🔒 ENCAPSULATION – 15 SCENARIO QUESTIONS

1. Bank Locker Security System

Create a Python class LockerAccount using encapsulation.

- **Private variables**: __locker_pin, __locker_balance
- **Public variables**: holder_name, locker_id
- Use a **parameterized constructor**
- Create a **public method** show_holder_details()
- Create a **private method** __verify_pin()
- Create a **public method** display_balance()

Condition:
If __locker_pin == 2580
→ print "Locker Balance:" and balance
Else → print "Invalid PIN – Access Blocked"

---

2. Online Wallet Application

Create a class DigitalWallet.

- **Private variables**: __wallet_balance, __security_code
- **Public variables**: user_name, wallet_id
- Constructor with parameters
- Public method: show_user_info()
- Private method: __validate_code()
- Public method: check_balance()

Condition:
If __security_code == 7777
→ print balance
Else → print "Security Check Failed"

---

3. Student Result Portal

Create a class ResultPortal.

- Private: __marks, __access_key

- Public: student_name, roll_number
- Public method: display_student_details()
- Private method: __check_key()
- Public method: display_marks()

Condition:
If __access_key == 9999 → show marks
Else → print "Unauthorized Access"

---

4. Mobile Phone Lock System

Class name: SmartPhone

- Private: __screen_pin, __storage
- Public: brand, model
- Public method: phone_details()
- Private method: __unlock_phone()
- Public method: show_storage()

Condition:
PIN must be 4321

---

5. Employee Payroll System

Class: Payroll

- Private: __basic_salary, __admin_code
- Public: employee_name, emp_id
- Public method: employee_details()
- Private method: __verify_admin()
- Public method: calculate_salary()

Condition:
Admin code must be "PAY123"

6–15 (same format, different scenarios)

6. ElectricityBilling – private __units, __meter_pin
7. ATMTransaction – private __balance, __pin
8. OnlineExam – private __score, __exam_code
9. LibraryAccount – private __fine_amount, __password
10. HotelBooking – private __room_charge, __booking_pin
11. CarService – private __service_cost, __service_code
12. InsurancePolicy – private __policy_amount, __policy_key
13. MovieBooking – private __ticket_price, __confirm_code
14. HospitalBilling – private __bill_amount, __auth_code
15. InternetBanking – private __balance, __login_pin

🔁 POLYMORPHISM – 15 SCENARIO QUESTIONS

1. Payment System

Create classes UPIPayment and CardPayment.

- Both must have method process_payment()
- Each method prints a different payment message
- Demonstrate polymorphism using a common function call

---

2. Notification System

Create classes EmailNotification and SMSNotification.

- Method name: send_notification()
- Output must differ based on object

---

3. Vehicle Speed System

Classes: Car, Train

- Method: max_speed()
- Same method name, different output

---

4. Shape Area Calculation

Classes: Circle, Rectangle

- Method: calculate_area()

---

5. Login System

Classes: PasswordLogin, BiometricLogin

- Method: authenticate_user()

6–15 (same structure)

6. OnlineClass, OfflineClass → class_mode()
7. DebitCard, CreditCard → payment_limit()
8. Dog, Cat → make_sound()
9. Teacher, Student → get_role()
10. Printer, Scanner → device_function()
11. LinuxOS, WindowsOS → shutdown()
12. FileLogger, DBLogger → log_data()
13. Bus, Bike → fuel_type()
14. Admin, User → access_level()
15. FreePlan, PremiumPlan → subscription_details()

🧬 MRO – 15 SCENARIO QUESTIONS

1. Company Role Hierarchy

Create classes:

- Employee
- Developer(Employee)
- Tester(Employee)
- TeamLead(Developer, Tester)

Print MRO and explain which method executes first.

---

2. Login System Inheritance

Classes:

- Login
- AdminLogin(Login)
- UserLogin(Login)
- Portal(AdminLogin, UserLogin)

Demonstrate MRO using help().

---

3. Vehicle Engine System

Classes:

- Engine
- PetrolEngine(Engine)
- ElectricEngine(Engine)
- HybridCar(PetrolEngine, ElectricEngine)

4–15 (same style)

4. A → B → C → D(B, C)
5. Animal → Mammal → Bird → Bat(Mammal, Bird)
6. School → College → University(College, School)
7. UI → Backend → FullStack(UI, Backend)
8. Base1, Base2, Child(Base1, Base2)
9. Cloud → AWS → Azure → DevOps(AWS, Azure)
10. Logger → FileLogger → ConsoleLogger → AppLogger
11. OS → Windows → Linux → DualBoot
12. Account → Savings → Current → Bank(Savings, Current)
13. Shape → Polygon → Square → Figure
14. Auth → OAuth → JWT → AppAuth

15. Device → Mobile → Laptop → SmartDevice

🧱 ABSTRACTION – 15 SCENARIO QUESTIONS

1. Vehicle System

Create abstract class Vehicle.

- Abstract method: start_engine()
- Implement in class Car

---

2. Bank Interest System

Abstract class: Bank

- Abstract method: calculate_interest()
- Implement in SBI and HDFC

---

3. Payment Gateway

Abstract class: Payment

- Abstract method: make_payment()
- Implement in UPI and Card

4–15 (same pattern)

4. Shape → calculate_area()
5. Employee → calculate_salary()
6. Database → connect()
7. Notification → send()
8. Appliance → power_usage()
9. Account → withdraw()
10. Exam → evaluate()
11. Food → prepare()
12. Game → play()

13. OS → boot_system()

14. Remote → operate()

15. VehicleRental → rent_cost()