSE(GAS, LIQUID),
        SUBLIME(SOLID, GAS), DEPOSIT(GAS, SOLID);

        final Phase src;
        final Phase dst;

        Transition(Phase src, Phase dst) { ... }
        ...
    }
}
```

EnumMap EnumMap-ов, пример с (2)

```
public enum Transition {
    ...
    private static final Map<Phase, Map<Phase, Transition>> m =
      new EnumMap<Phase, Map<Phase, Transition>>(Phase.class);

    static {
      for (Phase p : Phase.values()) {
        m.put(p, new EnumMap<Phase, Transition>(Phase.class));
      }
      for (Transition trans : Transition.values()) {
        m.get(trans.src).put(trans.dst, trans);
      }
    }

    public static Transition from(Phase src, Phase dst) {
      return m.get(src).get(dst);
    }
  }
}
```