1. **Customer Class:**
    - Private fields: **customerName** and **customerAge**.
    - Getter and setter methods for these fields.
2. **Account Abstract Class:**
    - Protected fields: **balance**, **accountId**, **accountType**, and **custobj** (a reference to the associated customer object).
    - Abstract method **withdraw(double amount)** that must be implemented by subclasses.
    - Getter and setter methods for the fields.
3. **SavingsAccount Class (extends Account):**
    - Additional private field: **minimumBalance**.
    - Implements the **withdraw** method to ensure withdrawals maintain the minimum balance.
4. **Bank Class:**
    - Uses a **Scanner** for input.
    - Has instances of **SavingsAccount** and **Customer**.
    - Methods for creating accounts, withdrawing funds, depositing funds, checking balances, and displaying account information.
5. **Case Class (main class):**
    - Creates instances of **Bank** and **SavingsAccount**.
    - Uses a menu-driven approach to interact with the banking system.
    - Options include creating an account, displaying account details, checking balance, depositing funds, withdrawing funds, and exiting.

**Explanation of Operations:**

- The user can create an account by providing personal and account information.
- Account details can be displayed, balance can be checked, and funds can be deposited.
- Withdrawals are subject to a maximum limit of Rs. 20,000 and ensure that the minimum balance is maintained.
- The program runs in a loop until the user chooses to exit.

In summary, this program demonstrates basic banking operations with an emphasis on encapsulation and abstraction. It allows users to interact with a simplified banking system through a command-line interface