

ОТГОВОРИ, УПЪТВАНИЯ И ПРИМЕРНИ ИЗГЛЕДИ
НА РЕШЕНИЯТА НА ПРАКТИЧЕСКИТЕ
ЗАДАЧИ ОТ

ТЕМА 6

Въпрос	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Отговор	А	Б	В	Б	В	Б	Б	Б	Б	В	В	Б	Г	А	В	А

17.

C#	Java
Even numbers up to 5: 2 4	

18.

C#
<pre>using System; class MyProgram { public static void Main() { int n = 10, sum = 0; while (n > 0) { sum += n; n -= 1; } Console.WriteLine("Сумата е " + sum); } }</pre>
Java
<pre>public class MyProgram { public static void main(String[] args) { int n = 10, sum = 0; while (n > 0) { sum += n; n -= 1; } System.out.println("Сумата е " + sum); } }</pre>

19.

C#
<pre>using System; class Program { static void Main() { int a, b;</pre>

```

        Console.WriteLine("Въведете a: ");
        a = int.Parse(Console.ReadLine());
        Console.WriteLine("Въведете b: ");          b = int.
Parse(Console.ReadLine());
        int temp = a;
        a = b;
        b = temp;
        Console.WriteLine("След размяната: a=" + a + " b=" + b);
    }
}

```

Java

```

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Въведете a: ");
        int a = scanner.nextInt();
        System.out.print("Въведете b: ");
        int b = scanner.nextInt();
        int temp = a;
        a = b;
        b = temp;
        System.out.println("След размяната: a=" + a + " b=" + b);
    }
}

```

20. 6

21.

teacher	course	class
Иванов	информатика	11
Петрова	математика	10
Петрова	математика	11
Петрова	математика	12
Христова	Технологии	10
Каменова	литература	NULL

22. A

23.

Exam_Title	AVG(Result)
Английски език	5.25

24. Възможен отговор: Методът **SumOfSquares** пресмята сумата от квадратите на всички цели числа в интервала $[a, b]$, включително. Резултатът е общата сума от квадратите на числата в този интервал.

25. Възможно решение

C#

```
using System;
// Базов клас за фигура
public abstract class Shape
{
    // Абстрактен метод за изчисление на площта
    public abstract double CalculateArea();
}
// Клас за кръг
public class Circle : Shape
{
    private double radius;
    public Circle(double radius)
    {
        this.radius = radius;
    }
    // Предефиниране на метода за изчисление на площта за кръга
    public override double CalculateArea()
    {
        return Math.PI * radius * radius;
    }
}
// Клас за правоъгълник
public class Rectangle : Shape
{
    private double length;
    private double width;
    public Rectangle(double length, double width)
    {
        this.length = length;
        this.width = width;
    }
    // Предефиниране на метода за изчисление на площта за
    // правоъгълника
    public override double CalculateArea()
    {
        return length * width;
    }
}
// Клас за триъгълник
public class Triangle : Shape
{
    private double sideA;
    private double sideB;
    private double sideC;
    public Triangle(double sideA, double sideB, double sideC)
    {
        this.sideA = sideA;
        this.sideB = sideB;
        this.sideC = sideC;
    }
    // Предефиниране на метода за изчисление на площта за
    // триъгълника
    public override double CalculateArea()
    {
```

```

    }
    double p = (sideA + sideB + sideC) / 2;
    return Math.Sqrt(p * (p - sideA) * (p - sideB) * (p -
sideC));
} class Program
{
    static void Main()
    {
        Circle circle = new Circle(5);
        Rectangle rectangle = new Rectangle(4, 6);
        Triangle triangle = new Triangle(3, 4, 5);
        Console.WriteLine("Circle Area: " + circle.
CalculateArea());
        Console.WriteLine("Rectangle Area: " + rectangle.
CalculateArea());
        Console.WriteLine("Triangle Area: " + triangle.
CalculateArea());
    }
}

```

Java

```

// Main.java
public class Main {
    public static void main(String[] args) {
        Circle circle = new Circle(5);
        Rectangle rectangle = new Rectangle(4, 6);
        Triangle triangle = new Triangle(3, 4, 5);
        System.out.println("Circle Area: " + circle.
calculateArea());
        System.out.println("Rectangle Area: " + rectangle.
calculateArea());
        System.out.println("Triangle Area: " + triangle.
calculateArea());
    }
}
abstract class Shape {
    public abstract double calculateArea();
}
class Circle extends Shape {
    private double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;
    }
}
class Rectangle extends Shape {
    private double length;

```

```

    private double width;
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    @Override
    public double calculateArea() {
        return length * width;
    }
}
class Triangle extends Shape {
    private double sideA;
    private double sideB;
    private double sideC;
    public Triangle(double sideA, double sideB, double sideC) {
        this.sideA = sideA;
        this.sideB = sideB;
        this.sideC = sideC;
    }
    @Override
    public double calculateArea() {
        double p = (sideA + sideB + sideC) / 2;
        return Math.sqrt(p * (p - sideA) * (p - sideB) * (p -
sideC));
    }
}

```

26. Създаване на таблица **Department**

```

CREATE TABLE Departments (
    ID CHAR(3) PRIMARY KEY,
    Department_Name VARCHAR(100)
);

```

Създаване на таблица **Students**

```

CREATE TABLE Students (
    Id VARCHAR(6) PRIMARY KEY,
    Student_name VARCHAR(100),
    Department_Id CHAR(3),
    Results DECIMAL(5,2),
    Birth_date DATE,
    FOREIGN KEY (Department_Id) REFERENCES Departments(ID)
);

```

Добавяне на записи в таблиците

– За таблица **Departments**

```

INSERT INTO Departments (ID, Department_Name) VALUES
(,001', 'Компютърни системи и технологии'),
(,002', 'Информационни и комуникационни технологии');

```

– За таблица **Students**

```
INSERT INTO Students (Id, Student_name, Department_Id, Results, Birth_date)
VALUES
(,233023', 'Иван Петров Иванов', '001', 33.50, '2005-12-23'),
(,233043', 'Алекс Драгомиров Колев', '001', 36, '2005-03-02'),
(,233046', 'Ивайла Стоименова Николова', '002', 30.25, '2005-06-24');
```

Отговор: Напишете заявка, която да изведе информация за имената на студентите, специалността, за която кандидатстват, и бала им, сортирани по намаляващ ред на бала.

```
SELECT S.Student_name, D.Department_Name, S.Results
FROM Students S
JOIN Departments D ON S.Department_Id = D.ID
ORDER BY S.Results DESC;
```

Отговор: Напишете заявка, която да изведе имената на студентите с най-висок бал.

```
SELECT TOP 1 Student_name, Results
FROM Students
ORDER BY Results DESC;
```

Отговор: Напишете заявка, която да връща информация за имената на специалностите и броя на кандидатите във всяка от тях.

```
SELECT D.Department_Name, COUNT(*) AS NumberOfStudents
FROM Students S
JOIN Departments D ON S.Department_Id = D.ID
GROUP BY D.Department_Name;
```

27.

C#
<pre>using System; using System.Collections.Generic; class Asset { public int AssetId { get; set; } public string Name { get; set; } public double InitialCost { get; set; } public double ResidualValue { get; set; } public int UsefulLife { get; set; } } class DepreciationCalculator { private List<Asset> assets; public DepreciationCalculator() { assets = new List<Asset>(); } }</pre>

```

    public void AddAsset(Asset asset)
    {
        assets.Add(asset);
    }
    public void CalculateDepreciation()
    {
        foreach (var asset in assets)
        {
            double depreciationAmount = (asset.InitialCost -
asset.ResidualValue) / asset.UsefulLife;
            double accumulatedDepreciation = 0;
            Console.WriteLine($"AssetID: {asset.AssetId}, Name:
{asset.Name}, Initial Cost: {asset.InitialCost:C2}, " +
            $"Residual Value: {asset.ResidualValue:C2},
Useful Life: {asset.UsefulLife}");
            Console.WriteLine($"Depreciation:
{depreciationAmount:C2} per year");
            for (int year = 1; year <= asset.UsefulLife; year++)
            {
                accumulatedDepreciation += depreciationAmount;
                Console.WriteLine($"Year {year}: Accumulated
Depreciation: {accumulatedDepreciation:C2}");
            }
            Console.WriteLine();
        }
    }
}
class Program
{
    static void Main()
    {
        DepreciationCalculator calculator = new
DepreciationCalculator();
        Console.Write("Enter the number of assets: ");
        int n = int.Parse(Console.ReadLine());
        for (int i = 0; i < n; i++)
        {
            Console.WriteLine($"Enter information for asset {i +
1}:");
            string[] input = Console.ReadLine().Split(" ");
            Asset asset = new Asset();
            asset.AssetId = int.Parse(input[0]);
            asset.Name = input[1];
            asset.InitialCost = double.Parse(input[2]);
            asset.ResidualValue = double.Parse(input[3]);
            asset.UsefulLife = int.Parse(input[4]);
            calculator.AddAsset(asset);
        }
        calculator.CalculateDepreciation();
    }
}

```

Java

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class DepreciationManagement {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Четене на броя на дълготрайните активи
        System.out.print("Enter the number of assets: ");
        int n = Integer.parseInt(scanner.nextLine());
        // Създаване на инстанция на DepreciationCalculator
        DepreciationCalculator depreciationCalculator = new
        DepreciationCalculator();
        // Четене на информацията за активите и добавяне към
        DepreciationCalculator
        for (int i = 0; i < n; i++) {
            System.out.println("Enter information for asset " +
            (i + 1) + ":");
            String[] assetInfo = scanner.nextLine().split(" ");
            int assetId = Integer.parseInt(assetInfo[0]);
            String name = assetInfo[1];
            double initialCost = Double.
            parseDouble(assetInfo[2]);
            double residualValue = Double.
            parseDouble(assetInfo[3]);
            int usefulLife = Integer.parseInt(assetInfo[4]);
            Asset asset = new Asset(assetId, name, initialCost,
            residualValue, usefulLife);
            depreciationCalculator.addAsset(asset);
        }
        // Изчисляване на амортизацията и извеждане на
        информацията
        depreciationCalculator.calculateDepreciation();
        depreciationCalculator.displayDepreciationInfo();
    }
}
class Asset {
    private int assetId;
    private String name;
    private double initialCost;
    private double residualValue;
    private int usefulLife;
    private double annualDepreciation;
    private double accumulatedDepreciation;
    public Asset(int assetId, String name, double initialCost,
    double residualValue, int usefulLife) {
        this.assetId = assetId;
        this.name = name;
        this.initialCost = initialCost;
        this.residualValue = residualValue;
```



```

        this.usefulLife = usefulLife;
    }
    public double getAnnualDepreciation() {
        return annualDepreciation;
    }
    public void setAnnualDepreciation(double annualDepreciation)
{
        this.annualDepreciation = annualDepreciation;
    }
    public double getAccumulatedDepreciation() {
        return accumulatedDepreciation;
    }
    public void setAccumulatedDepreciation(double
accumulatedDepreciation) {
        this.accumulatedDepreciation = accumulatedDepreciation;
    }
    public int getAssetId() {
        return assetId;
    }
    public String getName() {
        return name;
    }
    public double getInitialCost() {
        return initialCost;
    }
    public double getResidualValue() {
        return residualValue;
    }
    public int getUsefulLife() {
        return usefulLife;
    }
}
class DepreciationCalculator {
    private List<Asset> assets;
    public DepreciationCalculator() {
        assets = new ArrayList<>();
    }
    public void addAsset(Asset asset) {
        assets.add(asset);
    }
    public void calculateDepreciation() {
        for (Asset asset : assets) {
            double annualDepreciation = (asset.getInitialCost() -
asset.getResidualValue()) / asset.getUsefulLife();
            asset.setAnnualDepreciation(annualDepreciation);
            double accumulatedDepreciation = 0;
            System.out.println("AssetID: " + asset.getAssetId() +
                                ", Name: " + asset.getName() +
                                ", Initial Cost: ?" + String.format("%.2f",
asset.getInitialCost()) +

```

```

        ", Residual Value: ?" + String.format("%.2f",
asset.getResidualValue()) +
        ", Useful Life: " + asset.getUsefulLife());
        for (int year = 1; year <= asset.getUsefulLife();
year++) {
            accumulatedDepreciation += annualDepreciation;
            System.out.println("Year " + year + ":
Accumulated Depreciation: ?" + String.format("%.2f",
accumulatedDepreciation));
        }
        asset.setAccumulatedDepreciation(accumulatedDepreciat
ion);
    }
}
public void displayDepreciationInfo() {
    for (Asset asset : assets) {
        System.out.println("Depreciation: ?" + String.
format("%.2f", asset.getAnnualDepreciation()) + " per year");
        System.out.println();
    }
}
}

```

28.

C#

```

using System;
using System.Collections.Generic;
class Person
{
    public string name;
    public int age;
    public Person(string name, int age)
    {
        this.name = name;
        this.age = age;
    }
}
class Animal
{
    public string name;
    public int age;
    public Animal(string name, int age)
    {
        this.name = name;
        this.age = age;
    }
    public override string ToString()
    {
        return $"{name},{age}";
    }
}

```

```

    }
}
class Zookeeper : Person
{
    public Animal favoriteAnimal;
    public string specialization;
    public Zookeeper(string name, int age, Animal favoriteAnimal,
string specialization)
        : base(name, age)
    {
        this.favoriteAnimal = favoriteAnimal;
        this.specialization = specialization;
    }
    public override string ToString()
    {
        return $"{name},{age},{specialization},{favoriteAnimal}";
    }
}
class Zoo
{
    public string name;
    public string location;
    public List<Zookeeper> zookeepers;
    public Zoo(string name, string location)
    {
        this.name = name;
        this.location = location;
        this.zookeepers = new List<Zookeeper>();
    }
    public void AddZookeeper(Zookeeper zookeeper)
    {
        zookeepers.Add(zookeeper);
    }
    public void ViewZoo()
    {
        Console.WriteLine($"Zoo: {name} - {location}");
        foreach (var zookeeper in zookeepers)
        {
            Console.WriteLine(zookeeper);
        }
    }
}
class Program
{
    static void Main()
    {
        Console.WriteLine("Enter Zoo name:");
        string zooName = Console.ReadLine();
        Console.WriteLine("Enter Zoo location:");
        string zooLocation = Console.ReadLine();
    }
}

```

```

        Zoo zoo = new Zoo(zooName, zooLocation);
        char option;
        do
        {
            Console.WriteLine("[a]dd [v]iew [q]uit");
            option = Console.ReadLine()[0];
            switch (option)
            {
                case ,a':
                    Console.WriteLine("Enter zookeeper name:");
                    string zkName = Console.ReadLine();
                    Console.WriteLine("Enter zookeeper age:");
                    int zkAge = int.Parse(Console.ReadLine());
                    Console.WriteLine("Enter specialization:");
                    string specialization = Console.ReadLine();
                    Console.WriteLine("Enter favorite animal
name:");
                    string favAnimalName = Console.ReadLine();
                    Console.WriteLine("Enter favorite animal
age:");
                    int favAnimalAge = int.Parse(Console.
ReadLine());
                    Animal favoriteAnimal = new
Animal(favAnimalName, favAnimalAge);
                    Zookeeper zookeeper = new Zookeeper(zkName,
zkAge, favoriteAnimal, specialization);
                    zoo.AddZookeeper(zookeeper);
                    break;
                case ,v':
                    zoo.ViewZoo();
                    break;
                case ,q':
                    Console.WriteLine("Exiting the program.");
                    break;
                default:
                    Console.WriteLine("Invalid option. Try
again.");
                    break;
            }
        } while (option != ,q');
    }
}

```

Java

```

import java.util.ArrayList;
import java.util.Scanner;
class Animal {
    String name;
    int age;
    public Animal(String name, int age) {

```

```

        this.name = name;
        this.age = age;
    }
    @Override
    public String toString() {
        return name + "," + age;
    }
}
class Person {
    String name;
    int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    @Override
    public String toString() {
        return name + "," + age;
    }
}
class Zookeeper extends Person {
    String specialization;
    Animal favoriteAnimal;
    public Zookeeper(String name, int age, String specialization,
Animal favoriteAnimal) {
        super(name, age);
        this.specialization = specialization;
        this.favoriteAnimal = favoriteAnimal;
    }
    @Override
    public String toString() {
        return super.toString() + "," + specialization + "," +
favoriteAnimal.toString();
    }
}
class Zoo {
    String name;
    String location;
    ArrayList<Zookeeper> zookeepers;
    public Zoo(String name, String location) {
        this.name = name;
        this.location = location;
        this.zookeepers = new ArrayList<>();
    }
    public void addZookeeper(Zookeeper zookeeper) {
        zookeepers.add(zookeeper);
    }
    public void displayInfo() {
        System.out.println("Zoo: " + name + " - " + location);
    }
}

```

```

        for (Zookeeper zookeeper : zookeepers) {
            System.out.println(zookeeper.toString());
        }
    }
}

class ZooApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter Zoo name:");
        String zooName = scanner.nextLine();
        System.out.println("Enter Zoo location:");
        String zooLocation = scanner.nextLine();
        Zoo zoo = new Zoo(zooName, zooLocation);
        while (true) {
            System.out.println("[a]dd [v]iew [q]uit");
            String option = scanner.nextLine();
            switch (option.toLowerCase()) {
                case "a":
                    System.out.println("Enter zookeeper name:");
                    String zkName = scanner.nextLine();
                    System.out.println("Enter zookeeper age:");
                    int zkAge = Integer.parseInt(scanner.
nextLine());

                    System.out.println("Enter specialization:");
                    String specialization = scanner.nextLine();
                    System.out.println("Enter favorite animal
name:");

                    String favAnimalName = scanner.nextLine();
                    System.out.println("Enter favorite animal
age:");

                    int favAnimalAge = Integer.parseInt(scanner.
nextLine());

                    Animal favAnimal = new Animal(favAnimalName,
favAnimalAge);

                    Zookeeper zookeeper = new Zookeeper(zkName,
zkAge, specialization, favAnimal);
                    zoo.addZookeeper(zookeeper);
                    break;
                case "v":
                    zoo.displayInfo();
                    break;
                case "q":
                    scanner.close();
                    System.exit(0);
                    break;
                default:
                    System.out.println("Invalid option. Try
again.");
            }
        }
    }
}

```