Program:1

```
class Member {
    String name;
    int age;
    String phoneNumber;
    String address;
    double salary;
    public Member(String name, int age, String phoneNumber, String address, double salary) {
        this.name = name;
        this.age = age;
        this.phoneNumber = phoneNumber;
        this.address = address;
        this.salary = salary;
    }
    public void printSalary() {
        System.out.println("Salary: $" + salary);
    }
}
class Employee extends Member {
    String specialization;
    public Employee(String name, int age, String phoneNumber, String address, double salary,
String specialization) {
        super(name, age, phoneNumber, address, salary);
        this.specialization = specialization;
    }
}
class Manager extends Member {
    String department;
    public Manager(String name, int age, String phoneNumber, String address, double salary,
String department) {
        super(name, age, phoneNumber, address, salary);
        this.department = department;
    }
}
public class Main {
    public static void main(String[] args) {
        Employee employee = new Employee("John Doe", 30, "1234567890", "123 Main St",
50000.0, "Software Developer");
        Manager manager = new Manager("David", 35, "9876543210", "456 Oak St", 70000.0,
"Human Resources");
        System.out.println("Employee Details:");
        System.out.println("Name: " + employee.name);
        System.out.println("Age: " + employee.age);
        System.out.println("Phone Number: " + employee.phoneNumber);
        System.out.println("Address: " + employee.address);
        employee.printSalary();
```

```
        System.out.println("Specialization: " + employee.specialization);
        System.out.println();

        System.out.println("Manager Details:");
        System.out.println("Name: " + manager.name);
        System.out.println("Age: " + manager.age);
        System.out.println("Phone Number: " + manager.phoneNumber);
        System.out.println("Address: " + manager.address);
        manager.printSalary();
        System.out.println("Department: " + manager.department);
    }
}
```

Program:2

```
interface AccountOperations {
    void deposit(double amount);
    void withdraw(double amount);
    double checkBalance();
}
abstract class BankAccount implements AccountOperations {
    private String accountNumber;
    private double balance;
    public BankAccount(String accountNumber, double initialBalance) {
        this.accountNumber = accountNumber;
        this.balance = initialBalance;
    }
    public String getAccountNumber() {
        return accountNumber;
    }
    public double getBalance() {
        return balance;
    }
    @Override
    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: $" + amount);
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }
    @Override
    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: $" + amount);
        } else {
```

```
            System.out.println("Invalid withdrawal amount or insufficient funds.");
        }
    }
    @Override
    public double checkBalance() {
        System.out.println("Account Balance: $" + balance);
        return balance;
    }
}
class SavingsAccount extends BankAccount {
    private double interestRate;
    public SavingsAccount(String accountNumber, double initialBalance, double interestRate)
{
        super(accountNumber, initialBalance);
        this.interestRate = interestRate;
    }
    public void applyInterest() {
        double interestAmount = getBalance() * interestRate / 100;
        deposit(interestAmount);
        System.out.println("Interest applied: $" + interestAmount);
    }
}
class CheckingAccount extends BankAccount {
    private double overdraftLimit;
    public CheckingAccount(String accountNumber, double initialBalance, double
overdraftLimit) {
        super(accountNumber, initialBalance);
        this.overdraftLimit = overdraftLimit;
    }

    @Override
    public void withdraw(double amount) {
        if (amount > 0 && amount <= getBalance() + overdraftLimit) {
            super.withdraw(amount);
        } else {
            System.out.println("Invalid withdrawal amount or exceeding overdraft limit.");
        }
    }
}
class LoanAccount extends BankAccount {
    private double interestRate;
    public LoanAccount(String accountNumber, double initialBalance, double interestRate) {
        super(accountNumber, initialBalance);
        this.interestRate = interestRate;
    }
    public void applyInterest() {
```

```
        double interestAmount = getBalance() * interestRate / 100;
        withdraw(interestAmount);
        System.out.println("Interest applied: $" + interestAmount);
    }
}
public class Main {
    public static void main(String[] args) {
        SavingsAccount savingsAccount = new SavingsAccount("SA123", 1000.0, 2.5);
        savingsAccount.deposit(500.0);
        savingsAccount.applyInterest();
        savingsAccount.checkBalance();

        CheckingAccount checkingAccount = new CheckingAccount("CA456", 2000.0,
1000.0);
        checkingAccount.withdraw(1500.0);
        checkingAccount.checkBalance();

        LoanAccount loanAccount = new LoanAccount("LA789", 5000.0, 5.0);
        loanAccount.applyInterest();
        loanAccount.checkBalance();
    }
}
```

Program:3

```
import java.util.ArrayList;
class Person {
    String name;
    int age;

    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
class Student extends Person {
    Student(String name, int age) {
        super(name, age);
    }
}
class Professor extends Person {
    Professor(String name, int age) {
        super(name, age);
    }
}
class Course {
    String courseName;
    ArrayList<String> prerequisites;
```

```
    ArrayList<Student> enrolledStudents;

    Course(String courseName, ArrayList<String> prerequisites) {
        this.courseName = courseName;
        this.prerequisites = prerequisites;
        this.enrolledStudents = new ArrayList<>();
    }

    void enrollStudent(Student student) {
        if (hasCompletedPrerequisites(student)) {
            enrolledStudents.add(student);
            System.out.println(student.name + " enrolled in " + courseName);
        } else {
            System.out.println(student.name + " cannot be enrolled in " + courseName +
                    " due to incomplete prerequisites.");
        }
    }

    private boolean hasCompletedPrerequisites(Student student) {
        return true;
    }

    void displayEnrolledStudents() {
        System.out.println("Enrolled students in " + courseName + ":");
        for (Student student : enrolledStudents) {
            System.out.println("Name: " + student.name + ", Age: " + student.age);
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Student student1 = new Student("Alice", 20);
        Student student2 = new Student("Bob", 22);

        Course programmingCourse = new Course("Programming 101", new ArrayList<>());
        programmingCourse.enrollStudent(student1);
        programmingCourse.enrollStudent(student2);

        programmingCourse.displayEnrolledStudents();
    }
}
```