Object-Oriented Programming
Group Assignment #5
Adopt Me!

CMP_SC/INFO_TC 3330

Spring 2025

Objective

The goal of this assignment is to develop a Java-based desktop application that applies key Object-Oriented Programming (OOP) concepts in an engaging and practical project. Students will build a Pet Adoption Center management system, leveraging Swing for the GUI and Maven for project structure and dependency management.

Project Summary

Students will build a system to manage adoptable pets, including `Dog`, `Cat`, `Rabbit`, and `ExoticAnimal`. The app will support adding, viewing, sorting, and adopting pets, while maintaining clean software architecture and applying design principles.

Concepts Covered

- Java Generics
- Adapter Design Pattern
- SOLID Principles
- Comparator and Comparable interfaces
- Model-View-Controller (MVC) pattern
- Swing GUI development
- Maven project setup

Scenario

The Pet Adoption Center needs a desktop tool to manage its list of available pets. Occasionally, exotic pets are provided through a third-party system that does not follow the same structure. Your task is to create a unified application to manage all types of pets using clean OOP principles.

Requirements

Functional Features

- Load the pets and the exotic pets from the given JSON files (*pets.json* and *exotic_pets.json*) when the program starts. These files must go to the "*src/main/resources*" directory of your Maven project.
- Use the `Gson` library to work with JSON files. The library must be added as a dependency to your Maven project. Adding it as a jar is invalid. Use the 2.13.1 version.
- Add, remove, and view adoptable pets.
- Sort pets by name, age, or species.
- Adopt a pet. Update the adopted pet's status, and once it is adopted, it cannot be adopted again (prompt message on multiple attempts).
- Import exotic pets via a third-party format using an Adapter.
- Save the pet list back to JSON file. For the saved file name, use the current date and time when the save button is clicked. Ex: *YYYYMMDD_HHMMSS_pets.json*

Technical Implementation

- **Generics:** Create a generic `Shelter<T extends Pet>` class to store pets.
- **Adapter Pattern:** Wrap a third-party `ExoticAnimal` class to make it compatible with the `Pet` abstract class.
- **SOLID Principles:** Follow Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, and Dependency Inversion principles throughout the codebase.
- **Comparator & Comparable:** Implement sorting logic using both `Comparable` (default: by name) and multiple `Comparator` classes (e.g., by age or species).
- **MVC:** Structure the project using MVC:
  - **Model:** Pet data classes and Shelter logic.
  - **View:** Swing-based GUI.
  - **Controller:** Connect GUI events to model updates.
- **Swing GUI:** Include:
  - List or table of pets
  - Buttons for actions (Add, Adopt, Remove, View Details, Save)
  - Combobox for sort by category selection
  - Dialogs for adding/viewing details
- **Maven:** Use Maven for dependency management and project structure.

Important Notes

- Start early!
- Class and method details are not given this time. You need to apply the design principles and good programming practices you have learned in this class so far. This is the main objective of this assignment. Therefore, asking for design suggestions or validation from the TAs or the instructor before the due date is a form of collaboration and assessment. The TAs or the instructor cannot answer these questions. Please note that there could be multiple designs and solutions. The important point is that your design choices must be meaningful and valid (see SOLID, YAGNI, design patterns, MVC)
- You can write additional classes and helper methods.
- Follow Java naming conventions, or you will lose points.
- Use packages, or you will lose points.
- Add Javadoc to your code, or you will lose points.
- Export your project properly, or you will lose points.
- Don't want to 1 commit project, or commit messages like "*Adding Java code*" or "*Update code*", otherwise you will lose points. Commits must be small and meaningful with a commit message that is relevant to the code you pushed. Only I am allowed to do the above, because I am the professor of this class, and I can do whatever I want. This is my class :D
- Write your code considering edge cases. Make sure you have error controls.
- Don't ask how many points will be deducted for the notes above. There is no negotiation here. These are good practices that you must adopt and follow to have a successful career. You can try to violate one of the good practices above and see what happens :) (not recommended).
- Everyone in the group must contribute to the project. Use Git efficiently and communicate!
- If there is a group drama, you have to wait until the next group assignment to split from your group or work alone. See syllabus for details.
- **Due date:** 5/4/2025, 11:59 PM.
- **Submission:** You must submit your GitHub repository and your exported project through Canvas. Submit your GitHub link repository in a file along with your project file submission.

Grading Rubric

Category	Points
Project Setup and Structure	15
• Maven project correctly structured (src/main/java, src/main/resources, pom.xml) (5 pts) • Dependencies (Gson, etc.) added through Maven properly (5 pts) • Proper use of packages and Java naming conventions (5 pts)	
OOP Principles and Design	25
• Proper application of Generics in Shelter class (5 pts) • Correct implementation of Adapter pattern for ExoticAnimal (5 pts) • Application of SOLID principles across codebase (10 pts) • Correct use of Comparator and Comparable for sorting (5 pts)	
Functionality	30
• Loading pets and exotic pets from JSON on startup (5 pts) • Adding, adopting, removing pets with appropriate status handling (5 pts) • Sorting pets by name, age, or species via GUI (5 pts) • Viewing pet details and error handling (5 pts) • Saving updated pet list to a timestamped JSON file (5 pts) • Handling edge cases (invalid input, already adopted pets, etc.) (5 pts)	
Swing GUI and MVC Implementation	20
• GUI with list/table of pets and action buttons (5 pts) • MVC structure clearly separating Model, View, Controller (5 pts) • Dialogs for adding and viewing pet details (5 pts) • Smooth user interaction (no freezing, clear messages, basic usability) (5 pts)	
Code Quality and Good Practices	10

• Clean, readable, and well-structured code (variable names, indentation) (3 pts) • Meaningful commit messages and multiple commits (3 pts) • Proper Javadoc comments for classes and methods (4 pts)	
Penalty Points	-
• Missing packages (-5 pts) • Missing Javadoc (-5 pts) • Unstructured Maven project (-5 pts) • Poor Git usage (1 commit, poor messages) (-5 pts) • Hardcoding values where not appropriate (-5 pts) • Missing meaningful error handling (-5 pts)	