**Advanced Java Lab Manual for Object-Oriented Programming**

**JAVA LAB EXPERIMENTS**

**Experiments**

1. **Passing Arguments to a Method & Returning Values**
2. Scenario 1: Insurance Premium Calculation System
3. Scenario 2:AI-Based Health Diagnosis System
4. **Method Overloading & Constructor Overloading**

a) Scenario 1: Multi-Level User Authentication System

b) Scenario 2: Stock Market Prediction System

1. **Passing Arrays & Objects as Arguments, Returning Objects**
2. Scenario 1: AI-Based Resume Screening System
3. Scenario 2: AI-Based Resume Ranking & Job Matching System
4. **Aggregation & Composition**

a) Scenario 1: Smart City Traffic Management System

b) Scenario 2 : Application Scenario: IoT-Based Smart Home System

5. **Inheritance & Overriding Superclass Methods**

1. Scenario 1: Application Scenario: Enterprise Role-Based Access Control (RBAC) System
2. Scenario 2: Multi-Factor Authentication System (MFA)

**6.Abstract Base Classes & Abstract Methods**

a) Scenario 1: Digital Wallet & Payment Processing System

b) Scenario 2: AI-Based Credit Card Fraud Detection System

7. **File Handling & Object Serialization**

a) Scenario 1: Cryptocurrency Transaction Ledger

b) Scenario 2: Digital Certificate Management System

8. **Exception Handling**

a)Scenario 1: Cybersecurity Intrusion Detection System

b) Scenario 2: Autonomous Vehicle Error Handling System

**Experiment 1: Passing Arguments to a Method & Returning Values**

**Objective:**

Develop a Java program that processes complex data using method arguments and return values.

1. **Application Scenario: Insurance Premium Calculation System**

* An insurance company calculates the premium for customers based on factors like **age, medical history, vehicle type, and coverage plan**.
* The program should take these inputs, apply different calculations, and return the final premium amount.

**Task to be Performed:**

✅ Implement an InsuranceCalculator class with a method calculatePremium(int age, String vehicleType, boolean hasMedicalHistory, String coveragePlan).  
✅ Use conditional logic and formula-based calculations to determine the premium.  
✅ Return the final premium amount and display detailed computation steps.

**Expected Outcome:**

* A complex decision-based premium calculation system.
* Demonstrates effective parameter passing and returning computed values.

.

**b) Application Scenario : AI-Based Health Diagnosis System**

* A hospital uses an AI system to analyze **symptoms and medical history** to predict diseases.
* The system takes symptoms, medical records, and risk factors as **inputs** and returns a **probability score of potential diseases**.

**Task to be Performed:**

✅ Implement a HealthDiagnosis class with analyzeSymptoms(String[] symptoms, int age, boolean hasChronicDisease).  
✅ Process the input data and return **a probability score for disease prediction**.  
✅ Display the **computed health risk** for patients.

**Expected Outcome:**

* A **real-world AI-driven healthcare system** that evaluates medical conditions.
* Demonstrates **advanced data passing and return handling in Java methods**.

**Experiment 2: Method Overloading & Constructor Overloading**

**Objective:**

Demonstrate method and constructor overloading using **complex user authentication and access control mechanisms**.

1. **Application Scenario: Multi-Level User Authentication System**

* A system where **users can log in using different credentials**:
  1. Username & Password
  2. Email & OTP
  3. Biometric Authentication

**Task to be Performed:**

✅ Implement an AuthenticationSystem class with **overloaded methods** for login authentication.  
✅ Overload constructors to initialize users with different authentication modes.  
✅ Display authentication success or failure based on input credentials.

**Expected Outcome:**

* A dynamic user authentication system with multiple authentication strategies.
* Demonstrates real-world security handling using method and constructor overloading.

1. **Application Scenario: Stock Market Prediction System**

* A stock market platform **calculates risk and return on investments** based on different parameters.

**Task to be Performed:**

✅ Implement a StockAnalysis class with **multiple constructors** to initialize investments based on stock type.  
✅ Overload a method predictReturns() to calculate stock returns for:

* **Short-term investment (1 year)**
* **Mid-term investment (3 years)**
* **Long-term investment (5+ years)**  
  ✅ Display **predicted stock performance based on user inputs**.

**Expected Outcome:**

* A **data-driven stock forecasting system** using method and constructor overloading.
* Demonstrates **real-world financial prediction in Java applications**.

**Experiment 3: Passing Arrays & Objects as Arguments, Returning Objects**

**Objective:**

Develop a Java program to **process multiple records and return computed objects**.

**a)Application Scenario: AI-Based Resume Screening System**

* A **HR system processes multiple job applications**, evaluating them based on skills, experience, and job role suitability.

**Task to be Performed:**

✅ Implement a Candidate class with attributes like skills, experience, and appliedRole.  
✅ Pass an **array of candidates** to a method that processes and ranks them.  
✅ Return an **updated candidate object** with scores and suitability status.

**Expected Outcome:**

* A **real-world AI-powered recruitment processing system** using Java objects and arrays.
* Demonstrates handling of **large datasets in method processing**.

**b)Application Scenario: AI-Based Resume Ranking & Job Matching System**

* A **job portal** filters and ranks candidates for **multiple job openings** based on skills and experience.

**Task to be Performed:**

✅ Implement a JobMatcher class with a method matchCandidates(Candidate[] candidates, String jobRole).  
✅ Process multiple candidate objects and return a **ranked list** based on relevance.  
✅ Display **top candidates** for the job.

**Expected Outcome:**

* A **real-world AI-driven recruitment system** for job portals.
* Demonstrates **handling of large datasets with object passing and returning**.

**Experiment 4: Aggregation & Composition**

**Objective:**

Demonstrate aggregation and composition with **multiple levels of object dependencies**.

**a)Application Scenario: Smart City Traffic Management System**

* A **Traffic Control System** contains multiple **junctions**, and each junction has multiple **signal controllers**.

**Task to be Performed:**

✅ Implement classes:

* City → contains multiple Junctions.
* Junction → has multiple TrafficSignals (composition).
* TrafficSignal → operates independently (aggregation).  
  ✅ Display **real-time traffic signal data** at various junctions.

**Expected Outcome:**

* A **multi-level aggregation and composition-based system**.
* Demonstrates **hierarchical object relationships in urban infrastructure**.

**b)Application Scenario: IoT-Based Smart Home System**

* A **smart home ecosystem** consists of **multiple rooms**, each containing **various smart devices**.

**Task to be Performed:**

✅ Implement classes:

* SmartHome → has multiple Rooms (aggregation).
* Room → contains multiple SmartDevices (composition).
* SmartDevice → represents different IoT-enabled devices.  
  ✅ Display **device status and control automation**.

**Expected Outcome:**

* A **multi-layered smart home automation system**.
* Demonstrates **aggregation and composition for IoT applications**.

**Experiment 5: Inheritance & Overriding Superclass Methods**

**Objective:**

Use inheritance and method overriding to build a **role-based access control system**.

**a)Application Scenario: Enterprise Role-Based Access Control (RBAC) System**

* A **corporate IT system** assigns different **access levels** to employees (Admin, Manager, Employee).

**Task to be Performed:**

✅ Implement a superclass User with method accessSystem().  
✅ Subclasses Admin, Manager, and Employee override this method with different access permissions.  
✅ Demonstrate **hierarchical security management**.

**Expected Outcome:**

* A **multi-tiered access management system** with real-time authorization control.
* Demonstrates **polymorphism and method overriding in a security-based application**.

**b) Application Scenario: Multi-Factor Authentication System (MFA)**

* A **user authentication system** supports **basic login**, **OTP verification**, and **biometric authentication**.

**Task to be Performed:**

✅ Implement a superclass UserAuthentication with a method authenticate().  
✅ Subclasses PasswordLogin, OTPLogin, and BiometricLogin override authenticate().  
✅ Implement **multi-step authentication and access control**.

**Expected Outcome:**

* A **real-world MFA security system**.
* Demonstrates **hierarchical authentication with method overriding**.

**Experiment 6: Abstract Base Classes & Abstract Methods**

**Objective:**

Demonstrate abstraction in a **complex financial transaction system**.

**a)Application Scenario: Digital Wallet & Payment Processing System**

* A digital payment platform (PayTM, Google Pay) handles different **payment types**: UPI, Credit Card, and Net Banking.

**Task to be Performed:**

✅ Implement an abstract class PaymentGateway with an abstract method processPayment().  
✅ Subclasses UPIPayment, CreditCardPayment, NetBankingPayment provide their own implementation.  
✅ Demonstrate a transaction with different payment methods.

**Expected Outcome:**

* A **modular, payment-processing system** using **abstraction**.
* Demonstrates **real-world financial transactions with security features**.

**b) Application Scenario: AI-Based Credit Card Fraud Detection System**

* A bank uses **AI models** to analyze transactions and detect fraudulent activities.

**Task to be Performed:**

✅ Implement an abstract class FraudDetection with an abstract method analyzeTransaction().  
✅ Subclasses CreditCardFraud and BankTransferFraud implement transaction analysis logic.  
✅ Simulate **fraud detection with probability scores**.

**Expected Outcome:**

* A **real-world AI-driven fraud detection system** using abstraction.

Demonstrates **modular security mechanisms for financial applications**

**Experiment 7: File Handling & Object Serialization**

**Objective:**

Store and retrieve **large-scale customer transaction data** using file handling and serialization.

**a)Application Scenario: Cryptocurrency Transaction Ledger**

* A **crypto exchange** stores transaction history in files using serialization.

**Task to be Performed:**

✅ Implement a CryptoTransaction class with attributes (walletID, transactionAmount, timestamp).  
✅ Serialize transaction objects to a file.  
✅ Deserialize and display **stored transactions**.

**Expected Outcome:**

* A **real-world crypto transaction ledger system**.
* Demonstrates **persistence and data integrity using object serialization**.

**b) Application Scenario: Digital Certificate Management System**

* A government system **issues and stores encrypted digital certificates** (e.g., passports, identity cards).

**Task to be Performed:**

✅ Implement a DigitalCertificate class with attributes (holderName, certificateID, expiryDate).  
✅ Serialize and store **user certificates in a secure file**.  
✅ Deserialize and retrieve **user information** when needed.

**Expected Outcome:**

* A **real-world identity management system**.
* Demonstrates **secure object serialization and file handling**.

**Experiment 8: Exception Handling**

**Objective:**

Develop an exception handling mechanism for **a real-time, high-risk application**.

**a)Application Scenario: Cybersecurity Intrusion Detection System**

* A **security system** detects and handles unauthorized login attempts.

**Task to be Performed:**

✅ Implement an IntrusionDetection class with a method validateLogin().  
✅ Throw a **custom exception** (UnauthorizedAccessException) if invalid login attempts exceed a threshold.  
✅ Implement try-catch-finally blocks to handle exceptions and log security breaches.

**Expected Outcome:**

* A **real-world cybersecurity defense mechanism**.
* Demonstrates **exception handling in a high-risk security application**.

**b)Application Scenario: Autonomous Vehicle Error Handling System**

* A **self-driving car system** detects obstacles and handles errors in real-time.

**Task to be Performed:**

✅ Implement an AutonomousCar class with a method navigate().  
✅ Throw a SensorFailureException when a sensor malfunctions.  
✅ Implement try-catch-finally blocks for **real-time error handling**.

**Expected Outcome:**

* A **real-world self-driving car safety system**.
* Demonstrates **handling critical failures in real-time applications**.