Object-Oriented Systems Development

Code Smells, Anti-Patterns and Refactoring

Semester 1, 2020

Bad code smells

- Application-level smells
- Class-level smells
- Method-level smells

Application-level smells

- Duplicated code
- Contrived complexity
- Shotgun surgery
- Dead code
- Comments

Class-level smells

- Large class/God object
- Lazy class/freeloader
- Excessive use of literals
- Cyclomatic complexity
- Orphan variable or constant class
- Data clumps

```
def cyclomatic_complexity(n):
    if (n == 0):
        return 'Zero'
    elif (n == 1):
        return 'One'
    elif (n == 2):
        return 'Two'
    elif (n == 3):
        return 'Three'
    elif (n == 4):
        return 'Four'
    else:
        return 'N/A'


def refactor_cyclomatic_complexity(n):
    num = ['Zero', 'One', 'Two', 'Three', 'Four']
    if ((n >= 0) and (n <= 4)):
        return num[n]
    else:
        return 'N/A'

print(cyclomatic_complexity(5))
print(refactor_cyclomatic_complexity(3))
```

Method-level smells

- Too many parameters
- Long method
- Excessively long identifiers
- Excessively short identifiers
- Embedding types in identifiers
- Excessively long line of code/God line

```
# Excessively short identifiers
def x(y, z):
    return y * z


# Excessively long identifiers
def my_name_is_grayson_orr_how_are_you_today():
    return 'Grayson, how are you today?'


# Embedding types in identifiers
def embedding_types():
    list_of_names = ['Grayson', 'John', 'Jane']
    return list_of_names
```

Anti-patterns

- Design patterns
- Boat-anchoring
- Magic numbers and strings
- Spaghetti code
- Copy and pasting programming
- Not invented here syndrome
- Reinventing the square wheel

Refactoring

- Restructuring code
- Make code readable and maintainable
- Code still works
- Semantics are preserved
- Low in coupling and high in cohesion
- Do my unit tests still pass???

Refactoring code examples – class

```
// Class refactoring example

public class Customer {
    private String name;
    private String workPhoneAreaCode;
    private String workPhoneNumber;
}

public class Customer {
    private String name;
    private Phone workPhone;
}

public class Phone {
    private String areaCode;
    private String number;
}
```

Refactoring code examples – subclass

```
// Sub-class refactoring example

public class Person {
    private String name;
    private String jobTitle;
}

public class Person {
    protected String name;
}

public class Employee extends Person {
    private String jobTitle;
}
```

Refactoring code examples – super class

```
// Super-class refactoring example

public class Employee {
    private String name;
    private String jobTitle;
}

public class Student {
    private String name;
    private Course course;
}

public abstract class Person {
    protected String name;
}

public class Employee extends Person {
    private String jobTitle;
}

public class Student extends Person {
    private Course course;
}
```