BCPR301 – Advanced Programming

Contents

[1. NAME: 4](#_Toc514790728)

[Location: 4](#_Toc514790729)

[Reason: 4](#_Toc514790730)

[Strategies / Approaches: 4](#_Toc514790731)

[Refactoring Step Taken: 4](#_Toc514790732)

[2. NAME: 4](#_Toc514790733)

[Location: 4](#_Toc514790734)

[Reason: 4](#_Toc514790735)

[Strategies / Approaches: 4](#_Toc514790736)

[Refactoring Step Taken: 4](#_Toc514790737)

[3. NAME: Large Class: Divergent Change 5](#_Toc514790738)

[Location: 5](#_Toc514790739)

[Reason: 5](#_Toc514790740)

[Strategies / Approaches: 5](#_Toc514790741)

[Refactoring Step Taken: 5](#_Toc514790742)

[4. NAME: Large Class: Shotgun Surgery 5](#_Toc514790743)

[Location: 5](#_Toc514790744)

[Reason: 5](#_Toc514790745)

[Strategies / Approaches: 5](#_Toc514790746)

[Refactoring Step Taken: 5](#_Toc514790747)

Assessment Four- Bad Smells Documentation

# NAME: BLOASTERS

Large Class:

# Location:

Folder: Employee Management System> Employee Database > Employee-Management-System > Refactoring

File: employee\_data.py

Line: 99 to 156

# Reason:

1. In my Assignment one, I think “Large Class” is the second most worst code smell as the “EmployeeData” class is too large.
2. Too many functions and arrays.
3. I found this is the code smell as class is very large, very hard to understand as well as any changes or update to the class will make the class more clumsy
4. it could also break the “Open /Closed principle”

# Strategies / Approaches:

Extract Subclass

# Refactoring Step Taken:

1. Create “GetEmployee(EmployeeData) class
2. Import employee\_get\_data module in InterpreterController(cms) class
3. Create Instance of the GetEmployee() in InterpreterController(cmd) class
4. Use the fields (instance variable) from the super class
5. Unit test
6. Run Test

# NAME: OBJECT ORIENTED ABUSDERS

Refuse Bequest

# Location:

Folder: Employee Management System> Employee Database > Employee-Management-System > Refactoring

File: employee\_get\_data.py

Line: 6 to 71

# Reason:

1. While try to remove first worst code smell “Large Class: EmployeeData”; I have abstract “Subclass:GetEmployee”.
2. The Subclass “GetEmployee” has nothing common with the super class “EmployeeData”
3. The super class and sub class is completely different
4. A subclass, that inherits from a parent class, but the subclass does not need all behaviour provided by the parent class. Because of that, the subclass refuses some behaviour (bequest) of the parent class. That's why this is a code smell.

# Strategies / Approaches:

Replace Inheritance with Delegation

# Refactoring Step Taken:

1. Remove the Inheritance from subclass
2. So both class are independent now, only the super class will be the help of sub class
3. Create a field in the subclass for holding the super class.
   1. Def \_\_init\_\_(self):

Super().\_\_init\_\_()

1. The super() builtin returns a proxy object, a substitute object that has ability to call method of the base class via delegation. This is called indirection (ability to reference base object with super())

# NAME: Large Class: Divergent Change

# Location:

# Reason:

# Strategies / Approaches:

# Refactoring Step Taken:

# NAME: Large Class: Shotgun Surgery

# Location:

# Reason:

# Strategies / Approaches:

# Refactoring Step Taken: