Effective Java
Item 20

haocheng

Prefer class hierarchies to tagged classes

```
1   public class Figure {
2           enum Shape {
3                   RECTANGLE, CIRCLE
4           };
5
6           // Tag field - the shape of this figure
7           final Shape shape;
8
9           // These fields are used only if shape is RECTANGLE
10          double length;
11
12          double width;
13
14          // This field is used only if shape is CIRCLE
15          double radius;
16
17          // Constructor for circle
18          Figure(double radius) {
19                  shape = Shape.CIRCLE;
20                  this.radius = radius;
21          }
22
23          // Constructor for rectangle
24          Figure(double length, double width) {
25                  shape = Shape.RECTANGLE;
26                  this.length = length;
27                  this.width = width;
28          }
29
30          double area() {
31                  switch (shape) {
32                  case RECTANGLE:
33                          return length * width;
34                  case CIRCLE:
35                          return Math.PI * (radius * radius);
36                  default:
37                          throw new AssertionError();
38                  }
39          }
40  }
```

Verbose

```
1   public class Figure {
2           enum Shape {
3                   RECTANGLE, CIRCLE
4           };
5
6           // Tag field - the shape of this figure
7           final Shape shape;
8
9           // These fields are used only if shape is RECTANGLE
10          double length;
11
12          double width;
13
14          // This field is used only if shape is CIRCLE
15          double radius;
16
17          // Constructor for circle
18          Figure(double radius) {
19                  shape = Shape.CIRCLE;
20                  this.radius = radius;
21          }
22
23          // Constructor for rectangle
24          Figure(double length, double width) {
25                  shape = Shape.RECTANGLE;
26                  this.length = length;
27                  this.width = width;
28          }
29
30          double area() {
31                  switch (shape) {
32                  case RECTANGLE:
33                          return length * width;
34                  case CIRCLE:
35                          return Math.PI * (radius * radius);
36                  default:
37                          throw new AssertionError();
38                  }
39          }
40  }
```

error-prone

```
1   public class Figure {
2           enum Shape {
3                   RECTANGLE, CIRCLE
4           };
5
6           // Tag field - the shape of this figure
7           final Shape shape;
8
9           // These fields are used only if shape is RECTANGLE
10          double length;
11
12          double width;
13
14          // This field is used only if shape is CIRCLE
15          double radius;
16
17          // Constructor for circle
18          Figure(double radius) {
19                  shape = Shape.CIRCLE;
20                  this.radius = radius;
21          }
22
23          // Constructor for rectangle
24          Figure(double length, double width) {
25                  shape = Shape.RECTANGLE;
26                  this.length = length;
27                  this.width = width;
28          }
29
30          double area() {
31                  switch (shape) {
32                  case RECTANGLE:
33                          return length * width;
34                  case CIRCLE:
35                          return Math.PI * (radius * radius);
36                  default:
37                          throw new AssertionError();
38                  }
39          }
40  }
```

Inefficient

```
1   public class Figure {
2           enum Shape {
3                   RECTANGLE, CIRCLE
4           };
5
6           // Tag field - the shape of this figure
7           final Shape shape;
8
9           // These fields are used only if shape is RECTANGLE
10          double length;
11
12          double width;
13
14          // This field is used only if shape is CIRCLE
15          double radius;
16
17          // Constructor for circle
18          Figure(double radius) {
19                  shape = Shape.CIRCLE;
20                  this.radius = radius;
21          }
22
23          // Constructor for rectangle
24          Figure(double length, double width) {
25                  shape = Shape.RECTANGLE;
26                  this.length = length;
27                  this.width = width;
28          }
29
30          double area() {
31                  switch (shape) {
32                  case RECTANGLE:
33                          return length * width;
34                  case CIRCLE:
35                          return Math.PI * (radius * radius);
36                  default:
37                          throw new AssertionError();
38                  }
39          }
40  }
```

Class Hierarchy

```
1    public abstract class Figure {
2            abstract double area();
3    }
```

```java
public class Rectangle extends Figure {

        final double length;

        final double width;

        Rectangle(double length, double width) {
                this.length = length;
                this.width = width;
        }

        @Override
        double area() {
                return length * width;
        }
}
```

```java
public class Circle extends Figure {

        final double radius;

        Circle(double radius) {
                this.radius = radius;
        }

        @Override
        double area() {
                return Math.PI*(radius * radius)
;
        }
}
```

natural
hierarchical
relationships

```
1   public class Square extends Rectangle {
2
3           Square(double side) {
4                   super(side, side);
5           }
6
7   }
```

Thank you!