

Homework #1 Solution

1.

| Refactoring | Smells |
|-------------------------------|------------------------------|
| Duplicate Observed Data | Large Class (B) |
| Extract Class | Large Class (B) |
| Extract Interface | Large Class (B) |
| Extract Method | Comments (A), Long Method(C) |
| Extract Subclass | Large Class (B) |
| Introduce Assertion | Comments (A) |
| Introduce Parameter Object | Long Parameter List (D) |
| Preserve Whole Object | Long Parameter List (D) |
| Rename Method | Comments (A) |
| Replace Parameter With Method | Long Parameter List (D) |

2.

a. Duplication:

- substitution for %CODE% and %ALTCODE% is nearly identical (different destinations for writing)
- 6 is the length of %CODE%, 9 is the length of %ALTCODE%
- Construction of output strings is similar (templatePartOne + code/altcode + templatePartTwo)

b. Removing duplication:

- Extract Method to remove duplication between %CODE% and %ALTCODE%
- Replace Magic Number to use '%CODE%'.length() instead of hardcoding 6 (and the same for %ALTCODE%)
- Extract Method to minimize differences in output methods

3.

a. Code Smells:

- Magic Numbers
- Complex Conditionals

b. De Morgan's Law:

```
if ((score <= 700) &&  
    ((income < 40000) || (income > 100000))  
    || !authorized || (score <= 500) &&  
    (income <= 100000))  
    reject();  
else  
    accept();
```

c. Explaining Variables:

```
boolean hasMiddleRangeIncome = (income >= 40000) && (income <= 100000);  
boolean hasHighIncome = (income > 100000);
```

```

boolean hasHighScore = (score > 700);
boolean hasMidScore = (score > 500);

if (! (hasHighScore || (hasMidRangeIncome && authorized &&
hasMidRangeScore) || hasHighIncome ))
    reject();
else
    accept();

```

d. Flip the if/else clauses:

```

if (score > 700)
    accept();
else if ((income >= 40000) && (income <= 100000) && authorized && (score >
500)
    accept();
else if (income > 100000)
    accept();
else
    reject();

```

e. Consolidate Conditional Expression:

```

boolean acceptable(int income, int score, boolean authorized) {
    if (score > 700) || (income > 100000)
        return true;
    if ((income > 40000) && (income <= 100000) && authorized
&& (score > 500))
        return true;
    return false;
}

if (acceptable(income, score, authorized))
    accept();
else
    reject();

```

f. The simplest and clearest are a matter of opinion. Certainly, these techniques could be combined – pretty much any of the answers to parts b, d, and e could use the explaining variables from part c. The answer for part e is much easier keeping in mind the answer to part d, so they’re somewhat already combined.

4.

```

double getPayAmount() {
    if(_isDead) return deadAmount();
    if(_isSeparated) return separatedAmount();
    if(_isRetired) return retiredAmount();
    return normalPayAmount();
}

```

}

Refactoring used is *Replace Nested Conditional with Guard Clauses*