```
import java.util.*;

// 1. Employee Management System
abstract class Employee {
private int employeeId;
private String name;
private double baseSalary;

public Employee(int employeeId, String name, double baseSalary) {
this.employeeId = employeeId;
this.name = name;
this.baseSalary = baseSalary;
}

public int getEmployeeId() { return employeeId; }
public String getName() { return name; }
public double getBaseSalary() { return baseSalary; }

public void setBaseSalary(double baseSalary) { this.baseSalary = baseSalary; }

public abstract double calculateSalary();

public void displayDetails() {
System.out.println("ID: " + employeeId + ", Name: " + name + ", Base Salary: " + baseSalary);
}
}

interface Department {
void assignDepartment(String dept);
String getDepartmentDetails();
}

class FullTimeEmployee extends Employee implements Department {
private String department;

public FullTimeEmployee(int id, String name, double salary) {
super(id, name, salary);
}

public double calculateSalary() { return getBaseSalary(); }

public void assignDepartment(String dept) { this.department = dept; }
```

```
public String getDepartmentDetails() { return "Department: " + department; }
}

class PartTimeEmployee extends Employee implements Department {
private int hoursWorked;
private double hourlyRate;
private String department;

public PartTimeEmployee(int id, String name, double rate, int hours) {
super(id, name, 0);
this.hourlyRate = rate;
this.hoursWorked = hours;
}

public double calculateSalary() { return hoursWorked * hourlyRate; }

public void assignDepartment(String dept) { this.department = dept; }
public String getDepartmentDetails() { return "Department: " + department; }
}

// 2. E-Commerce Platform
abstract class Product {
private int productId;
private String name;
private double price;

public Product(int productId, String name, double price) {
this.productId = productId;
this.name = name;
this.price = price;
}

public double getPrice() { return price; }
public String getName() { return name; }

public abstract double calculateDiscount();
}

interface Taxable {
double calculateTax();
String getTaxDetails();
}
```

```
class Electronics extends Product implements Taxable{
public Electronics(int id, String name, double price) {
super(id, name, price);
}

public double calculateDiscount() { return getPrice() * 0.1; }
public double calculateTax(){ return getPrice() * 0.18; }
public String getTaxDetails(){ return "18% GST"; }
}

class Clothing extends Product implements Taxable{
public Clothing(int id, String name, double price) {
super(id, name, price);
}

public double calculateDiscount() { return getPrice() * 0.2; }
public double calculateTax(){ return getPrice() * 0.05; }
public String getTaxDetails(){ return "5% VAT";}
}

class Groceries extends Product {
public Groceries(int id, String name, double price) {
super(id, name, price);
}
public double calculateDiscount() { return getPrice() * 0.05; }
}

// 3. Vehicle Rental System
abstract class Vehicle{
private String vehicleNumber;
private String type;
private double rentalRate;

public Vehicle(String vehicleNumber, String type, double rentalRate) {
this.vehicleNumber = vehicleNumber;
this.type = type;
this.rentalRate = rentalRate;
}

public double getRentalRate() { return rentalRate; }
public abstract double calculateRentalCost(int days);
```

```
}

interface Insurable {
double calculateInsurance();
String getInsuranceDetails();
}

class Car extends Vehicle implements Insurable {
public Car(String num, double rate) { super(num, "Car", rate); }

public double calculateRentalCost(int days) { return days * getRentalRate(); }
public double calculateInsurance() { return 500; }
public String getInsuranceDetails() { return "Car insurance: Flat 500"; }
}

class Bike extends Vehicle implements Insurable {
public Bike(String num, double rate) { super(num, "Bike", rate); }

public double calculateRentalCost(int days) { return days * getRentalRate(); }
public double calculateInsurance() { return 200; }
public String getInsuranceDetails() { return "Bike insurance: Flat 200"; }
}

class Truck extends Vehicle implements Insurable {
public Truck(String num, double rate) { super(num, "Truck", rate); }

public double calculateRentalCost(int days) { return days * getRentalRate(); }
public double calculateInsurance() { return 1000; }
public String getInsuranceDetails() { return "Truck insurance: Flat 1000"; }
}

// 4. Banking System
abstract class BankAccount {
private int accountNumber;
private String holderName;
private double balance;

public BankAccount(int accountNumber, String holderName, double balance) {
this.accountNumber = accountNumber;
this.holderName = holderName;
this.balance = balance;
}
```

```
public double getBalance() { return balance; }
public void deposit(double amount) { balance += amount; }
public void withdraw(double amount) { if (balance >= amount) balance -= amount; }

public abstract double calculateInterest();
}

interface Loanable {
void applyForLoan(double amount);
boolean calculateLoanEligibility();
}

class SavingsAccount extends BankAccount implements Loanable {
public SavingsAccount(int acc, String name, double bal) {
super(acc, name, bal);
}

public double calculateInterest() { return getBalance() * 0.04; }
public void applyForLoan(double amount) { System.out.println("Savings loan applied: " +
amount); }
public boolean calculateLoanEligibility() { return getBalance() > 5000; }
}

class CurrentAccount extends BankAccount implements Loanable {
public CurrentAccount(int acc, String name, double bal) {
super(acc, name, bal);
}

public double calculateInterest() { return getBalance() * 0.02; }
public void applyForLoan(double amount) { System.out.println("Current loan applied: " +
amount); }
public boolean calculateLoanEligibility() { return getBalance() > 10000; }
}

// 5. Library Management System
abstract class LibraryItem {
private int itemId;
private String title;
private String author;

public LibraryItem(int itemId, String title, String author) {
```

```
this.itemId = itemId;
this.title = title;
this.author = author;
}

public void getItemDetails() {
System.out.println("ID: " + itemId + ", Title: " + title + ", Author: " + author);
}

public abstract int getLoanDuration();
}

interface Reservable {
void reserveItem();
boolean checkAvailability();
}

class Book extends LibraryItem implements Reservable {
public Book(int id, String title, String author) { super(id, title, author); }

public int getLoanDuration() { return 14; }
public void reserveItem() { System.out.println("Book reserved."); }
public boolean checkAvailability() { return true; }
}

class Magazine extends LibraryItem implements Reservable {
public Magazine(int id, String title, String author) { super(id, title, author); }

public int getLoanDuration() { return 7; }
public void reserveItem() { System.out.println("Magazine reserved."); }
public boolean checkAvailability() { return true; }
}

class DVD extends LibraryItem implements Reservable {
public DVD(int id, String title, String author) { super(id, title, author); }

public int getLoanDuration() { return 3; }
public void reserveItem() { System.out.println("DVD reserved."); }
public boolean checkAvailability() { return false; }
}

// 6. Online Food Delivery System
```

```
abstract class FoodItem {
private String itemName;
private double price;
private int quantity;

public FoodItem(String itemName, double price, int quantity) {
this.itemName = itemName;
this.price = price;
this.quantity = quantity;
}

public double getPrice() { return price; }
public int getQuantity() { return quantity; }

public void getItemDetails() {
System.out.println(itemName + " Price: " + price + " Qty: " + quantity);
}

public abstract double calculateTotalPrice();
}

interface Discountable {
double applyDiscount();
String getDiscountDetails();
}

class VegItem extends FoodItem implements Discountable {
public VegItem(String name, double price, int qty) { super(name, price, qty); }

public double calculateTotalPrice(){ return getPrice() * getQuantity(); }
public double applyDiscount() { return calculateTotalPrice()* 0.1; }
public String getDiscountDetails() { return "10% discount on Veg"; }
}

class NonVegItem extends FoodItem implements Discountable {
public NonVegItem(String name, double price, int qty) { super(name, price, qty); }

public double calculateTotalPrice(){ return (getPrice() * getQuantity()) + 20; } // extra charge
public double applyDiscount() { return calculateTotalPrice()* 0.05; }
public String getDiscountDetails() { return "5% discount on Non-Veg";}
}
```

```
// 7. Hospital Patient Management
abstract class Patient {
private int patientId;
private String name;
private int age;

public Patient(int id, String name, int age) {
this.patientId = id;
this.name = name;
this.age = age;
}

public void getPatientDetails() {
System.out.println("ID: " + patientId + ", Name: " + name + ", Age: " + age);
}

public abstract double calculateBill();
}

interface MedicalRecord {
void addRecord(String record);
void viewRecords();
}

class InPatient extends Patient implements MedicalRecord {
private double dailyCharge;
private int days;

public InPatient(int id, String name, int age, double charge, int days) {
super(id, name, age);
this.dailyCharge = charge;
this.days = days;
}

public double calculateBill() { return dailyCharge * days; }

public void addRecord(String record) { System.out.println("Record added: " + record); }
public void viewRecords() { System.out.println("Viewing InPatient records..."); }
}

class OutPatient extends Patient implements MedicalRecord {
private double consultationFee;
```

```
public OutPatient(int id, String name, int age, double fee) {
super(id, name, age);
this.consultationFee = fee;
}

public double calculateBill() { return consultationFee; }

public void addRecord(String record) { System.out.println("Record added: " + record); }
public void viewRecords() { System.out.println("Viewing OutPatient records..."); }
}

// 8. Ride-Hailing Application
abstract class RideVehicle {
private String vehicleId;
private String driverName;
private double ratePerKm;

public RideVehicle(String id, String driver, double rate) {
this.vehicleId = id;
this.driverName = driver;
this.ratePerKm = rate;
}

public double getRatePerKm() { return ratePerKm; }

public void getVehicleDetails() {
System.out.println("VehicleID: " + vehicleId + ", Driver: " + driverName + ", Rate/km: " +
ratePerKm);
}

public abstract double calculateFare(double distance);
}

interface GPS {
String getCurrentLocation();
void updateLocation(String location);
}

class CarRide extends RideVehicle implements GPS {
private String location;
```

```
public CarRide(String id, String driver, double rate) { super(id, driver, rate); }

public double calculateFare(double distance) { return distance * getRatePerKm(); }
public String getCurrentLocation() { return location; }
public void updateLocation(String location) { this.location = location; }
}

class BikeRide extends RideVehicle implements GPS {
private String location;

public BikeRide(String id, String driver, double rate) { super(id, driver, rate); }

public double calculateFare(double distance) { return distance * getRatePerKm(); }
public String getCurrentLocation() { return location; }
public void updateLocation(String location) { this.location = location; }
}

class AutoRide extends RideVehicle implements GPS {
private String location;

public AutoRide(String id, String driver, double rate) { super(id, driver, rate); }

public double calculateFare(double distance) { return distance * getRatePerKm(); }
public String getCurrentLocation() { return location; }
public void updateLocation(String location) { this.location = location; }
}

// ------------------ Main Runner ------------------
public class Main {
public static void main(String[] args) {
// Example Polymorphism
Employee e1 = new FullTimeEmployee(1, "John", 50000);
Employee e2 = new PartTimeEmployee(2, "Mike", 500, 40);
e1.displayDetails();
System.out.println("Salary: " + e1.calculateSalary());
e2.displayDetails();
System.out.println("Salary: " + e2.calculateSalary());

Product p1 = new Electronics(101, "Laptop", 60000);
Product p2 = new Clothing(102, "Shirt", 2000);
Product p3 = new Groceries(103, "Rice", 1000);
```

```
System.out.println(p1.getName() + " Final Price: " + (p1.getPrice() +
((Taxable)p1).calculateTax()- p1.calculateDiscount()));
System.out.println(p2.getName() + " Final Price: " + (p2.getPrice() +
((Taxable)p2).calculateTax()- p2.calculateDiscount()));
System.out.println(p3.getName() + " Final Price: " + (p3.getPrice() - p3.calculateDiscount()));

Vehicle v1 = new Car("CAR123", 2000);
System.out.println("Car Rental (5 days): " + v1.calculateRentalCost(5));
System.out.println(((Insurable)v1).getInsuranceDetails());
}
}
```