МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ ПО

ИНФОРМАТИКА

23 август 2024 г.

ПРОФИЛИРАНА ПОДГОТОВКА

ВАРИАНТ 2

Задача от 1. до 16. Ключ с верните отговори

Въпрос №	Верен отговор	Брой точки
1.	Г	1
2.	Г	1
3.	Г	1
4.	Г	1
5.	В	1
6.	А	1
7.	Г	1
8.	Б	1
9.	В	1
10.	А	1
11.	В	1
12.	Б	1
13.	А	1
14.	В	1
15.	Б	1
16.	В	1

Задача 17. – 2 точки

<1> етикет / Label

<2> комбинирана кутия /Combo Box

<3> поле за отметка / Check Box

<4> радио бутон / Radio Button

<5> бутон / Button

Задача 18. – 2 точки

А) is-a / е

Б) has-a / има

Задача 19. – 2 точки

(1) 4

(2) 2

(3) 5, 20, 6, 0 или 0, 6, 20, 5

(4) 1, 1

Задача 20. – 3 точки

<1> **За C#:** `Convert.ToInt32(Console.ReadLine())`

За Java: `scr.nextInt()`

<2> `i <= number`

<3> `i++ или ++i или i+=1 или i= i+1`

<4> `number % i == 0`

<5> `number % i == 0`

<6> `number /= i или number = number / i`

Задача 21. – 3 точки

```
Welcome to Java
Welcome to Java and C#
End of the block
```

Задача 22. – 3 точки

C#	Java
Ред 6 for (int j = 0; j < n; j++) Ред 11 if(arr[i][n-1-i] % 2 != 0) Ред 12 sum += arr[i][n-1-i];	Ред 6 for (int j = 0; j< n; j++) Ред 11 if(arr[i][n-1-i] % 2 != 0) Ред 12 sum += arr[i][n-1-i];

Задача 23. – 3 точки

(1) Брой на аргументите
(3) Подредбата (позицията) на аргументите, които са от различен тип
(4) Тип на аргументите
Признава се и ако са записани само номерата или само думите, като не се взема под внимание реда на изброяването.

Задача 24. – 3 точки

ред	C#	Java
< 1 >	Add	add
< 2 >	Push	push / add
< 3 >	Enqueue	offer / add

Задача 25. – 3 точки

```
SELECT s.ProfileId, p.Name,  MAX(s.Points) AS [Max Points]
FROM Students  as s
JOIN Profiles as p ON s.ProfileId = p.Id
GROUP BY s.ProfileId, p.Name
```

Задача 26 – 15 точки

Примерно решение

C#

```
using System;

namespace DZI
{
    class Zad26
    {
        static void Main(string[] args)
        {
            int n = Convert.ToInt32(Console.ReadLine());
            int[] profit = new int[n];

            string[] input = Console.ReadLine().Split(' '); ;
            for (int i = 0; i < n; i++)
            {
                profit[i] = Convert.ToInt32(input[i]);
            }

            int longestStreak = 1;
            int currentStreak = 1;
            for (int i = 1; i < n; ++i)
            {
                if (profit[i] <= profit[i - 1])
                {
                    if (currentStreak > longestStreak)
                    {
                        longestStreak = currentStreak;
                    }
                    currentStreak = 0;
                }
                ++currentStreak;
            }
```

```
            if (currentStreak > longestStreak)
            {
                longestStreak = currentStreak;
            }
            Console.WriteLine("The longest period with bigger profit is "
+ longestStreak + " mounths.");

            int smallestProfit = profit[0];
            int targetPeriod = 0;
            for (int i = 1; i < n; i++)
            {
                if (profit[i] < smallestProfit)
                {
                    smallestProfit = profit[i];
                    targetPeriod = i;
                }
            }
            int closestProfit;
            if (targetPeriod == n - 1)
            {
                closestProfit = profit[n - 2];
            }
            else
            {
                closestProfit = profit[targetPeriod + 1];
            }
            string str = "%.";
            double smallerPercentage = (closestProfit - smallestProfit) /
(double)closestProfit * 100;
            Console.WriteLine($"Smaller with
{smallerPercentage:F2}{str}");
        }
    }
}
```

Java

```
package DZI;

import java.util.Scanner;

public class Zad26 {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] profit = new int[n];

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

```
            profit[i] = scanner.nextInt();
        }

        int longestStreak = 1;
        int currentStreak = 1;
        for (int i = 1; i < n; ++i) {
            if (profit[i] <= profit[i - 1]) {
                if (currentStreak > longestStreak) {
                    longestStreak = currentStreak;
                }
                currentStreak = 0;
            }
            ++currentStreak;
        }
        if (currentStreak > longestStreak) {
            longestStreak = currentStreak;
        }
        System.out.println("The longest period with bigger profit is " +
longestStreak + " mounths.");

        int smallestProfit = profit[0];
        int targetPeriod = 0;
        for (int i = 1; i < n; i++) {
            if (profit[i] < smallestProfit) {
                smallestProfit = profit[i];
                targetPeriod = i;
            }
        }
        int closestProfit;
        if (targetPeriod == n - 1) {
            closestProfit = profit[n - 2];
        } else {
            closestProfit = profit[targetPeriod + 1];
        }
        String str ="%.";
        double smallerPercentage = (closestProfit - smallestProfit) /
(double) closestProfit * 100;
        System.out.println(String.format("Smaller with %.2f%s",
smallerPercentage, str));
    }
}
```

Задача 27 – 20 точки

Примерно решение

```
CREATE DATABASE Travel;
USE Travel;
```

-- А

MySql	SQL
CREATE TABLE Countries (ID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(30));	CREATE TABLE Countries (ID INT Identity PRIMARY KEY, "Name" NVARCHAR(30))

-- Б

MySql	SQL
CREATE TABLE Destinations (ID INT PRIMARY KEY AUTO_INCREMENT, Town VARCHAR(30), Distance INT, Duration INT, Price DECIMAL(10, 2), CountryId INT, FOREIGN KEY (CountryId) REFERENCES Countries (ID));	CREATE TABLE Destinations (ID INT Identity PRIMARY KEY, Town NVARCHAR(30), Distance INT, Duration INT, Price DECIMAL(10, 2), CountryId INT FOREIGN KEY REFERENCES Countries(ID))

```
-- В
INSERT INTO Countries (Name)
VALUES ('France'), ('Germany'), ('Italy'), ('Spain'), ('Austria');

-- Г
INSERT INTO Destinations (Town, Distance, Duration, Price, CountryId)
VALUES
  ('Paris', 2169 , 4, 1800.00, 1),
  ('Berlin', 4006, 6, 2100.00, 2),
  ('Rome', 1666, 3, 1500.00, 3),
  ('Madrid', 2966, 7, 1800.00, 4),
  ('Milan', 1408, 4, 1900.00, 3),
  ('Venice', 1152 , 3, 1200.00, 3),
  ('Barcelona', 2375, 7, 2800.00, 4);

-- Д
UPDATE Destinations
SET Price = 1700
WHERE Town = 'Paris';
```

```
-- Е
DELETE FROM Destinations
WHERE Town LIKE 'Ba%';

-- Ж
SELECT Town, Distance
FROM Destinations
WHERE Distance BETWEEN 1500 AND 2500;

-- З
```

MySql	SQL
SELECT Town, Price FROM Destinations ORDER BY Price DESC LIMIT 1;	SELECT TOP 1 Town, Price FROM Destinations ORDER BY Price DESC

```
-- И
SELECT c.Name, COUNT(d.CountryId)
FROM Countries AS c
JOIN Destinations AS d ON d.CountryId = c.ID
GROUP BY c.Name
ORDER BY COUNT(d.CountryId) DESC, c.Name;
```

Задача 28 – 25 точки

Примерно решение

C#

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace Zad28
{
    class Zad28
    {
        public static void Main(string[] args)
        {
            var clubMembers = new List<ClubMember>();
            using (StreamReader reader = new StreamReader("input.txt"))
            {
                string input;
                while ((input = reader.ReadLine()) != null)
                {
                    string[] line = input.Split(' ');
                    int len = line.Length;
                    switch (len)
                    {
                        case 9:
                            try
```

```
                            {
                                FootballPlayer newPlayer = new
FootballPlayer(line[0], line[1],
                                        int.Parse(line[2]),
double.Parse(line[3]),
                                        line[4], int.Parse(line[5]),
int.Parse(line[6]),
                                        int.Parse(line[7]),
int.Parse(line[8]));
                                clubMembers.Add(newPlayer);
                            }
                            catch (Exception e)
                            {
                                Console.WriteLine(e.Message);
                            }
                            break;
                        case 6:
                            try
                            {
                                Coach coach = new Coach(line[0], line[1],
int.Parse(line[2]),
                                        double.Parse(line[3]), line[4],
int.Parse(line[5]));
                                clubMembers.Add(coach);
                            }
                            catch (Exception e)
                            {
                                Console.WriteLine(e.Message);
                            }
                            break;
                        case 5:
                            try
                            {
                                Director dirctor = new Director(line[0],
line[1],
                                        int.Parse(line[2]),
double.Parse(line[3]), line[4]);
                                clubMembers.Add(dirctor);
                            }
                            catch (Exception e)
                            {
                                Console.WriteLine(e.Message);
                            }
                            break;
                    }
                }
            }
            clubMembers.Sort((x, y) => x.Age.CompareTo(y.Age));
            foreach (ClubMember member in clubMembers)
            {
                member.Info();
                Console.WriteLine("********************");
            }
            double maxSalary = clubMembers.Max(cm => cm.Salary);
```

```
            ClubMember memberMaxSalery = clubMembers.FirstOrDefault(cm =>
cm.Salary == maxSalary); ;

            Console.WriteLine($"The person with the highest salary in the club
is {memberMaxSalery.FirstName} " +
                $"{memberMaxSalery.LastName} with {memberMaxSalery.Salary:f2}
lv salary.");

            FootballPlayer goalMaster = clubMembers
                .Where(x => x.GetType() == typeof(FootballPlayer))
                .Select(x => x as FootballPlayer)
                .OrderByDescending(x => x.Goals)
                .FirstOrDefault();

            Console.WriteLine($"The team's top scorer is {goalMaster.FirstName}
" +
                $"{goalMaster.LastName} with {goalMaster.Goals} goals.");
        }
    }

abstract class ClubMember
    {
        string firstName, lastName;
        int age;
        double salary;

        public string FirstName
        {
            get
            {
                return this.firstName;
            }
            set
            {
                if (value == "") throw new Exception("The name can’t be an
empty string!");
                else this.firstName = value;
            }
        }

        public string LastName
        {
            get
            {
                return this.lastName;
            }
            set
            {
                if (value == "") throw new Exception("The name can’t be an
empty string!");
                else this.lastName = value;
            }
        }
        public int Age
```

```
        {
            get
            {
                return this.age;
            }
            set
            {
                if (value <= 17) throw new Exception("Age must be greater than
17 years!");
                else this.age = value;
            }
        }

        public double Salary
        {
            get
            {
                return this.salary;
            }
            set
            {
                if (value <= 0) throw new Exception("Salary must be a positive
number!");
                else this.salary = value;
            }
        }

        public ClubMember(string firstName, string lastName, int age, double
salary)
        {
            FirstName = firstName;
            LastName = lastName;
            Age = age;
            Salary = salary;
        }

        public abstract void Info();
    }

class Director : ClubMember
    {
        public Director(string firstName, string lastName, int age, double
salary, string directorType) : base(firstName, lastName, age, salary)
        {
            DirectorType = directorType;
        }

        public string DirectorType { get; set; }

    public override void Info()
        {
            Console.WriteLine($"{this.DirectorType} director: {this.FirstName}
{this.LastName}");
            Console.WriteLine($"salary: {this.Salary:F2} lv");
```

```
            Console.WriteLine($"age: {this.Age} years");
        }
    }

class Coach : ClubMember
    {
        public string CoachType { get; set; }

        public int ContractLength { get; set; }

        public Coach(string firstName, string lastName, int age, double salary,
string coachType, int contractLength) : base(firstName, lastName, age, salary)
        {
            CoachType = coachType;
            ContractLength = contractLength;
        }

    public override void Info()
        {
            Console.WriteLine($"{this.CoachType} coach: {this.FirstName}
{this.LastName}");
            Console.WriteLine($"salary: {this.Salary:f2} lv");
            Console.WriteLine($"age: {this.Age} years");
        }
    }
class FootballPlayer : ClubMember
    {
        public string Position { get; set; }

        public int ContractLength { get; set; }

        public int Matches { get; set; }

        public int Goals { get; set; }

        public int Assist { get; set; }

        public FootballPlayer(string firstName, string lastName, int age,
double salary, string position, int contractLength, int matches, int goals, int
assist) : base(firstName, lastName, age, salary)
        {
            Position = position;
            ContractLength = contractLength;
            Matches = matches;
            Goals = goals;
            Assist = assist;
        }

    public override void Info()
        {
            Console.WriteLine($"{FirstName} {LastName} - {Position}");
            Console.WriteLine($"salary: {Salary:f2} lv");
            Console.WriteLine($"age: {Age} years");
```

```
            Console.WriteLine($"{Goals} goals and {Assist} assists in {Matches}
matches");
        }
    }
}
```

Java

```
package DZI;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;

public class Zad28 {

    public static void main(String[] args) throws FileNotFoundException {
        List<ClubMember> clubMembers = new ArrayList<>();
        try ( Scanner reader = new Scanner(new FileInputStream("input.txt"))) {
            while (reader.hasNext()) {
                String[] line = reader.nextLine().split(" ");
                int len = line.length;
                switch (len) {
                    case 9:
                        try {
                        FootballPlayer player = new FootballPlayer(line[0],
line[1],
                                Integer.parseInt(line[2]),
Double.parseDouble(line[3]),
                                line[4], Integer.parseInt(line[5]),
Integer.parseInt(line[6]),
                                Integer.parseInt(line[7]),
Integer.parseInt(line[8]));
                        clubMembers.add(player);
                    } catch (Exception e) {
                        System.out.println(e.getMessage());
                    }
                    break;
                    case 6:
                        try {
                        Coach coach = new Coach(line[0], line[1],
Integer.parseInt(line[2]),
                                Double.parseDouble(line[3]), line[4],
Integer.parseInt(line[5]));
                        clubMembers.add(coach);
                    } catch (Exception e) {
                        System.out.println(e.getMessage());
                    }
                    break;
                    case 5:
                        try {
                        Director dirctor = new Director(line[0], line[1],
                                Integer.parseInt(line[2]),
Double.parseDouble(line[3]), line[4]);
                        clubMembers.add(dirctor);
                    } catch (Exception e) {
                        System.out.println(e.getMessage());
```

```
                    }
                    break;
                }
            }
        }
        Collections.sort(clubMembers, (p1, p2) -> Integer.compare(p1.age,
p2.age));
        for (ClubMember member : clubMembers) {
            member.info();
            System.out.println("********************");
        }
        ClubMember maxSalery = clubMembers.get(0);
        for (ClubMember member : clubMembers) {
            if(maxSalery.getSalary() < member.getSalary()) maxSalery = member;
        }
        System.out.println(String.format("The person with the highest salary in
the club is "
                + "%s %s with %.2f lv salary.", maxSalery.getFirstName(),
maxSalery.getLastName(),
                maxSalery.getSalary()));

        FootballPlayer player, goalMaster = null;
        for (ClubMember member : clubMembers) {
            if(member instanceof FootballPlayer){
                player = (FootballPlayer) member;
                if(goalMaster == null || player.getGoals() >
goalMaster.getGoals())
                    goalMaster = player;
            }
        }
        System.out.println(String.format("The team's top scorer is %s %s with
%d goals.",
                goalMaster.getFirstName(), goalMaster.getLastName(),
goalMaster.getGoals()));

    }
}

abstract class ClubMember {
    String firstName, lastName;
    int age;
    double salary;

    public void setFirstName(String firstName) throws Exception{
        if(firstName.equals("")) throw new Exception("The name can’t be an
empty string!");
        else this.firstName = firstName;
    }

    public void setLastName(String lastName) throws Exception{
        if(lastName.equals("")) throw new Exception("The name can’t be an empty
string!");
        else this.lastName = lastName;
    }
```

```
    public void setAge(int age) throws Exception{
        if(age <= 17) throw new Exception("Age must be greater than 17
years!");
        else this.age = age;
    }

    public void setSalary(double salary) throws Exception{
        if(salary <= 0) throw new Exception("Salary must be a positive
number!");
        else this.salary = salary;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public int getAge() {
        return age;
    }

    public double getSalary() {
        return salary;
    }

    public ClubMember(String firstName, String lastName, int age, double
salary) throws Exception{
        setFirstName(firstName);
        setLastName(lastName);
        setAge(age);
        setSalary(salary);
    }

    public abstract void info();
}
public class Director extends ClubMember{
    String directorType;

    public Director(String firstName, String lastName, int age, double salary,
String directorType) throws Exception {
        super(firstName, lastName, age, salary);
        this.directorType = directorType;
    }

    public String getDirectorType() {
        return directorType;
    }

    public void setDirectorType(String directorType) {
        this.directorType = directorType;
```

```
    }

    @Override
    public void info() {
        System.out.println(String.format("%s director: %s %s", directorType,
firstName, lastName));
        System.out.println(String.format("salary: %.2f lv", salary));
        System.out.println(String.format("age: %d years", age));
    }
}

public class Coach extends ClubMember{
    String coachType;
    int contractLength;

    public String getCoachType() {
        return coachType;
    }

    public void setCoachType(String coachType) {
        this.coachType = coachType;
    }

    public int getContractLength() {
        return contractLength;
    }

    public void setContractLength(int contractLength) {
        this.contractLength = contractLength;
    }

    public Coach(String firstName, String lastName, int age, double salary,
String coachType, int contractLength) throws Exception {
        super(firstName, lastName, age, salary);
        this.coachType = coachType;
        this.contractLength = contractLength;
    }

    @Override
    public void info() {
        System.out.println(String.format("%s coach: %s %s", coachType ,
firstName, lastName));
        System.out.println(String.format("salary: %.2f lv", salary));
        System.out.println(String.format("age: %d years", age));
    }
}

public class FootballPlayer extends ClubMember {

    String position;
    int contractLength, matches, goals, assist;

    public String getPosition() {
        return position;
```

```
    }

    public void setPosition(String position) {
        this.position = position;
    }

    public int getContractLength() {
        return contractLength;
    }

    public void setContractLength(int contractLength) {
        this.contractLength = contractLength;
    }

    public int getMatches() {
        return matches;
    }

    public void setMatches(int matches) {
        this.matches = matches;
    }

    public int getGoals() {
        return goals;
    }

    public void setGoals(int goals) {
        this.goals = goals;
    }

    public int getAssist() {
        return assist;
    }

    public void setAssist(int assist) {
        this.assist = assist;
    }

    public FootballPlayer(String firstName, String lastName, int age, double
salary, String position, int contractLength, int matches, int goals, int
assist) throws Exception {
        super(firstName, lastName, age, salary);
        this.position = position;
        this.contractLength = contractLength;
        this.matches = matches;
        this.goals = goals;
        this.assist = assist;
    }

    @Override
    public void info() {
        System.out.println(String.format("%s %s - %s", firstName, lastName,
position));
        System.out.println(String.format("salary: %.2f lv", salary));
```

```
        System.out.println(String.format("age: %d years", age));
        System.out.println(String.format("%d goals and %d assists in %d
matches", goals , assist, matches));
    }
}
```