// Topic 1: toString() and getClass()

class Employee {

    int id;

    String name;

    double salary;

    Employee(int i, String n, double s) {

        id = i;

        name = n;

        salary = s;

    }

    @Override

    public String toString() {

        return id + " " + name + " " + salary;

    }

}

public class Main1 {

    public static void main(String[] args) {

        Employee e1 = new Employee(1, "Alok", 50000);

        Employee e2 = new Employee(2, "Singh", 60000);

        System.out.println(e1);

        System.out.println(e1.getClass().getName());

        System.out.println(e2);

        System.out.println(e2.getClass().getName());

    }}

// Topic 2: equals() vs ==

class Product {

    int productId;

    String productName;

    Product(int id, String name) {

        productId = id;

        productName = name;

    }

    @Override

    public boolean equals(Object o) {

        if (this == o)

            return true;

        if (!(o instanceof Product))

            return false;

        return productId == ((Product) o).productId;

    }

}

public class Main2 {

    public static void main(String[] args) {

        Product p1 = new Product(101, "Phone");

        Product p2 = new Product(101, "Phone");

        System.out.println(p1 == p2);

        System.out.println(p1.equals(p2));

    }}

// Topic 3: hashCode() and equals() Contract

import java.util.\*;

class Student {

    int rollNo;

    String name;

    Student(int r, String n) {

        rollNo = r;

        name = n;

    }

    public boolean equals(Object o) {

        if (this == o)

            return true;

        if (!(o instanceof Student))

            return false;

        return rollNo == ((Student) o).rollNo;

    }

    public int hashCode() {

        return Objects.hash(rollNo);

    }

}

public class Main3 {

    public static void main(String[] args) {

        HashSet<Student> set = new HashSet<>();

        set.add(new Student(1, "Alok"));

        set.add(new Student(1, "Singh"));

        System.out.println(set);

    }

}

// Topic 4: Deep vs Shallow Cloning

import java.util.\*;

class Book implements Cloneable {

    String title;

    Book(String t) {

        title = t;

    }

    public Book clone() throws CloneNotSupportedException {

        return (Book) super.clone();

    }

}

class Library implements Cloneable {

    List<Book> books;

    Library(List<Book> b) {

        books = b;

    }

    public Library clone() throws CloneNotSupportedException {

        Library copy = (Library) super.clone();

        copy.books = new ArrayList<>();

        for (Book bk : books)

            copy.books.add(bk.clone());

        return copy;

    }

}

public class Main4 {

    public static void main(String[] args) throws Exception {

        List<Book> list = new ArrayList<>();

        list.add(new Book("Java"));

        Library l1 = new Library(list);

        Library l2 = l1.clone();

        l2.books.get(0).title = "C++";

        System.out.println(l1.books.get(0).title);

        System.out.println(l2.books.get(0).title);

    }

}

// Topic 5: Member and Static Inner Classes

class University {

    String uniName = "Tech University";

    class Department {

        void showDept() {

            System.out.println("Dept of CSE at " + uniName);

        }

    }

    static class ExamCell {

        static void conductExam() {

            System.out.println("Exam conducted successfully.");

        }

    }

}

public class Main5 {

    public static void main(String[] args) {

        University u = new University();

        University.Department d = u.new Department();

        d.showDept();

        University.ExamCell.conductExam();

    }

}

//Topic 6: Local and Anonymous Inner Classes

interface Discount {

    double apply(double amt);

}

class Payment {

    void processTransaction(double amt) {

        class Validator {

            boolean valid(double a) {

                return a > 0;

            }

        }

        Validator v = new Validator();

        if (v.valid(amt)) {

            Discount d = new Discount() {

                public double apply(double a) {

                    return a \* 0.9;

                }

            };

            System.out.println("Final Amount: " + d.apply(amt));

        } else

            System.out.println("Invalid Payment");

    }

}

public class Main6 {

    public static void main(String[] args) {

        Payment p = new Payment();

        p.processTransaction(1000);

    }

}