1. Identify and design the necessary classes and components:
   * VendingMachine: Responsible for handling transactions, tracking inventory, and financial information.
   * Can: Represents a flavor of the can with a price.
   * PaymentProcessor: Handles payment processing and keeps track of cash and credit card payments.
   * InventoryManager: Tracks the number of cans available and provides restocking functionality.
   * GUI: Develop the graphical user interface for the vending machine.
2. Implement the classes adhering to the SOLID principles:
   * Ensure each class has a single responsibility.
   * Design classes to be open for extension but closed for modification.
   * Ensure Liskov substitution by carefully designing class hierarchies, if necessary.
   * Segregate interfaces based on specific functionality and dependencies.
   * Apply dependency inversion by depending on abstractions and using dependency injection where appropriate.
3. Implement necessary tests:
   * Write unit tests for each class to verify its behavior and ensure correctness.
   * Use mocking or stubbing techniques to isolate dependencies during testing.
4. Develop the GUI:
   * Implement the graphical user interface based on the requirements.
   * Integrate the GUI with the underlying vending machine logic.
5. Refactor and optimize the codebase:
   * Review the code for readability, maintainability, and adherence to SOLID principles.
   * Eliminate code smells and improve overall design.

Assumptions

1. The vending machine supports only a fixed number of flavors for the cans.
   * This assumption is based on the statement that the vending machine can stock up to 10 different flavors of cans. It implies that the number of flavors is fixed and cannot be dynamically changed during runtime.
2. All cans in the vending machine have the same price.
   * This assumption is based on the statement that the value of cans may be decided by you at design time. It suggests that all cans in the vending machine have a uniform price and do not vary based on flavor or other factors.
3. The GUI will display real-time information about the vending machine's state.
   * This assumption is inferred from the requirement to visually display the number of available cans, the money held in the machine, and the credit card payments. It implies that the GUI should reflect the current state of the vending machine, including any changes made during transactions or restocking.
4. Cash and credit card payments are made with the correct change.
   * This assumption is mentioned in the requirement that states the vending machine expects cash and card payments to be made with the correct change. It suggests that the system does not handle providing change or handling scenarios where incorrect payment amounts are made.
5. The vending machine's data is stored in memory during runtime and not persisted.
   * This assumption is based on the statement that the information about the number of cans available, the money held in the machine, and credit card payments does not necessarily need to be persisted. It implies that the data is maintained in memory for the duration of the program's execution.