# Experiment – 10 ( Improvised )

1. **Create an OnlineStore class with the following structure:**
   a) **Order Placement:** The user can attempt to place an order for an item. The method will check if the item is in stock. If it's not, it will throw a custom exception.
   b) **Payment Processing:** Once an item is in stock, the system will process the payment. If the payment fails (e.g., insufficient funds), another custom exception will be thrown.
   c) **Order Confirmation:** If the order is successful, the system will confirm the order. Otherwise, it will catch any exceptions, display appropriate messages, and indicate that the process has ended.

   **Key Points to be covered**

   a) **Define Custom Exception Classes:**
      i. **OutOfStockException**: Thrown if the item is out of stock.
      ii. **PaymentFailureException:** Thrown if there's a problem with payment (e.g., insufficient funds).
   b) **Implement OnlineStore Class:** The **OnlineStore** class will have methods for:
      i. Checking item stock.
      ii. Processing payment.
      iii. Placing an order (combines checking stock and processing payment).
   c) **Main Method to Test:** A main() method will simulate a test case of placing an order, including handling exceptions and finalizing the process.

**2.** Create a system where users try to access age-restricted content, such as a movie streaming platform that restricts access to certain movies or shows based on the user's age. The platform will have a method to check a user's age before allowing access, and if the user is below the required age, it will throw a custom exception, **AgeRestrictionException**.

**Problem Description:**

a) **User Access to Age-Restricted Content**: We will implement a class called **ContentAccessSystem** that manages content viewing access. The user will try to view age-restricted content, such as an 18+ movie.

b) **Age Restriction Check**: The system will check the user's age to see if it meets the minimum age requirement for the content. If the user is underage, the system will throw an **AgeRestrictionException**.

c) **Exception Handling**: When the exception is thrown, the system will catch it, display an appropriate message, and stop the user from accessing the content.

**Steps to Implement:**

a) **Define the Custom Exception**: We will define an **AgeRestrictionException** class that takes a custom message as a parameter to explain why the exception was thrown (e.g., "Access denied - age below minimum requirement").

b) **Create ContentAccessSystem Class**: This class will have:

   i. A method **checkAgeForContent()** that verifies if the user meets the minimum age requirement for viewing certain content.

   ii. A main method to simulate a user trying to access restricted content.

❖ **NOTE**

**These problems allow room for creativity. Focus on implementing a technically sound, clear, and simple solution. Avoid unnecessary complexity, but feel free to add enhancements like more exceptions or extra features. Keep it functional and easy to understand, while leaving space for future improvements**.