

第1章 《シンボリック定数によるマジックナンバーの置き換え》 ソースに 100 と書いてあったら

書籍: Java 言語で学ぶリファクタリング入門 p.22~p.40

著者: 結城浩

発表者: 野口耕二郎

目次

- リファクタリング・カタログ
- サンプルプログラム
- 考えを広げるためのヒント
- まとめ

リファクタリング・カタログ

名前	《シンボリック定数によるマジックナンバーの置き換え》
英語名	Replace Magic Number with Symbolic Constant
文脈	定数を使っている
問題	- マジックナンバーは意味がわかりにくい - マジックナンバーが複数箇所に書いてあると、変更しにくい
解法	マジックナンバーをシンボリック定数で置き換えましょう
結果	○定数の意味がわかりやすくなる ○シンボリックな定数の値を変更するだけで、その定数を使っている全ての箇所が自動的に変更される ×わかりにくい名前をつけると、かえって誤解を招くことがある
手順	1.シンボリック定数を宣言する (1)シンボリック定数を宣言する (2)マジックナンバーをシンボリック定数で置き換える (3)そのシンボリック定数に依存する別のマジックナンバーを探し、シンボリック定数を使った式に変換する(←ポイント) (4)コンパイルする 2.テストする (1)全てのシンボリック定数の置き換えが終わったらコンパイルしてテストする (2)可能なら、シンボリック定数の値を変更してからコンパイルしてテストする(←ポイント)
関連	《クラスによるタイプコードの置き換え》(第7章) 定数がタイプコードを表している場合には、クラスで置き換えたほうが有効なことがある 《State/Strategyによるタイプコードの置き換え》(第9章) 定数がタイプコードを表している場合には、State/Strategyパターンが使えることがある

サンプルプログラム

リファクタリング前

```
1 public class Robot {
2     private final String _name;
3     public Robot(String name) {
4         _name = name;
5     }
6     public void order(int command) {
7         if (command == 0) {
8             System.out.println(name + " walks.");
9         } else if (command == 1) {
10            System.out.println(name + " stops.");
11        } else if (command == 2) {
12            System.out.println(name + " jumps.");
13        } else {
14            System.out.println("Command error. command = " + command);
15        }
16    }
17 }
```

Robot.java

赤丸の部分がマジックナンバー

```
1 public class Main {
2     public static void main(String[] args) {
3         Robot robot = new Robot("Andrew");
4         robot.order(0); // walk
5         robot.order(1); // stop
6         robot.order(2); // jump
7     }
8 }
```

Main.java

(1) シンボリック定数を宣言

```
1 public class Robot {  
2     public static final int COMMAND_WALK = 0;  
3     public static final int COMMAND_STOP = 1;  
4     public static final int COMMAND_JUMP = 2;  
5     private final String _name;  
6     public Robot(String name) {  
7         _name = name;  
8     }  
9     public void order(int command) {  
10        if (command == 0) {  
11            System.out.println(_name + " walks.");  
12        } else if (command == 1) {  
13            System.out.println(_name + " stops.");  
14        } else if (command == 2) {  
15            System.out.println(_name + " jumps.");  
16        } else {  
17            System.out.println("Command error. command = " + command);  
18        }  
19    }  
20 }
```

public はクラスの外
からでも参照できるよ
うにするため
static はクラスから直
接利用できるよにな
るため
final は変数を更新で
きないようにするた
め

Robot.java

(2)マジックナンバーをシンボリック定数で置き換える

```
1 public class Robot {
2     public static final int COMMAND_WALK = 0;
3     public static final int COMMAND_STOP = 1;
4     public static final int COMMAND_JUMP = 2;
5     private final String _name;
6     public Robot(String name) {
7         _name = name;
8     }
9     public void order(int command) {
10        if (command == COMMAND_WALK) {
11            System.out.println(_name + " walks.");
12        } else if (command == COMMAND_STOP) {
13            System.out.println(_name + " stops.");
14        } else if (command == COMMAND_JUMP) {
15            System.out.println(_name + " jumps.");
16        } else {
17            System.out.println("Command error. command = " + command);
18        }
19    }
20 }
```

Robot.java

```
1 public class Main {
2     public static void main(String[] args) {
3         Robot robot = new Robot("Andrew");
4         robot.order(Robot.COMMAND_WALK); // walk
5         robot.order(Robot.COMMAND_STOP); // stop
6         robot.order(Robot.COMMAND_JUMP); // jump
7     }
8 }
```

Main.java

黄色で塗った部分を
(1) で宣言したシンボ
リック定数に置き換え
た

(3) そのシンボリック定数に依存する別のマジックナンバーを探し, シンボリック定数を使った式に変換する

- 定数の間に比例などの依存関係があるかどうかを探し, あればマジックナンバーの部分をシンボリック定数に直す

➤例) $RADIUS = 2$ $DIAMETER = 4$ → $RADIUS = 2$ $DIAMETER = 2 * RADIUS$

- 定数の値を変更するとき依存関係にある定数の値も自動的に変更されるので楽が出来る

リファクタリング後

```
1 public class Robot {
2     public static final int COMMAND_WALK = 0;
3     public static final int COMMAND_STOP = 1;
4     public static final int COMMAND_JUMP = 2;
5     private final String _name;
6     public Robot(String name) {
7         _name = name;
8     }
9     public void order(int command) {
10        if (command == COMMAND_WALK) {
11            System.out.println(_name + " walks.");
12        } else if (command == COMMAND_STOP) {
13            System.out.println(_name + " stops.");
14        } else if (command == COMMAND_JUMP) {
15            System.out.println(_name + " jumps.");
16        } else {
17            System.out.println("Command error. command = " + command);
18        }
19    }
20 }
```

Robot.java

```
1 public class Main {
2     public static void main(String[] args) {
3         Robot robot = new Robot("Andrew");
4         robot.order(Robot.COMMAND_WALK); // walk
5         robot.order(Robot.COMMAND_STOP); // stop
6         robot.order(Robot.COMMAND_JUMP); // jump
7     }
8 }
```

Main.java

考えを広げるためのヒント

タイプコードをクラスで置き換える

- タイプコードとは事物を区別するために割り当てた数値のこと
- このリファクタリングでは他のプログラマが, シンボリック定数を置くべき場所にマジックナンバーを置いてもコンパイルエラーが出ない
- そこで, 新たな型を用意してタイプコードでその型を使うようにする
- 他のプログラマがマジックナンバーを使うと型が違いためコンパイルエラーが発生し, シンボリック定数の使用を強制することが出来る

タイプコードをクラスで置き換える

新たに追加したクラス

```
1 public class RobotCommand {
2     private final String _name;
3     public RobotCommand(String name) {
4         _name = name;
5     }
6     public String toString() {
7         return "[ RobotCommand: " + _name + " ]";
8     }
9 }
```

RobotCommand.java

RobotCommand を使
うように変更

```
1 public class Robot {
2     private final String _name;
3     public static final RobotCommand COMMAND_WALK = new RobotCommand("WALK");
4     public static final RobotCommand COMMAND_STOP = new RobotCommand("STOP");
5     public static final RobotCommand COMMAND_JUMP = new RobotCommand("JUMP");
6     public Robot(String name) {
7         _name = name;
8     }
9     public void order(RobotCommand command) {
10         if (command == COMMAND_WALK) {
11             System.out.println(_name + " walks.");
12         } else if (command == COMMAND_STOP) {
13             System.out.println(_name + " stops.");
14         } else if (command == COMMAND_JUMP) {
15             System.out.println(_name + " jumps.");
16         } else {
17             System.out.println("Command error. command = " + command);
18         }
19     }
20 }
```

Robot.java

Main クラスを書き換え
る必要はない

enum の使用

新たに追加した
列挙型
WALK には 0
STOP には 1
JUMP には 2
と 0 から順番に値
が格納されていく

コードがスッキリ
するので読みやす
くなる

```
1 public class Robot {
2     private final String name;
3     public enum Command {
4         WALK,
5         STOP,
6         JUMP,
7     };
8     public Robot(String name) {
9         _name = name;
10    }
11    public void order(Robot.Command command) {
12        if (command == Command.WALK) {
13            System.out.println(_name + " walks.");
14        } else if (command == Command.STOP) {
15            System.out.println(_name + " stops.");
16        } else if (command == Command.JUMP) {
17            System.out.println(_name + " jumps.");
18        } else {
19            System.out.println("Command error. command = " + command);
20        }
21    }
22 }
```

Robot.java

```
1 public class Main {
2     public static void main(String[] args) {
3         Robot robot = new Robot("Andrew");
4         robot.order(Robot.Command.WALK);
5         robot.order(Robot.Command.STOP);
6         robot.order(Robot.Command.JUMP);
7     }
8 }
```

Main.java

黄色で塗った部分を
列挙型を使った定数
に置き換える
若干書き方が変わる
ので注意

シンボリック定数にしないほうがよい場合

- 配列の長さは **length** で得られるのでこの場合はシンボリック定数は必要無い
- 自明な値もわざわざ書くと面倒な上, コードが読みにくくなる可能性があるのでシンボリック定数を使う必要が無い

バイトコードに埋め込まれる定数に注意

- 定数 (**public static final**) があるクラスで宣言されていてそれを使う別のクラスがあるとき、定数を使う別のクラスのバイトコード上にその定数が埋め込まれる
- 定数があるクラスで定数の値を変更してコンパイルしても、定数をつかう別のクラスには再コンパイルされる前の値の定数が埋め込まれたままであり、定数に変更されたことが伝わっていない
- この事態を避けるためには定数があるクラスと定数を使うクラスを一緒にコンパイルする必要がある

まとめ

まとめ

- マジックナンバーをシンボリック定数で置き換えることで値の変更が便利になったり可読性が上がったりする
- マジックナンバーを使わないように強制したい場合はシンボリック定数用のクラスを作る
- シンボリック定数が適さない場合や定数の扱いで注意が必要になることもある

所感

- マジックナンバーはバグの温床になると思うので、今後はできるだけシンボリック定数を使うようにしようと思う
- このリファクタリングではマジックナンバーの数によってはかなり退屈な作業が続きそうだった
- それでもマジックナンバーをなくすことでバグの温床を減らすことが出来るので重要なリファクタリングだった