import java.util.Scanner;

public class Problem1 {

    public static *void* main(String[] *args*) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the size of inputs: ");

*int* n = sc.nextInt();

        if (n < 3) {

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

            return;

        }

*int*[] numbers = new *int*[n];

        System.out.println("Enter " + n + " numbers:");

        for (*int* i = 0; i < n; i++) {

            numbers[i] = sc.nextInt();

        }

*int* max = numbers[0];

*int* min = numbers[0];

        for (*int* i = 1; i < n; i++) {

            if (numbers[i] > max) max = numbers[i];

            if (numbers[i] < min) min = numbers[i];

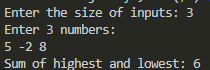
        }

        System.out.println("Sum of highest and lowest: " + (max + min));

        sc.close();

    }

}



import java.util.Scanner;

public class Problem2 {

    public static *void* main(String[] *args*) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a positive number n: ");

*int* n = sc.nextInt();

        if (n <= 0) {

            System.out.println("Wrong input");

            return;

        }

        for (*int* i = 1; i <= n; i++) {

*int* value = i \* (i + 1);  // 1\*2=2, 2\*3=6, 3\*4=12, 4\*5=20, ...

            System.out.print(value + " ");

        }

        System.out.println();

        sc.close();

    }

}



import java.util.Scanner;

public class Problem3 {

public static boolean isPalindrome(int num) {

int original = num;

int reversed = 0;

while (num > 0) {

int digit = num % 10;

reversed = reversed \* 10 + digit;

num /= 10;

}

return original == reversed;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter a number (>=11): ");

int n = sc.nextInt();

if (n < 11) {

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

return;

}

for (int i = 11; i <= n; i++) {

if (isPalindrome(i)) {

System.out.print(i + " ");

}

}

System.out.println();

sc.close();

}

}



import java.util.Arrays;

import java.util.Scanner;

class Applicant implements Comparable<Applicant> {

    private String name;

    private *int* subject1;

    private *int* subject2;

    private *int* subject3;

    private *int* total;

    private *double* percentage;

    public Applicant(String *name*, *int* *subject1*, *int* *subject2*, *int* *subject3*) {

        this.name = *name*;

        this.subject1 = *subject1*;

        this.subject2 = *subject2*;

        this.subject3 = *subject3*;

        this.total = totalCalculation();

        this.percentage = percentageCalculation();

    }

    public String getName() { return name; }

    public *int* getSubject1() { return subject1; }

    public *int* getSubject2() { return subject2; }

    public *int* getSubject3() { return subject3; }

    public *int* getTotal() { return total; }

    public *double* getPercentage() { return percentage; }

    public *void* setName(String *name*) { this.name = *name*; }

    public *void* setSubject1(*int* *subject1*) { this.subject1 = *subject1*; }

    public *void* setSubject2(*int* *subject2*) { this.subject2 = *subject2*; }

    public *void* setSubject3(*int* *subject3*) { this.subject3 = *subject3*; }

    public *int* totalCalculation() {

*int* sum = subject1 + subject2 + subject3;

        // Return 0 if any subject < 50 (fail)

        if (subject1 < 50 || subject2 < 50 || subject3 < 50) {

            return 0;

        }

        return sum;

    }

    public *double* percentageCalculation() {

        if (total == 0) return 0;

        return (total / 300.0) \* 100;

    }

    @Override

    public String toString() {

        return String.format("%-10s %-5d %-5d %-5d %-10d %-10.2f",

                name, subject1, subject2, subject3, total, percentage);

    }

    // For sorting by name ascending

    @Override

    public *int* compareTo(Applicant *other*) {

        return this.name.compareToIgnoreCase(*other*.name);

    }

}

public class Main {

    public static *void* main(String[] *args*) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of applicants: ");

*int* n = sc.nextInt();

        sc.nextLine();  // consume leftover newline

        Applicant[] applicants = new Applicant[n];

        for (*int* i = 0; i < n; i++) {

            System.out.print("Enter details for applicant " + (i + 1) + " (Name,subj1,subj2,subj3): ");

            String input = sc.nextLine();

            String[] parts = input.split(",");

            if (parts.length != 4) {

                System.out.println("Invalid input format, please enter again.");

                i--;

                continue;

            }

            String name = parts[0];

            try {

*int* sub1 = Integer.parseInt(parts[1]);

*int* sub2 = Integer.parseInt(parts[2]);

*int* sub3 = Integer.parseInt(parts[3]);

                applicants[i] = new Applicant(name, sub1, sub2, sub3);

            } catch (NumberFormatException e) {

                System.out.println("Invalid marks input, please enter again.");

                i--;

            }

        }

        // Filter applicants who passed (total > 0)

        Applicant[] passedApplicants = Arrays.stream(applicants)

                .filter(*a* *->* *a*.getTotal() > 0)

                .toArray(Applicant[]::new);

        // Sort by name ascending

        Arrays.sort(passedApplicants);

        // Print header

        System.out.printf("%-10s %-5s %-5s %-5s %-10s %-10s\n",

                "Name", "S1", "S2", "S3", "Total", "Percentage");

        // Print passed applicants

        for (Applicant a : passedApplicants) {

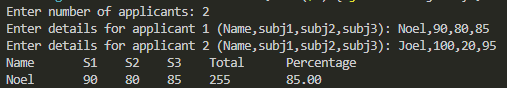
            System.out.println(a);

        }

        sc.close();

    }

}



import java.util.\*;

abstract class Employee implements Comparable<Employee> {

    private String employeeId;

    private String employeeName;

    private String department;

    public Employee(String *employeeId*, String *employeeName*, String *department*) {

        this.employeeId = *employeeId*;

        this.employeeName = *employeeName*;

        this.department = *department*;

    }

    // Getters and setters

    public String getEmployeeId() { return employeeId; }

    public String getEmployeeName() { return employeeName; }

    public String getDepartment() { return department; }

    public *void* setEmployeeId(String *employeeId*) { this.employeeId = *employeeId*; }

    public *void* setEmployeeName(String *employeeName*) { this.employeeName = *employeeName*; }

    public *void* setDepartment(String *department*) { this.department = *department*; }

    // Abstract method to calculate tax

    public abstract *double* calculateTax();

    @Override

    public *int* compareTo(Employee *other*) {

        return this.employeeName.compareToIgnoreCase(*other*.employeeName);

    }

    @Override

    public String toString() {

        return String.format("%-10s %-20s %-15s", employeeId, employeeName, department);

    }

}

class PermanentEmployee extends Employee {

    private *double* monthlySalary;

    private *double* pf;   // 15% of monthlySalary

    private *double* tax;  // 10% of annual salary

    public PermanentEmployee(String *employeeId*, String *employeeName*, String *department*, *double* *monthlySalary*) {

        super(*employeeId*, *employeeName*, *department*);

        this.monthlySalary = *monthlySalary*;

        this.pf = *monthlySalary* \* 0.15;

        this.tax = calculateTax();

    }

    public *double* getMonthlySalary() { return monthlySalary; }

    public *double* getPf() { return pf; }

    public *double* getTax() { return tax; }

    public *void* setMonthlySalary(*double* *monthlySalary*) {

        this.monthlySalary = *monthlySalary*;

        this.pf = *monthlySalary* \* 0.15;

        this.tax = calculateTax();

    }

    @Override

    public *double* calculateTax() {

        // 10% of annual salary

        return monthlySalary \* 12 \* 0.10;

    }

    @Override

    public String toString() {

        return String.format("%-10s %-20s %-15s Monthly Salary: %.2f PF: %.2f Tax: %.2f",

                getEmployeeId(), getEmployeeName(), getDepartment(),

                monthlySalary, pf, tax);

    }

}

class ContractualEmployee extends Employee {

    private *int* contractPeriod; // in months

    private *double* contractAmount;

    private *double* tax; // 10% of contractAmount

    public ContractualEmployee(String *employeeId*, String *employeeName*, String *department*, *int* *contractPeriod*, *double* *contractAmount*) {

        super(*employeeId*, *employeeName*, *department*);

        this.contractPeriod = *contractPeriod*;

        this.contractAmount = *contractAmount*;

        this.tax = calculateTax();

    }

    public *int* getContractPeriod() { return contractPeriod; }

    public *double* getContractAmount() { return contractAmount; }

    public *double* getTax() { return tax; }

    public *void* setContractPeriod(*int* *contractPeriod*) {

        this.contractPeriod = *contractPeriod*;

    }

    public *void* setContractAmount(*double* *contractAmount*) {

        this.contractAmount = *contractAmount*;

        this.tax = calculateTax();

    }

    @Override

    public *double* calculateTax() {

        return contractAmount \* 0.10;

    }

    @Override

    public String toString() {

        return String.format("%-10s %-20s %-15s Contract Period: %d Contract Amount: %.2f Tax: %.2f",

                getEmployeeId(), getEmployeeName(), getDepartment(),

                contractPeriod, contractAmount, tax);

    }

}

public class Main {

    public static *void* main(String[] *args*) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of employees: ");

*int* n = sc.nextInt();

        sc.nextLine(); // consume newline

        Employee[] employees = new Employee[n];

        System.out.println("Enter details for each employee in comma separated format:");

        System.out.println("For Permanent Employee: employeeId,employeeName,department,monthlySalary");

        System.out.println("For Contractual Employee: employeeId,employeeName,department,contractPeriod,contractAmount");

        System.out.println("Example: P101,Eric Miller,Finance,35000.00");

        System.out.println("Example: C1002,Roger Steven,Sales,5,750000.00");

        for (*int* i = 0; i < n; i++) {

            System.out.print("Employee " + (i + 1) + ": ");

            String input = sc.nextLine();

            String[] parts = input.split(",");

            try {

                String id = parts[0].trim();

                String name = parts[1].trim();

                String dept = parts[2].trim();

                if (id.startsWith("P")) {

                    // Permanent employee, expect 4 parts

*double* monthlySalary = Double.parseDouble(parts[3].trim());

                    employees[i] = new PermanentEmployee(id, name, dept, monthlySalary);

                } else if (id.startsWith("C")) {

                    // Contractual employee, expect 5 parts

*int* contractPeriod = Integer.parseInt(parts[3].trim());

*double* contractAmount = Double.parseDouble(parts[4].trim());

                    employees[i] = new ContractualEmployee(id, name, dept, contractPeriod, contractAmount);

                } else {

                    System.out.println("Invalid employee id format. Skipping entry.");

                    i--;

                    continue;

                }

            } catch (Exception e) {

                System.out.println("Invalid input format or data. Please enter again.");

                i--;

            }

        }

        // Sort by employee name

        Arrays.sort(employees);

        System.out.println("\nEmployee details sorted by name:\n");

        for (Employee emp : employees) {

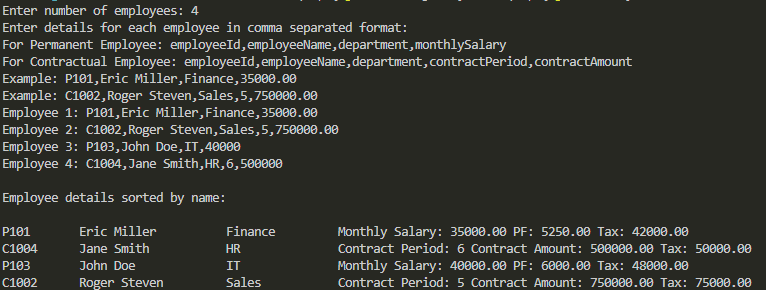
            System.out.println(emp);

        }

        sc.close();

    }

}



import java.util.\*;

import java.util.stream.\*;

class Agent {

    private String name;

    private *long* generatedFund;

    public Agent(String *name*, *long* *generatedFund*) {

        this.name = *name*;

        this.generatedFund = *generatedFund*;

    }

    public String getName() { return name; }

    public *long* getGeneratedFund() { return generatedFund; }

    public *void* setName(String *name*) { this.name = *name*; }

    public *void* setGeneratedFund(*long* *generatedFund*) { this.generatedFund = *generatedFund*; }

}

public class Main {

    public static *void* main(String[] *args*) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of agents: ");

*int* n = sc.nextInt();

        sc.nextLine(); // consume newline

        List<Agent> agents = new ArrayList<>();

        for (*int* i = 0; i < n; i++) {

            System.out.print("Enter name and generated fund (comma separated): ");

            String[] input = sc.nextLine().split(",");

            String name = input[0].trim();

*long* fund = Long.parseLong(input[1].trim());

            agents.add(new Agent(name, fund));

        }

        // Use Stream API to create Map<String, String> of name and stars

        Map<String, String> gradedAgents = agents.stream()

                .collect(Collectors.toMap(

                        Agent::getName,

*agent* *->* getStars(*agent*.getGeneratedFund())

                ));

        // Display each agent with stars

        gradedAgents.forEach((*name*, *stars*) *->* System.out.println(*name* + "=" + *stars*));

        sc.close();

    }

    private static String getStars(*long* *fund*) {

        if (*fund* >= 2000000) return "\*\*\*\*\*";

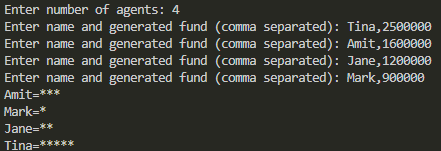
        else if (*fund* >= 1500000) return "\*\*\*";

        else if (*fund* >= 1000000) return "\*\*";

        else return "\*";

    }

}



import java.util.\*;

// Custom Exceptions

class PriceException extends Exception {

    public PriceException(String *message*) { super(*message*); }

}

class EssentialCommodityException extends Exception {

    public EssentialCommodityException(String *message*) { super(*message*); }

}

class GradeMismatchException extends Exception {

    public GradeMismatchException(String *message*) { super(*message*); }

}

// Item class

class Item {

    private Integer id;

    private String name;

    private Double purchasedPrice;

    private Double salesPrice;

    private String grade;

    public Item(Integer *id*, String *name*, Double *purchasedPrice*, Double *salesPrice*, String *grade*) {

        this.id = *id*;

        this.name = *name*;

        this.purchasedPrice = *purchasedPrice*;

        this.salesPrice = *salesPrice*;

        this.grade = *grade*;

    }

    // Getters and setters

    public Integer getId() { return id; }

    public String getName() { return name; }

    public Double getPurchasedPrice() { return purchasedPrice; }

    public Double getSalesPrice() { return salesPrice; }

    public String getGrade() { return grade; }

    public *void* setId(Integer *id*) { this.id = *id*; }

    public *void* setName(String *name*) { this.name = *name*; }

    public *void* setPurchasedPrice(Double *purchasedPrice*) { this.purchasedPrice = *purchasedPrice*; }

    public *void* setSalesPrice(Double *salesPrice*) { this.salesPrice = *salesPrice*; }

    public *void* setGrade(String *grade*) { this.grade = *grade*; }

    @Override

    public String toString() {

        return String.format("%-5d %-20s %-10.2f %-10.2f %-5s",

                id, name, purchasedPrice, salesPrice, grade);

    }

    // Override equals and hashCode to check uniqueness by id

    @Override

    public *boolean* equals(Object *o*) {

        if (this == *o*) return true;

        if (!(*o* instanceof Item)) return false;

        Item item = (Item) *o*;

        return Objects.equals(id, item.id);

    }

    @Override

    public *int* hashCode() {

        return Objects.hash(id);

    }

}

// Service class

class ItemService {

    public Set<Item> collectAllItems(List<String> *itemStrings*) {

        Set<Item> items = new HashSet<>();

        for (String itemStr : *itemStrings*) {

            try {

                Item item = parseItem(itemStr);

                // Validation

                if (!item.getGrade().equals("N") && !item.getGrade().equals("E")) {

                    throw new GradeMismatchException("Grade must be N or E");

                }

                if (item.getSalesPrice() <= item.getPurchasedPrice()) {

                    throw new PriceException("Sales price must be greater than purchase price");

                }

                if (item.getGrade().equals("E")) {

*double* maxSalesPrice = item.getPurchasedPrice() \* 1.25;

                    if (item.getSalesPrice() > maxSalesPrice) {

                        throw new EssentialCommodityException("Sales price cannot exceed 125% of purchase price for Essential Commodity");

                    }

                }

                // Add item to set - duplicates automatically rejected by equals/hashCode

*boolean* added = items.add(item);

                if (!added) {

                    System.out.println("Duplicate item id found and rejected: " + item.getId());

                }

            } catch (Exception e) {

                System.out.println("Error processing item: " + itemStr);

                System.out.println("Reason: " + e.getMessage());

            }

        }

        return items;

    }

    private Item parseItem(String *input*) throws Exception {

        // Expected format: id,name,purchasedPrice,salesPrice,grade

        String[] parts = *input*.split(",");

        if (parts.length != 5) throw new Exception("Invalid input format");

        Integer id = Integer.parseInt(parts[0].trim());

        String name = parts[1].trim();

        Double purchasedPrice = Double.parseDouble(parts[2].trim());

        Double salesPrice = Double.parseDouble(parts[3].trim());

        String grade = parts[4].trim();

        return new Item(id, name, purchasedPrice, salesPrice, grade);

    }

}

// Main class

public class Main {

    public static *void* main(String[] *args*) {

        Scanner sc = new Scanner(System.in);

        ItemService service = new ItemService();

        System.out.print("Enter number of items: ");

*int* n = sc.nextInt();

        sc.nextLine(); // consume newline

        List<String> inputs = new ArrayList<>();

        System.out.println("Enter item details (id,name,purchasedPrice,salesPrice,grade):");

        for (*int* i = 0; i < n; i++) {

            String line = sc.nextLine();

            inputs.add(line);

        }

        Set<Item> acceptedItems = service.collectAllItems(inputs);

        System.out.println("\nAccepted items:");

        System.out.printf("%-5s %-20s %-10s %-10s %-5s\n", "ID", "Name", "Purchased", "Sales", "Grade");

        for (Item item : acceptedItems) {

            System.out.println(item);

        }

        sc.close();

    }

}

