Report: Prototype Pattern Demo

Student full name: Tran Anh Luat

Student ID: 2131200009

Course: CSE 456 - Advanced Java Programming

Pattern: Prototype Pattern

Date: 2025-08-22

**1. Pattern description:**

In modern software engineering, **design patterns** provide proven solutions to common problems in object-oriented design. One such pattern is the **Prototype Pattern**, which focuses on object creation. Instead of instantiating new objects directly via constructors, this pattern allows the system to create new objects by **cloning existing instances**.

This approach is particularly useful when:

* The cost of creating a new object is high.
* Objects share many common attributes, and variations can be introduced after cloning.
* The system needs to dynamically decide which “type” of object to create at runtime.

**Intent**: Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.

**Key Participants**:

* **Prototype (interface/abstract class)**: Declares the clone operation.
* **ConcretePrototype**: Implements the clone operation. Examples: Resume, Report, Invoice.
* **Client**: Uses the prototype registry to create new objects by cloning existing prototypes.

The pattern promotes **loose coupling** between client code and the concrete classes.

**2. Implementation details:**

* **Prototype interface:** declares clonePrototype() and getDescription().
* **Document abstract:** common fields title, content, tags and helper for deep copy.
* **Concrete Prototypes**(Resume, Report, Invoice): implement clonePrototype() performing deep copy of mutable fields.
* **PrototypeRegistry:** stores master prototypes by key.
* **Prototype-app** contains PrototypeApp (Swing GUI) and PrototypeDemo (console) to demo operations:

**Client Application (**PrototypeApp**)**

Provides a **Swing-based GUI** to interact with prototypes and clones.

Features:

Register new prototypes manually.

Clone existing prototypes.

Edit and display clones.

Save/load both prototypes and clones in **JSON format** using Gson.

Auto-save registry to user’s home directory on exit.

Validation for duplicate keys when loading prototypes from JSON.

## 3.Benefits of the Prototype Pattern:

* **Simplifies object creation**: New objects are created by cloning instead of complex constructors.
* **Extensibility**: New types of documents can be introduced without changing the client code.
* **Runtime flexibility**: Prototypes can be registered dynamically, giving the application flexibility in handling unknown types.
* **Consistency**: Clones maintain consistent behavior with their prototypes.

**4. Challenges:**

Ensuring deep copy for lists (tags, experiences) so clones don't share state with the prototype. Designing a small but clear GUI for demo purposes.

**5. Conclusion:**

The implemented Java application illustrates the Prototype Pattern in a practical scenario of managing documents. By combining **prototype registry, cloning mechanism, Swing GUI, and JSON persistence**, the system demonstrates how the Prototype Pattern simplifies object creation and provides flexibility for future extension.

This pattern is particularly beneficial in situations where object creation is costly or when the system requires dynamic, runtime-determined types. The presented application can serve as a learning tool for students and developers exploring design patterns in Java.