Directory Management System Project

# Phase I Part 1

*Use Astah to draw a class diagram diagram. Use proper UML notation. Take a clear screenshot of your completed diagram and paste it on this page.*

A diagram of a work flow

Description automatically generated

# Phase I Part 2

*Identify the places in the code where there are object-oriented concept violations, content coupling, common coupling, control coupling, and stamp coupling situations. On this page, paste the code segments that correspond to each situation and explain how you would fix object-oriented concept violations, common coupling, control coupling, and content coupling issues. You may add pages if necessary.*

## Object Oriented Concept Violations

1. Public Property in Personnel Class

public ArrayList<Person> personList;

1. Public Properties in Person Class

public String last;  
public String first;  
public String middle;

These properties allow direct modification from outside classes. To fix this issue, I would make them private and allocate their respective getter/setter functionality.

## Content Coupling

Person p1 = new Person(lastN, firstN, " ");  
per.personList.add(p1);  
total.objectAdded();

PersonnelDirectory directly accesses the personList inside Personnel. Instead of directly manipulating the inner states of other classes, the PersonnelDirectory should make method calls to achieve its goals.

System.*out*.println("Enter the order 0: first, middle, last, 1: first, last, middle, 2: last, first , middle ");  
int order = scan.nextInt();  
for (int i = 0; i < per.personList.size(); i++) {  
 per.personList.get(i).printName(order);  
}

In this example, the PersonnelDirectory accesses the inner workings of Personnel to use printName, a method inside the list of Person objects. To fix this issue the Personnel object should encapsulate all Persons so thay only Personnel knows how to handle Person objects.

for (int i = 0; i < per.personList.size(); i++) {  
 if (per.personList.get(i).first.equals(firstN) && per.personList.get(i).last.equals(lastN)) {  
 found = true;  
 loc = i;  
 }  
}

PersonnelDirectory knows of and is directly accessing the internal structure of the Personnel class's personList. To fix this issue, we can implement methods within the Personnel class, like findPerson, that abstract away the internal workings instead of accessing personList directly.

## Common Coupling

Static property numObjects in Class totalObjects

private static int *numObjects* = 0;

The static numObjects in the totalObjects class can be modified by multiple class instances, making it challenging to determine which instance affected its value. Making it static introduces dependencies and unexpected behavior.

If the intent is to track the total number of objects, it's essential to ensure that only one instance of the totalObjects class can be created (i.e., implement it as a Singleton) or find another mechanism to count objects without using a static variable in this manner.

## Control Coupling

System.*out*.println("Enter the order 0: first, middle, last, 1: first, last, middle, 2: last, first , middle ");  
int order = scan.nextInt();  
for (int i = 0; i < per.personList.size(); i++) {  
 per.personList.get(i).printName(order);  
}

PersonnelFactory determines the order the Person list will execute and also knows the intricate workings of Person and controls the format that it will print. If Person were to change the control method of printing the Person, then we would need to shift PersonnelDirectory's interface.

## Stamp Coupling

Employee e1 = new Employee(lastN, firstN, middleN, empID, salary);  
per.addPersonnel(e1);

And

Person p1 = new Person(lastN, firstN, " ");  
per.personList.add(p1);

In these examples, the Personnel Object receives the objects as parameters. But now we have a dangling reference to the object in PersonnelDirectory as well. Instead what we can do is have the Personnel object handle all knowledge about Person objects. This could also be an example of Common coupling because now we have references to an internal object of Personnel inside PersonnelDirectory.

# Phase II Part 2

*After you have incorporated the PersonnelFactory, use Ashta to draw a UML class diagram of the Personnel Directory. Use proper UML notation. When you have completed your diagram, take a clear screenshot of your diagram and paste it on this page.*

A screenshot of a computer

Description automatically generated