

# Directory Management System Project

## Purpose:

Students will evaluate a system with object-oriented design and coupling violations. Students will gain hands-on experience with improving the software system's quality and modifiability by refactoring the system to remove object-oriented design and coupling issues without changing the system's intended functionality. An excellent opportunity to gain further exposure to the topics discussed in the unit (including object-oriented application design, modularity, coupling, and cohesion), students completing this project will also get to develop UML class diagrams that represent the design and implement object-oriented design in Java.

## Objectives:

Students will be able to:

- Evaluate an object-oriented design in the form of a class diagram
- Identify object-oriented design violations
- Use proper UML design tools to develop class diagrams
- Refactor a given code to eliminate object-oriented design violations, including coupling, cohesion, and object design principles
- Implement Java code based on a UML class diagram

## Technology Requirements:

- Astah
- Java
- PDF conversion tool

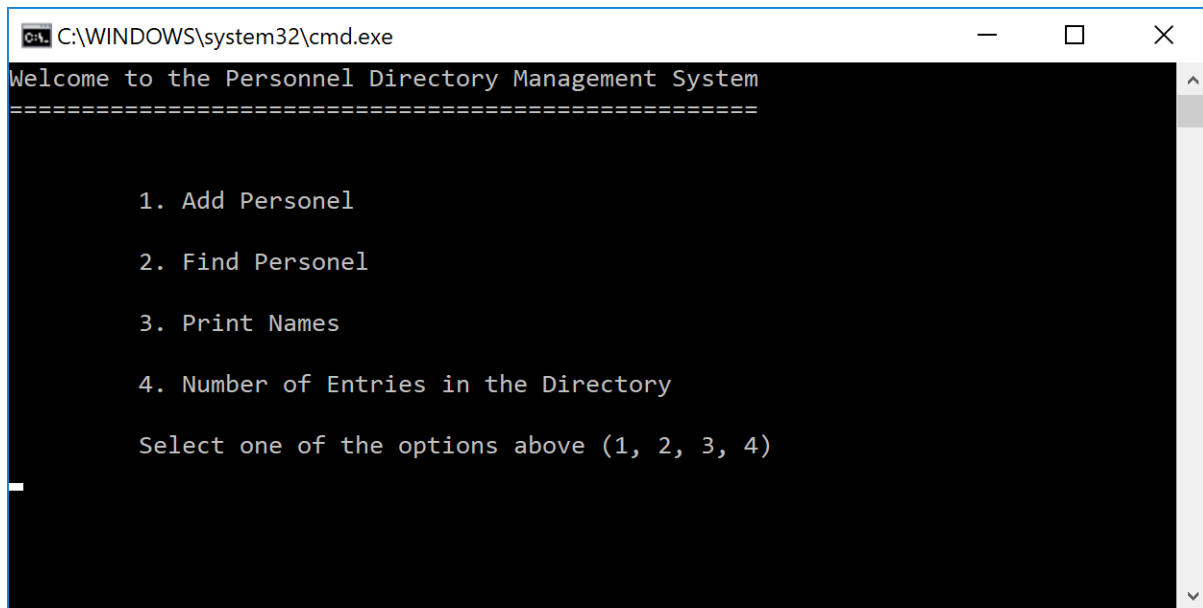
## Project Overview:

**Phase I:** Evaluate the given design's object-oriented design and coupling violations

**Phase II:** Apply a design pattern and refactor the design and implementation using Java

## Project Description:

Review the Personnel Directory Management System implementation provided in the project downloads. This implementation has several object-oriented concept violations and coupling issues.



```
C:\WINDOWS\system32\cmd.exe
Welcome to the Personnel Directory Management System
=====

1. Add Personel

2. Find Personel

3. Print Names

4. Number of Entries in the Directory

Select one of the options above (1, 2, 3, 4)
```

## Submission Directions for Project Deliverables

Use the Directory Management System Submission document to submit the diagrams and responses in Phases I and II

**Phase I:** Submit a zip file containing following items

For questions 1 and 2, save the submission as a **single** PDF titled “Last Name\_First Name\_Directory Management System Project\_Phase I\_Submission”.

For Phase I Part 3 (question 3), submit your refactored code files as a **separate** zip file titled “Last Name\_First Name\_Directory Management System Project\_Phase I\_Part 3\_Submission”.

**Phase II:** Submit a zip file containing following items

For questions 1 and 2, Submit this as a PDF titled “Last Name\_First Name\_Directory Management System Project\_Phase II\_Submission”.

For Phase II Part 3 (question 3), submit your code files as a separate zip file titled “Last Name\_First Name\_Directory Management System Project\_Phase II\_Part 3\_Submission”.

## Phase I - Directions:

Refer to the Directory Management System Submission document to complete Phase I Parts 1 and 2.

1. Use the Astah tool to draw the class diagram for the current implementation of the university system. Use correct UML notations. When you have completed the diagram, take a clear screenshot and paste it into the Directory Management System Submission Phase I document.
2. Identify the places in the code where there are object-oriented concept violations, content coupling, common coupling, control coupling, and stamp coupling situations. In the Directory Management System Submission Document, 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.
3. Refactor the code to remove the object-oriented concept violations, common coupling, control coupling, and content coupling issues **without** removing any intended system functionalities.
4. Save and submit your refactored code as a zip file titled "Last Name\_First Name\_Directory Management System Project\_Phase I\_Part 3\_Submission".

## Phase II - Directions:

In the current implementation, two types of personnel objects have developed: Person and Employee. Suppose the directory will be extended with a few other types, including Executives, Security, and Volunteers. In order to accommodate future extensions to multiple personnel categories, incorporate the factory pattern so that after calling the createPersonnel method of the PersonnelFactory, the Personnel Directory calls the PersonnelFactory to return the appropriate personnel type.

### Directions:

1. Design the PersonnelFactory class to satisfy these requirements.
2. After you have incorporated the PersonnelFactory, draw the UML class diagram of the Personnel Directory using Astah. Take a clear screenshot and include it in the Directory Management System Phase II Submission in the provided space. Submit this as a PDF titled "Last Name\_First Name\_Directory Management System Project\_Phase II\_Submission".
3. Implement the new PersonnelDirectory system that complies with class diagram in Part B above using Java. Submit your files as a single zip file titled "Last Name\_First Name\_Directory Management System Project\_Phase II\_Part 3\_Submission".