МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ ПО

ИНФОРМАТИКА

23 май 2023 г.

ПРОФИЛИРАНА ПОДГОТОВКА

ВАРИАНТ 2

Задача от 1. до 16. Ключ с верните отговори

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

Задача 17. Примерно решение – 3 точки

```
SELECT cooks.name
FROM cooks
JOIN restaurants ON restaurant_id = restaurants.id
WHERE stars = 5
ORDER BY cooks.name;
```

Задача 18. Примерно решение – 3 точки

1) Броят на елементите в стека е 3

2) Броят на елементите в опашката е 2

3) Елементите в опашката са -1 и -2

Задача 19. Примерно решение – 3 точки

(1) първият / последният / произволен

(2) по-малка

(3) записва / присвоява / запомня / замества

(4) `arr.Length` (за C#) / `arr.lenght` (за Java)

(5) `arr[i] < minEl` / `arr[i] <= minEl` / `minEl > arr[i]` / `minEl >= arr[i]`

(6) `minEl`

Задача 20. – 3 точки

Отговор: 16

Задача 21. Примерно решение – 3 точки

1) *има* или *has-a*

2) *е* или *is-a*

3) *има* или *has-a*

Задача 22. Примерно решение – 3 точки

1) `onLowerLine`

2) `(y == y2) && (x >= x1) && (x <= x2)`

Задача 23. Примерно решение – 3 точки

1) Всички думи от файла "`words.txt`" ще се запишат във файла "`groups.txt`" на групи по 5 на ред, разделени с интервал.

groups.txt
table bed TV computer DVD parket water other

2) Препоръчително е, защото при извикването на методите `Close()` или `close()` обектите, за които са били извикани методите приключват нормално работата, т.е. файловете се затварят и се освобождават използваните от тях ресурси.

Задача 24. Примерно решение – 3 точки

```
CREATE TABLE Booking(
agency_name VARCHAR(20) NOT NULL FOREIGN KEY REFERENCES
Agency([name]),
client_id int NOT NULL REFERENCES Client(id),
booking_date DATETIME,
[status] char(1),
PRIMARY KEY(agency_name, client_id)
);
```

Или

```
CREATE TABLE Booking(
agency_name VARCHAR(20) NOT NULL,
client_id int NOT NULL REFERENCES Client(id),
booking_date DATETIME,
[status] char(1),
FOREIGN KEY (agency_name) REFERENCES Agency([name]),
PRIMARY KEY(agency_name, client_id)
);
```

Или

```
CREATE TABLE Booking(
agency_name VARCHAR(20) NOT NULL CONSTRAINT FK1 REFERENCES
Agency(name),
client_id int NOT NULL CONSTRAINT FK2 REFERENCES Client(id),
booking_date DATETIME,
```

```
status char(1),
PRIMARY KEY(agency_name, client_id)
);
```

Или

```
CREATE TABLE booking(
agency_name VARCHAR(20) NOT NULL,
client_id int NOT NULL,
booking_date DATE,
status char(1),
FOREIGN KEY(agency_name) REFERENCES Agency(name),
FOREIGN KEY(client_id) REFERENCES Client(id),
PRIMARY KEY(agency_name, client_id)
);
```

Задача 25. Примерно решение – 10 точки

За C#

```
using System;
namespace DZI
{
    class Zad25
    {
        static void Main(string[] args)
        {
            try
            {
                int number = Int32.Parse(Console.ReadLine());
                int copy = number;
                while (copy != 0)
                {
                    if (copy % 10 == 0 || number % (copy % 10) != 0)
                    {
                        Console.WriteLine("No");
                        return;
                    }
                    copy /= 10;
                }
                Console.WriteLine("Yes");
            }
            catch (Exception e)
            {
                Console.WriteLine("Something went wrong!");
            }
        }
    }
}
```

За Java

```
package variant1;
import java.util.Scanner;
public class Zad25 {
     public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        try{
            int number = input.nextInt();
            int copy = number;
            while(copy != 0){
                if(copy % 10 == 0 || number % (copy % 10) != 0){
                    System.out.println("No");
                    return;
                }
                copy /= 10;
            }
            System.out.println("Yes");
        }
        catch(Exception e){
            System.out.println("Something went wrong!");
        }
    }
}
```

Задача 26: Примерно решение – 15 точки

За C#

```
using System;
using System.Collections.Generic;
using System.Linq;
namespace DZI
{
    class Zad26
    {
        static void Main(string[] args)
        {
            List<string> words = new List<string>();

            var input = Console.ReadLine().Split(' ').ToList();
            while (input[0] != "END")
            {
                switch (input[0])
                {
                    case "Add":
                        for (int i = 1; i < input.Count; i++)
                            words.Add(input[i]);
                        break;
                    case "Update":
                        for (int i = 0; i < words.Count; i++)
                        {
                            string firstLetter =
                               words[i].Substring(0, 1).ToUpper();
                            string restOfWord =
                               words[i].Substring(1);
                            words[i] = firstLetter + restOfWord;
                        }
                        break;
                    case "Remove":
                        int index = int.Parse(input[1]);
                        words.RemoveAt(index);
                        break;
                    case "Search":
                        string search = input[1];
                        if (words.IndexOf(search) != -1)
                    Console.WriteLine(words[words.IndexOf(search)]);
                        else Console.WriteLine("Not contained.");
                        break;
                    case "Length":
```

```
                    int length = int.Parse(input[1]);
                    var result = words.Where(x => x.Length ==
                                                length).ToList();
                    if (result.Any())
                        Console.WriteLine(string.Join("-",
                                                    result));
                    else Console.WriteLine("Not contained.");
                    break;
                case "Insert":
                    int position = int.Parse(input[1]);
                    string item = input[2];
                    try
                    {
                        words.Insert(position, item);
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine("There are not enough
                                    items in the list.", ex);
                    }
                    break;
                case "Print":
                    Console.WriteLine(string.Join("; ", words));
                    break;
            }
            input = Console.ReadLine().Split(' ').ToList();
        }
    }
}
```

За Java

```
package variant1;
```

```
import static java.lang.Character.toUpperCase;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;
public class Zad26 {

    public static void main(String[] args) {
        List<String> words = new ArrayList<>();
        Scanner input = new Scanner(System.in);
        String in[] = input.nextLine().split(" ");
        while (!in[0].equals("END")) {
            switch (in[0]) {
                case "Add":
                    for (int i = 1; i < in.length; i++) {
                        words.add(in[i]);
                    }
                    break;
                case "Remove":
                    words.remove(Integer.parseInt(in[1]));
                    break;
                case "Search":
                    if (!words.contains(in[1])) {
                        System.out.println("Not contained.");
                    } else {
                        System.out.println(in[1]);
                    }
```

```
        break;
    case "Update":
        int j = 0;
        for (String w : words) {
            char firstLetter = toUpperCase(w.charAt(0));
            String lastLetters = w.substring(1);
            w = firstLetter + lastLetters;
            words.set(j, w);
            j++;
        }
        break;
    case "Length":
        int length = Integer.parseInt(in[1]);
        List<String> wordsByLenght =
            (ArrayList<String>) words.stream().
            filter(x -> x.length() == length).
            collect(Collectors.toList());
        System.out.println(String.join("-",
            wordsByLenght));
        break;
    case "Insert":
        try {
        words.add(Integer.parseInt(in[1]), in[2]);
    } catch (IndexOutOfBoundsException ex) {
        System.out.println("There are not enough items
            in the list.");
    }
    break;
    case "Print":
```

```
                    System.out.println(String.join("; ", words));
            }
            in = input.nextLine().split(" ");
        }
    }
}
```

Задача 27: Примерно решение – 15 точки

```
DROP DATABASE IF EXISTS University;
CREATE DATABASE IF NOT EXISTS University;

USE University;
CREATE TABLE Student (
  admission_no int PRIMARY KEY,
  first_name varchar(25) NOT NULL,
  last_name varchar(25) NOT NULL,
  city varchar(25) NOT NULL
);
CREATE TABLE Fee (
  admission_no int NOT NULL,
  course varchar(25) NOT NULL,
  amount_paid int
  );
INSERT INTO Student (admission_no, first_name, last_name, city)
VALUES (3354, 'Георги', 'Георгиев', 'Варна'),
(4321, 'Милена', 'Красимирова', 'Стара Загора'),
(8345, 'Михаил', 'Мартинов', 'Пловдив'),
(7555, 'Антонио', 'Тачев', 'Стара Загора'),
```

(2135, 'Мартин', 'Иванов', 'София');

INSERT INTO Fee (admission_no, course, amount_paid)

VALUES (3354, 'Java', 2000),

(7555, 'C#', 1800),

(4321, 'SQL', 1600),

(4321, 'Java', 2000),

(8345, 'C++', 1700);

SELECT city FROM student WHERE admission_no = 8345;

SELECT Avg(amount_paid) FROM fee;

UPDATE fee SET course = 'Java' WHERE amount_paid = 1800;

SELECT first_name, last_name, course

FROM student

LEFT JOIN fee ON student.admission_no = fee.admission_no;

SELECT course, COUNT(course)

FROM fee

GROUP BY course;

Задача 28: Примерно решение – 20 точки

За C#

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

namespace DZI
{
    class Zad28
    {
        static void Main(string[] args)
        {
            List<Furniture> furnitures = new List<Furniture>();
```

```
            var input = Console.ReadLine().Split(' ').ToList();
            while (input[0] != "END")
            {
                if (input[0] == "table")
                {
                    Table table = new Table(input[0],
                          double.Parse(input[1]));
                    furnitures.Add(table);
                }
                if (input[0] == "cabinet")
                {
                    Cabinet cabinet = new Cabinet(input[0],
                       double.Parse(input[1]), int.Parse(input[2]));
                    furnitures.Add(cabinet);
                }

                input = Console.ReadLine().Split(' ').ToList();
            }

            Console.WriteLine("All tables:");
            foreach (Furniture furniture in furnitures)
            {
                if (furniture is Table)
                    Console.WriteLine(furniture.ToString());
            }

            Console.WriteLine("All cabinets:");
            foreach (Furniture furniture in furnitures)
            {
                if (furniture is Cabinet)
                    Console.WriteLine(furniture.ToString());
            }
        }
    }
}
namespace DZI
{
    public class Cabinet : Furniture
    {
        private int numberOfhinges;
```

```
        public int NumberOfHinges
        {
            get { return numberOfhinges; }
            set { numberOfhinges = value; }
        }
        public Cabinet(string type, double price, int
numberofhinges) : base(type, price)
        {
            this.NumberOfHinges = numberofhinges;
        }
        public override double PriceClient()
        {
           return base.ProductionPrice * 1.15 + this.NumberOfHinges
* 4.50;
        }
        public override string ToString()
        {
            return $"The cabinet costs {PriceClient():f2}  lv.";
        }
    }
}
namespace DZI
{
    public abstract class Furniture
    {
        private string typeProduct;
        private double productionPrice;
        public string TypeProduct
        {
            get { return typeProduct; }
```

```
            set
            {
                if (string.IsNullOrEmpty(value))
                    throw new Exception("Type not empty.");
                else typeProduct = value;
            }
        }
        public double ProductionPrice
        {
            get { return productionPrice; }
            set
            {
                if (value <= 0)
                    throw new Exception("Price not negative.");
                else productionPrice = value;
            }
        }
namespace DZI
{
    public class Table : Furniture
    {
        public Table(string type, double price) : base(type, price)
        {
        }
        public override double PriceClient()
        {
            return base.ProductionPrice * 1.20;
        }
        public override string ToString()
        {
```

```
            return $"The table costs {PriceClient():f2}  lv.";
        }
    }
}
```

За Java

```
public class Zad28 {
    public static void main(String[] args) {
        ArrayList<Furniture> list = new ArrayList<>();
        Scanner scan = new Scanner(System.in);
        String in[] = scan.nextLine().split(" ");
        while (in.length != 1) {
            if(in.length == 3)
                list.add(new Cabinet(in[0],
Double.parseDouble(in[1]), Integer.parseInt(in[2])));
            else
                list.add(new Table(in[0],
Double.parseDouble(in[1])));
            in = scan.nextLine().split(" ");
        }
                System.out.println("All tables:");
        for(Furniture furniture : list) {
            if(furniture.getTypeProduct().equals("table"))
                System.out.println(furniture.toString());
        }
                System.out.println("All cabinets:");
        for(Furniture furniture : list) {
            if(furniture.getTypeProduct().equals("cabinet"))
                System.out.println(furniture.toString());
        }
    }
```

```
}
public class Cabinet extends Furniture {
    private int numberOfHinges;
    public int getNumberOfHinges() {
        return numberOfHinges;
    }
    public void setNumberOfHinges(int numberOfHinges) {
        this.numberOfHinges = numberOfHinges;
    }
    @Override
    public double priceClient() {
        return 1.15*this.getProductionPrice() + numberOfHinges*4.5;
    }
    @Override
    public String toString() {
        return String.format("The cabinet costs %.2f lv.",
this.priceClient());
    }
    public Cabinet(String typeProduct, double productionPrice, int
numberOfHinges) {
        super(typeProduct, productionPrice);
        this.numberOfHinges = numberOfHinges;
    }
}
public abstract class Furniture {
    private String typeProduct;
    private double productionPrice;
    public String getTypeProduct() {
        return typeProduct;
    }
```

```
    public void setTypeProduct(String typeProduct) {
        if(!typeProduct.equals(""))
            this.typeProduct = typeProduct;
    }
    public double getProductionPrice() {
        return productionPrice;
    }
    public void setProductionPrice(double ProductionPrice) {
        if(ProductionPrice > 0)
            this.productionPrice = ProductionPrice;
    }
public class Table extends Furniture {
    public Table(String typeProduct, double ProductionPrice) {
        super(typeProduct, ProductionPrice);
    }
    @Override
    public double priceClient() {
        return 1.2*this.getProductionPrice();
    }
    @Override
    public String toString() {
        return String.format("The table costs %.2f lv.",
this.priceClient());
    }
}
```