1. How Inheritance was Used to Avoid Redundancy:

Inheritance was used to create a common BankAccount base class that provides the shared functionality (such as deposit, withdraw, and account_info). The SavingsAccount and CheckingAccount classes both inherit from BankAccount, allowing them to reuse these common methods. This avoids redundancy by centralizing the shared behavior in the base class. Specific behaviors for savings and checking accounts, such as interest and transaction fees, are implemented in their respective subclasses.

2. How the withdraw() Method Was Overridden in CheckingAccount:

The withdraw() method in the CheckingAccount class was overridden to deduct both the withdrawal amount and a transaction fee. In this class, the total amount withdrawn includes both the requested amount and the fee. This is implemented by calculating the total withdrawal as amount + transaction_fee and ensuring that there are sufficient funds for this combined amount before proceeding with the withdrawal.

3. Optimizations and Design Decisions:

- The base class contains the common methods for depositing and withdrawing, reducing code duplication.
- The SavingsAccount class uses its method apply_interest() to apply interest, which keeps the interest-specific logic separate and simple.
- The CheckingAccount class overrides the withdraw() method efficiently without rewriting other methods like deposit() that remain unchanged from the base class.

Test Cases:

1. **Standard BankAccount**:
    - Create a bank account for Shubham.
    - Deposit 500 and check the balance.
    - Withdraw 200 and check the balance again.
2. **SavingsAccount**:
    - Create a savings account for Shubham_1 with an initial balance of 1000.
    - Apply 2% interest and check that the new balance reflects this increase.
3. **CheckingAccount**:
    - Create a checking account for Shubham_2 with an initial balance of 500.
    - Withdraw 100 (which includes a 1 transaction fee) and ensure the balance is reduced by 101.