Question No 3.

**Objective**

Normally, code smell refers as a bad smell. In computer programming that indicate some symptom in the code of the program the probably indicates a deeper problem. According to the Martin Fowler, “this is a surface indication which correlated with the deeper problem of the system. Actually, code smell is not a bug and not any technical incorrect and don't right now keep the program from working of the code. Code smells are indicating the violation of the fundamentals design principle and effects on the quality of design. Code smell indicates the weakness in the quality of design which might be slowing down the programming development or increases the failure and risk of bugs in future. Fowler et al. (1999) suggests refactoring is the way to solve the such kind of trouble which is indicated by the code smell whereas faults refers to the errors into the programming code. With the faults in programs cannot give effective results. Code smell and faults are correlated to each other but code smell is not failure of the program which only indicates that the weakness of the quality in programming code.

**Targeted code bad smells**

Data clumps:

In Object Oriented Programming, Data Clump is the term that provides to the other groups of variables which can be passed around together throughout the program. It is like as other code smells which indicate the various problems in the program design and implementation. In data clumps is considered as specific kind of level of bad code smell that might be the indication of the weak programming code written. In general, those types of code smell should be refactored. The process of reducing data clumps generating a various type of code smell (In a data class that can only stores the data where doesn’t stores any methods for operating on the data; however, new programmer will encourage to see the actual functionality which might be included in the program.

Message chains:

This type of code smell arises when the number of classes are highly coupled to the other multiple classes in like a chain form. To demonstrates this, for example we have class A who want to take data from class E. For retrieving data, firstly object A needs to retrieve object E from object D Object C object B. The chain seems like that

a.get B(). get C(). get D(). get E();

here, when A want to access the information from Class E, A unnecessarily access data from B C D along in the way, when it wants data only to get form E. Message chains code smell could be refactored.

Middle man:

Middle man also be a code smell. It is a class which delegates most of its function to other classes. This smell can be the result of the essential task of the class being gradually moved to other classes.

Speculative Generality

Switch statements:

The switch statement also be a code smell. In this smell, programmer relatively used to switch and case major operators which is the major marks of object oriented program. In Switch statement code, the switch operators could be distributed in various place in the program. When the programmer wants to put new condition or requirements, have to find from all switch code and modify it. The switch statement code smell should be refactored. As a rule of thumb, we can solve this smell through polymorphism.

From the different data sets, I choose five bugs form Apache common package.

**Five different Bugs**

**Bug 1:**

# From common IO: 2.5 ExceptionInInitializerError

# Status: Resolved

# Priority: Major

# Created: 09/May/2017

# Updated: 17/May/2017

# Resolved: 17/May/2017

Description:  
In its static block, org.apache.commons.io.Java7Support executes:

ClassLoader cl = Thread.currentThread().getContextClassLoader();

This can be null.

In that case, I believe the class should fall back to using:

Java7Support.class.getClassLoader();

But someone with a better understanding of the security implications should weigh in on that change.

Thus, here the Message box code smell arise. So, the company is resolved this problem.

Bug 2:

Form common logging: **Wrong sample code in org/apache/commons/logging/package.html (using static and this.class)**

Status: closed

Priority: Minor

Created: 10/April/2003

Updated: 29/December/2009

Resolverd: 29 December/2009

Description:

In the org/apache/commons/logging/package.html file, there is sample code:

public class Foo

{ static Log log = LogFactory.getLog(this.class); ... }

This code cannot be compiled because of using "static" and "this.class" in the   
same statement. Should be change to Foo.class.

Thus, here data clumps code smell arises.

Bug 3:

From Common Net: SubnetUtils.SubnetInfo.isInRange("0.0.0.0") returns true for CIDR/31, 32

Status: Resolved

Priority: Minor

Created: 09/July/2017

Updated: 04/August/2017

Resolved: 04/August/2017

Description:

Code:  
import org.apache.commons.net.util.SubnetUtils;

public class A {  
public static void main(String[] args)

{

System.out.println(new SubnetUtils("192.168.1.0/30").getInfo().isInRange("0.0.0.0")); System.out.println(new SubnetUtils("192.168.1.0/31").getInfo().isInRange("0.0.0.0")); System.out.println(new SubnetUtils("192.168.1.0/32").getInfo().isInRange("0.0.0.0")); }

}

Result:  
false  
true  
true

Expected:  
false  
false  
false

Thus, the Duplicate code smell arises here which can be resolved.

Bug 4: