C-DAC Mumbai
OOPJ Lab Assignment

1. Bank Account Protection

Scenario: You are creating a simple banking system. A user should not be able to directly change their balance.

Problem Statement:
Create a class BankAccount with a private variable balance. Provide deposit() and withdraw() methods to change balance. Provide a getter to view balance. Validate that withdrawal cannot exceed balance.

Class/Fields:
- private double balance

Methods:
- deposit(double amount) → add to balance
- withdraw(double amount) → subtract from balance if sufficient
- getBalance() → returns current balance

Sample Input:
Deposit = 5000
Withdraw = 2000

Sample Output:
Updated Balance = 3000

---

2. Student Marks

Scenario: A teacher wants to update student marks but must ensure marks are valid.

Problem Statement:
Create a class Student with private marks. Create getter and setter with validation (0–100).

Fields:
- private int marks

Methods:
- setMarks(int marks) → validate and set
- getMarks() → return marks

Sample Input:
marks = 85

Sample Output:
Marks = 85

---

3. Employee Age Validation

Scenario: HR wants to ensure employees entered in the system are adults.

Problem Statement:
Create a class Employee with private age. Provide getter/setter. Setter should reject age < 18.
Fields:

- private int age

Methods:

- setAge(int age) → if age >=18, set; else show error
- getAge() → returns age

Sample Input:
age = 17

Sample Output:
"Invalid age"

---

4. Library Book Availability

Scenario: A library wants to keep track of available copies without letting external code change it directly.

Problem Statement:
Create a class Book with private copiesAvailable. Provide getter/setter to update or read copies.

Fields:

- private int copiesAvailable

Methods:

- addCopies(int n) → add copies
- removeCopies(int n) → subtract if enough copies
- getCopiesAvailable() → return current copies

Sample Input:
add 3 copies, remove 1 copy
Sample Output:
Copies available = 2

---

5. Temperature Sensor

Scenario: Sensor should only accept temperatures in safe range.

Problem Statement:
Create a class TemperatureSensor with private temperature. Setter validates range 0–100°C.

Fields:
- private int temperature

Methods:
- setTemperature(int t) → only 0–100 valid
- getTemperature() → return temperature

Sample Input:
temperature = -5
Sample Output:
"Temperature out of range"

---

6. Shape Area Calculation

Scenario: You are designing a program to calculate areas of different shapes.

Problem Statement:
Create a superclass Shape with a method area(). Derive two subclasses Rectangle and Circle. Implement area() in each subclass.

Classes/Fields:
- Shape → area() (method stub)
- Rectangle → length, breadth
- Circle → radius

Methods:
- Rectangle.area() → length × breadth
- Circle.area() → π × radius²

Sample Input:
Rectangle → length=5, breadth=10
Circle → radius=7

Sample Output:
Rectangle Area = 50
Circle Area = 153.86

---

7. Employee Hierarchy

Scenario: A company has regular and contractual employees with different pay structures.

Problem Statement:
Create a superclass Employee with name and basicSalary. Subclass RegularEmployee adds HRA 10%, Subclass ContractEmployee adds allowance 5%. Display net salary.

Classes/Fields:
- Employee → name, basicSalary
- RegularEmployee → HRA 10%
- ContractEmployee → allowance 5%

Sample Input:
Regular → name=Rahul, basicSalary=20000
Contract → name=Riya, basicSalary=15000

Sample Output:
Rahul Net Salary = 22000
Riya Net Salary = 15750

---

8. Vehicle Types

Scenario: You want to categorize vehicles.

Problem Statement:
Create a superclass Vehicle with brand, speed. Create subclasses Car and Bike with additional modelType. Display details of each vehicle.

Classes/Fields:
- Vehicle → brand, speed
- Car → modelType
- Bike → modelType

Sample Input:
Car → brand=Honda, speed=180, modelType=Civic
Bike → brand=Yamaha, speed=120, modelType=R15

Sample Output:
Car → Honda Civic, Speed=180
Bike → Yamaha R15, Speed=120

---

9. Animal Sound

Scenario: You are building a zoo management system to play animal sounds.

Problem Statement:
Create a superclass Animal with method makeSound(). Subclass Dog and Cat override makeSound().

Classes/Fields:

- Animal → makeSound()
- Dog → "Bark"
- Cat → "Meow"

Sample Output:
Dog → Bark
Cat → Meow

---

10. Academic Staff

Scenario: University has teaching and non-teaching staff.

Problem Statement:
Superclass Staff with name, salary. Subclass TeachingStaff adds subject, Subclass NonTeachingStaff adds department. Display staff info.

Classes/Fields:

- Staff → name, salary
- TeachingStaff → subject
- NonTeachingStaff → department

Sample Input:
Teaching → name=Anita, salary=50000, subject=Math
NonTeaching → name=Ramesh, salary=40000, department=Admin

Sample Output:
Anita → Math, 50000
Ramesh → Admin, 40000

---

11. Bank Account Types

Scenario: Bank provides different account types.
Problem Statement:
Superclass Account → accountNo, balance. Subclass SavingAccount → interestRate. Subclass CurrentAccount → overdraftLimit. Display account details.

Classes/Fields:

- Account → accountNo, balance
- SavingAccount → interestRate
- CurrentAccount → overdraftLimit

Sample Input:
Saving → accountNo=101, balance=5000, interestRate=5%
Current → accountNo=102, balance=10000, overdraftLimit=2000

Sample Output:
Saving → 101, Balance=5000, Interest=5%
Current → 102, Balance=10000, Overdraft=2000

---

12. Payment System

Scenario: A company accepts different payment modes.
Problem Statement:
Create an abstract class Payment with abstract method pay(). Create subclasses CreditCardPayment and UPIPayment that implement pay().

Classes/Fields:

- Payment → pay() (abstract)
- CreditCardPayment → cardNumber, amount
- UPIPayment → upiId, amount

Sample Input:
Credit Card → cardNumber=1234567890123456, amount=5000
UPI → upiId=rahul@upi, amount=2000

Sample Output:
Payment via Credit Card 1234567890123456 → Rs. 5000 Paid
Payment via UPI rahul@upi → Rs. 2000 Paid

---

13. Shape Drawing

Scenario: A graphics program needs to draw different shapes.

Problem Statement:
Create an abstract class Shape with abstract method draw(). Subclass Circle and Rectangle implement draw().

Classes/Fields:

- Shape → draw() (abstract)
- Circle → radius
- Rectangle → length, breadth

Sample Input:
Circle → radius=7
Rectangle → length=5, breadth=10

Sample Output:
Drawing Circle of radius 7
Drawing Rectangle of length 5 and breadth 10

---

14. Employee Bonus Calculation

Scenario: A company has different types of employees with specific bonus calculation rules.

Problem Statement:
Create an abstract class Employee with abstract method calculateBonus(). Subclass Manager → bonus=20% of salary, Subclass Developer → bonus=10% of salary.

Classes/Fields:

- Employee → name, salary, calculateBonus() (abstract)
- Manager → bonus=20% of salary
- Developer → bonus=10% of salary

Sample Input:
Manager → name=Anita, salary=50000
Developer → name=Rohit, salary=40000

Sample Output:
Anita Bonus = 10000
Rohit Bonus = 4000

---

15. Shape Area Calculation

Scenario: A program needs to calculate the area of different shapes using the same method name but different parameters.

Problem Statement:
Create a class ShapeArea with overloaded methods calculateArea().

Methods:
- calculateArea(int side) → calculates area of square
- calculateArea(int length, int breadth) → calculates area of rectangle
- calculateArea(double radius) → calculates area of circle

Sample Input:
Square → side=5
Rectangle → length=4, breadth=6
Circle → radius=3

Sample Output:
Square Area = 25
Rectangle Area = 24
Circle Area = 28.26

---

16. Employee Salary Display

Scenario: Company wants to display employee salary with different bonus calculations based on employee type.

Problem Statement:
Create class Employee with method displaySalary(). Subclass Manager and Developer override displaySalary() to include bonus.

Classes/Fields:
- Employee → name, salary, displaySalary() prints salary
- Manager → overrides displaySalary() → adds 20% bonus
- Developer → overrides displaySalary() → adds 10% bonus

Sample Input:
Manager → name=Anita, salary=50000
Developer → name=Rohit, salary=40000

Sample Output:
Anita Total Salary = 60000
Rohit Total Salary = 44000

---

17. Vehicle Speed Display

Scenario: Vehicle management system needs to display speed differently for different vehicle types.

Problem Statement:
Create class Vehicle with method displaySpeed(). Subclass Car and Bike override it.

Classes/Fields:
- Vehicle → displaySpeed() prints "Vehicle speed unknown"
- Car → overrides displaySpeed() → "Car speed 120 km/h"
- Bike → overrides displaySpeed() → "Bike speed 80 km/h"

Sample Input:
Car
Bike

Sample Output:
Car speed 120 km/h
Bike speed 80 km/h

---

18. Payment Process

Scenario: Company wants to process payments differently depending on mode of payment, but handle all payments through a single reference.

Problem Statement:
Create abstract class Payment with abstract method pay(). Subclass CreditCardPayment and UPIPayment implement pay().

Usage:
- Use Payment p reference → p = new CreditCardPayment(...) or p = new UPIPayment(...)
- Call p.pay() for runtime polymorphic behavior

Sample Input:
Credit Card → cardNumber=1234567890123456, amount=5000
UPI → upiId=rahul@upi, amount=2000

Sample Output:
Payment via Credit Card 1234567890123456 → Rs. 5000 Paid
Payment via UPI rahul@upi → Rs. 2000 Paid

---

19. Bank Account Types

Scenario: Bank manages different types of accounts: Savings and Current. Both share basic account details, but Savings accounts have interest and Current accounts have overdraft limit.

Problem Statement:
Create a superclass BankAccount with:

- Fields: accountNumber, accountHolder, balance
- Method: displayBalance()

Create subclasses:

- SavingsAccount → field: interestRate, method: calculateInterest()
- CurrentAccount → field: overdraftLimit, method: checkOverdraft()

Sample Input:
SavingsAccount → accountNumber=101, accountHolder=Ramesh, balance=5000, interestRate=5%
CurrentAccount → accountNumber=102, accountHolder=Anita, balance=2000, overdraftLimit=1000

Sample Output:
Ramesh → Balance=5000, Interest=250
Anita → Balance=2000, Overdraft Limit=1000

---

20. College Staff Hierarchy

Scenario: A college has employees who can be Teaching or Non-Teaching. Teaching staff can be Professors or Lecturers.

Problem Statement:
Create classes:

- Employee → name, salary, displaySalary()
- TeachingStaff → subject, overrides displaySalary()
- Professor → specialization, overrides displaySalary()
- Lecturer → department, overrides displaySalary()

Sample Input:
Professor → name=Dr. Sharma, salary=80000, subject=Math, specialization=Algebra
Lecturer → name=Ms. Mehta, salary=50000, subject=Physics, department=Science

Sample Output:
Dr. Sharma → Subject=Math, Specialization=Algebra, Salary=80000
Ms. Mehta → Subject=Physics, Department=Science, Salary=50000

---

21. Hospital Staff

Scenario: Hospital has Staff members. Both Doctors and Nurses are Staff.

Problem Statement:

- Staff → name, staffId, displayDetails()
- Doctor → specialization, displayDetails() override
- Nurse → shift, displayDetails() override

Sample Input:
Doctor → name=Dr. Reddy, staffId=101, specialization=Cardiology
Nurse → name=Nisha, staffId=102, shift=Night

Sample Output:
Dr. Reddy → Staff ID=101, Specialization=Cardiology
Nisha → Staff ID=102, Shift=Night

---

22. Vehicle Types

Scenario: Vehicles can be Land or Water types. Some vehicles can operate on both.

Problem Statement:

- Interface LandVehicle → method driveOnLand()
- Interface WaterVehicle → method driveOnWater()
- Class AmphibiousVehicle implements both interfaces → provides both methods

Sample Input:
AmphibiousVehicle → name=HydroCar

Sample Output:
HydroCar → Driving on Land
HydroCar → Driving on Water

---

23. School Members

Scenario: School has members: Teachers, Students, and Staff. All share common info.

Problem Statement:

- Member → name, id, displayInfo()
- Teacher → subject, overrides displayInfo()
- Student → grade, overrides displayInfo()
- Staff → department, overrides displayInfo()

Sample Input:
Teacher → name=Mr. Kumar, id=101, subject=English
Student → name=Riya, id=201, grade=10
Staff → name=Mr. Das, id=301, department=Maintenance

Sample Output:
Mr. Kumar → ID=101, Subject=English
Riya → ID=201, Grade=10
Mr. Das → ID=301, Department=Maintenance

---

24. Payment Gateway

Scenario: An e-commerce platform supports multiple payment methods like CreditCard and PayPal. All payments must implement a pay() method.

Problem Statement:

- Create an interface Payment → method pay(double amount)
- Classes CreditCardPayment and PayPalPayment implement Payment → provide their own pay() implementation
- In main(), take payment amount and process payment using both methods

Sample Input:
CreditCardPayment → amount=2500
PayPalPayment → amount=1500

Sample Output:
Processing Credit Card Payment of 2500
Processing PayPal Payment of 1500

---

25. Media Player

Scenario: A media player can play both Audio and Video files.

Problem Statement:

- Interface AudioPlayer → method playAudio(String song)
- Interface VideoPlayer → method playVideo(String movie)
- Class MediaPlayer implements both → provides implementation for both methods

Sample Input:
Audio → song="Shape of You"
Video → movie="Inception"

Sample Output:
Playing Audio: Shape of You
Playing Video: Inception

---

26. Smart Devices

Scenario: Smart devices can perform actions like calling, messaging, and browsing internet.

Problem Statement:

- Interface Callable → method makeCall(String number)
- Interface Messaging → method sendMessage(String number, String message)
- Interface Internet → method browse(String website)
- Class SmartPhone implements all three interfaces → provide respective implementations

Sample Input:
Call → number="9876543210"
Message → number="9876543210", message="Hello!"
Browse → website="www.google.com"

Sample Output:
Calling 9876543210
Sending message to 9876543210: Hello!
Browsing website: www.google.com

---

27. Shape Area Calculator

Scenario: A drawing application needs to calculate area for different shapes: Circle, Rectangle, and Square.

Problem Statement:

- Interface Shape → method calculateArea()
- Classes Circle, Rectangle, Square implement Shape → provide specific area calculation
- In main(), create objects of each shape, input dimensions, display calculated area

Sample Input:
Circle → radius=5
Rectangle → length=10, breadth=5
Square → side=4

Sample Output:
Circle Area = 78.5
Rectangle Area = 50
Square Area = 16

28. Online Shopping Cart System

Scenario: Build a simplified shopping cart system where users can add products, calculate total cost, and apply discounts.

Problem Statement:

- **Class Product** → instance variables: productId, name, price (Encapsulation: use private variables with getters/setters)
- **Abstract Class CartItem** → method calculateTotalPrice() (Abstract Class: define generic behavior for cart items)
- **Class Cart extends CartItem** → store list of products, implement calculateTotalPrice()
- **Interface Discountable** → method applyDiscount(double percentage) (Interface: any item can have discounts applied)
  In main(), create a cart, add 3 products, apply 10% discount to one product, display total cost

Sample Input:
Product1 → name="Laptop", price=50000
Product2 → name="Mouse", price=500
Product3 → name="Keyboard", price=1200

Sample Output:
Applying 10% discount to Laptop
Total Cart Price = 51800

29. Employee Management System

Scenario: Manage employee details, calculate salaries, and differentiate employee types.

Problem Statement:

- **Abstract Class Employee** → instance variables: name, id
  - Abstract method calculateSalary() → different calculation for each type
- **Class PermanentEmployee extends Employee** → include basicSalary and hra → implement calculateSalary()
- **Class ContractEmployee extends Employee** → include hourlyRate and hoursWorked → implement calculateSalary()
- **Interface BonusEligible** → method calculateBonus() → applies only to permanent employees
- In main(), create 2 permanent and 2 contract employees, display salary + bonus if eligible

Sample Input:
PermanentEmployee → name="Amit", basicSalary=50000, hra=5000
ContractEmployee → name="Neha", hourlyRate=300, hoursWorked=100

Sample Output:
Amit Salary = 55000, Bonus = 5500
Neha Salary = 30000

---

30. Library Management System

Scenario: Manage books and library members with borrowing functionality.

Problem Statement:

- **Class Book** → private variables: bookId, title, author (Encapsulation)
- **Abstract Class LibraryMember** → instance variables: memberId, name
  - Abstract method borrowBook(Book book)
- **Class StudentMember extends LibraryMember** → limit 3 books
- **Class FacultyMember extends LibraryMember** → limit 5 books
- **Interface Notifyable** → method sendNotification(String message) → notify members about overdue books
- In main(), create 1 student and 1 faculty, borrow books, send notifications

Sample Input:
Student → borrow 2 books
Faculty → borrow 4 books

Sample Output:
StudentMember Amit borrowed 2 books
FacultyMember Prof. Singh borrowed 4 books
Notification sent to Amit: Return books within 7 days
Notification sent to Prof. Singh: Return books within 14 days