

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

**ТЕМА 9**

|                |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |
|----------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| <b>Въпрос</b>  | <b>1</b> | <b>2</b> | <b>3</b> | <b>4</b> | <b>5</b> | <b>6</b> | <b>7</b> | <b>8</b> | <b>9</b> | <b>10</b> | <b>11</b> | <b>12</b> | <b>13</b> | <b>14</b> | <b>15</b> | <b>16</b> |
| <b>Отговор</b> | <b>А</b> | <b>В</b> | <b>А</b> | <b>Б</b> | <b>В</b> | <b>В</b> | <b>Б</b> | <b>А</b> | <b>А</b> | <b>Б</b>  | <b>Б</b>  | <b>В</b>  | <b>Б</b>  | <b>Г</b>  | <b>В</b>  | <b>В</b>  |

**17.**

| <b>ID</b> | <b>  Department_Name</b>                  | <b>AVG(Results)</b> |
|-----------|---|---------------------|
| 001       | Компютърни системи и технологии           | 34.75               |
| 002       | Информационни и комуникационни технологии | 30.25               |

**18.**

89 85 40 33 23 23 18 4 3

**19.**

|  |
|--|
| <b>C#</b>  |
| <pre>using System; public class Person {     public string Name = "Иван";     public override string ToString()     {         return Name;     }     static void Main()     {         Person person = new Person();         Console.WriteLine(person);     } }</pre> |
| <b>Java</b>  |
| <pre>public class Person {     public String Name = "Иван";     public String toString() {         return Name;     }     public static void main(String[] args) {         Person person = new Person();         System.out.println(person);     } }</pre>           |

20. В

21.

| teacher  | course      |
|----------|-------------|
| Иванов   | информатика |
| Каменова | литература  |

22.

|   |
|---|
| <b>C#</b>   |
| <pre>using System; class Program {     static void Main()     {         int[,] numbers = { { 2, 3 }, { 4, 5 } };         numbers[0, 1] = 6;          // Показване на променената матрица         for (int i = 0; i &lt; 2; i++)         {             for (int j = 0; j &lt; 2; j++)             {                 Console.Write(numbers[i, j] + " ");             }             Console.WriteLine();         }     } }</pre> |
| <b>Java</b>   |
| <pre>public class Main {     public static void main(String[] args) {         int[][] numbers = { { 2, 3 }, { 4, 5 } };         numbers[0][1] = 6;          // Показване на променената матрица         for (int i = 0; i &lt; 2; i++) {             for (int j = 0; j &lt; 2; j++) {                 System.out.print(numbers[i][j] + " ");             }             System.out.println();         }     } }</pre>          |

23. Б

24. А

## 25. Възможно решение

1.

```
CREATE TABLE Movies (  
ID INT PRIMARY KEY,  
Movie_Title VARCHAR(100) NOT NULL,  
Director VARCHAR(100) ,  
Number_of_purchases INT,  
Movie_date DATE);
```

2.

```
INSERT INTO Movies(ID, Movie_Title, Director, Number_of_purchases,  
Movie_date)  
VALUES (1, 'PUSS IN BOOTS: THE LAST WISH', 'Joel Crawford', 3567,  
'2022-12-20');  
INSERT INTO Movies(ID, Movie_Title, Director, Number_of_purchases,  
Movie_date)  
VALUES (2, 'AVATAR: THE WAY OF WATER', 'James Cameron', 6034,  
'2022-12-27');  
INSERT INTO Movies(ID, Movie_Title, Director, Number_of_purchases,  
Movie_date)  
VALUES (3, 'M3GAN', 'Gerard Johnstone', 609, '2023-01-11');  
INSERT INTO Movies(ID, Movie_Title, Director, Number_of_purchases,  
Movie_date)  
VALUES (4, 'AFTERSUN', 'Charlotte Wells', 102345, '2023-01-11');  
INSERT INTO Movies(ID, Movie_Title, Director, Number_of_purchases,  
Movie_date)  
VALUES (5, 'THE MENU', 'Mark Mylod', 105432, '2023-01-20');
```

3.

```
SELECT Movie_Title, Number_of_purchases  
FROM movies  
WHERE YEAR(Movie_date)=2022;
```

4.

```
SELECT COUNT(ID)  
FROM movies  
WHERE YEAR(Movie_date)=2023;
```

5.

```
SELECT Movie_Title, Number_of_purchases  
FROM movies  
WHERE Number_of_purchases>(SELECT AVG(Number_of_purchases) FROM  
movies);
```

6.

```
UPDATE movies  
SET Number_of_purchases=Number_of_purchases-10  
WHERE YEAR(Movie_date)=2023;
```

## 26. Възможно решение

```
C#
using System;
class Program
{
    static void Main()
    {
        try
        {
            Console.WriteLine("Въведете страни на триъгълника:");
            Console.Write("a = ");
            int a = int.Parse(Console.ReadLine());
            Console.Write("b = ");
            int b = int.Parse(Console.ReadLine());
            Console.Write("c = ");
            int c = int.Parse(Console.ReadLine());
            if (IsValidTriangle(a, b, c))
            {
                if (IsEquilateralTriangle(a, b, c))
                {
                    Console.WriteLine("Това е равностранен
триъгълник.");
                }
                else if (IsIsoscelesTriangle(a, b, c))
                {
                    Console.WriteLine("Това е равнобедрен триъгълник.");
                }
                else
                {
                    Console.WriteLine("Това е обикновен триъгълник.");
                }
            }
            else
            {
                Console.WriteLine("Това не е валиден триъгълник.");
            }
        }
        catch (Exception)
        {
            Console.WriteLine("Something went wrong!");
        }
    }

    static bool IsValidTriangle(int a, int b, int c)
    {
        return a + b > c && a + c > b && b + c > a;
    }

    static bool IsEquilateralTriangle(int a, int b, int c)
    {
        return a == b && b == c;
    }

    static bool IsIsoscelesTriangle(int a, int b, int c)
    {
        return a == b || b == c || a == c;
    }
}
```

## Java

```
import java.util.Scanner;
public class TriangleProgram {
    public static void main(String[] args) {
        try {
            Scanner scanner = new Scanner(System.in);
            System.out.println("Въведете страни на триъгълника:");
            System.out.print("a = ");
            int a = readSide(scanner.nextLine().replaceAll("\\D",
""));

            System.out.print("b = ");
            int b = readSide(scanner.nextLine());
            System.out.print("c = ");
            int c = readSide(scanner.nextLine());
            if (isValidTriangle(a, b, c)) {
                if (isEquilateralTriangle(a, b, c)) {
                    System.out.println("Това е равностранен триъгълник.");
                } else if (isIsoscelesTriangle(a, b, c)) {
                    System.out.println("Това е равнобедрен
триъгълник.");
                } else {
                    System.out.println("Това е обикновен
триъгълник.");
                }
            } else {
                System.out.println("Това не е валиден триъгълник.");
            }
        } catch (Exception e) {
            System.out.println("Something went wrong!");
        }
    }

    static int readSide(String input) {
        try {
            return Integer.parseInt(input);
        } catch (NumberFormatException e) {
            throw new IllegalArgumentException("Грешка при въвеждане
на страна на триъгълника.");
        }
    }

    static boolean isValidTriangle(int a, int b, int c) {
        return a + b > c && a + c > b && b + c > a;
    }

    static boolean isEquilateralTriangle(int a, int b, int c) {
        return a == b && b == c;
    }

    static boolean isIsoscelesTriangle(int a, int b, int c) {
        return a == b || b == c || a == c;
    }
}
```

27.

C#

```
using System;
using System.Collections.Generic;
class Product
{
    public int ID { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public int StockQuantity { get; set; }
    public void DisplayInfo()
    {
        Console.WriteLine($"{ID}. {Name} - {Price:C} -
{StockQuantity} in stock");
    }
}
class Customer
{
    public int ID { get; set; }
    public string Name { get; set; }
    public string Address { get; set; }
    public List<Product> PurchasedProducts { get; set; } = new
List<Product>();
    public void PlaceOrder(Product product)
    {
        if (product.StockQuantity > 0)
        {
            PurchasedProducts.Add(product);
            product.StockQuantity--;
            Console.WriteLine($"Order placed successfully. Total
amount: {product.Price:C}");
        }
        else
        {
            Console.WriteLine("Product out of stock.");
        }
    }
}
class Order
{
    public int ID { get; set; }
    public DateTime OrderDate { get; set; }
    public Customer Customer { get; set; }
    public List<Product> OrderedProducts { get; set; } = new
List<Product>();
}
class OnlineStore
{
    public List<Product> Products { get; set; } = new
List<Product>();
    public List<Customer> Customers { get; set; } = new
List<Customer>();
    public List<Order> Orders { get; set; } = new List<Order>();
    public void DisplayProductList()
```

```

    {
        Console.WriteLine("Available products:");
        foreach (Product product in Products)
        {
            product.DisplayInfo();
        }
    }
    public void DisplayCustomerPurchasedProducts(Customer
customer)
    {
        Console.WriteLine("Customer's Purchased Products:");
        foreach (Product purchasedProduct in customer.
PurchasedProducts)
        {
            Console.WriteLine(purchasedProduct.Name);
        }
    }
}
class Program
{
    static void Main()
    {
        OnlineStore myStore = new OnlineStore();
        Console.WriteLine("Enter product information for Laptop
(ID Name Price StockQuantity):");
        string[] laptopInfo = Console.ReadLine().Split();
        Product laptop = new Product { ID = int.
Parse(laptopInfo[0]), Name = laptopInfo[1], Price = decimal.
Parse(laptopInfo[2]), StockQuantity = int.Parse(laptopInfo[3]) };
        myStore.Products.Add(laptop);
        Console.WriteLine("Enter product information for
Smartphone (ID Name Price StockQuantity):");
        string[] smartphoneInfo = Console.ReadLine().Split();
        Product smartphone = new Product { ID = int.
Parse(smartphoneInfo[0]), Name = smartphoneInfo[1], Price
= decimal.Parse(smartphoneInfo[2]), StockQuantity = int.
Parse(smartphoneInfo[3]) };
        myStore.Products.Add(smartphone);
        Console.WriteLine("Enter customer name:");
        string customerName = Console.ReadLine();
        Console.WriteLine("Enter customer address:");
        string customerAddress = Console.ReadLine();
        Customer customer = new Customer { ID = 1, Name =
customerName, Address = customerAddress };
        myStore.Customers.Add(customer);
        myStore.DisplayProductList();
        Console.WriteLine("Enter the product ID you want to order:");
        int selectedProductID = int.Parse(Console.ReadLine());
        Product selectedProduct = myStore.Products.Find(p => p.ID
== selectedProductID);
    }
}

```

```

        if (selectedProduct != null)
        {
            customer.PlaceOrder(selectedProduct);
            myStore.DisplayCustomerPurchasedProducts(customer);
        }
        else
        {
            Console.WriteLine("Invalid product ID.");
        }
    }
}

```

## Java

```

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        OnlineStore myStore = new OnlineStore();
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter product information for Laptop
(ID Name Price StockQuantity):");
        String[] laptopInfo = scanner.nextLine().split(" ");
        Product laptop = new Product(Integer.
parseInt(laptopInfo[0]), laptopInfo[1], Double.
parseDouble(laptopInfo[2]), Integer.parseInt(laptopInfo[3]));
        myStore.Products.add(laptop);
        System.out.println("Enter product information for
Smartphone (ID Name Price StockQuantity):");
        String[] smartphoneInfo = scanner.nextLine().split(" ");
        Product smartphone = new Product(Integer.
parseInt(smartphoneInfo[0]), smartphoneInfo[1],
Double.parseDouble(smartphoneInfo[2]), Integer.
parseInt(smartphoneInfo[3]));
        myStore.Products.add(smartphone);
        System.out.println("Enter customer name:");
        String customerName = scanner.nextLine();
        System.out.println("Enter customer address:");
        String customerAddress = scanner.nextLine();
        Customer customer = new Customer();
        customer.ID = 1;
        customer.Name = customerName;
        customer.Address = customerAddress;
        myStore.Customers.add(customer);
        myStore.displayProductList();
        System.out.println("Enter the product ID you want to
order:");
    }
}

```



```

        int selectedProductID = scanner.nextInt();
        Product selectedProduct = null;
        for (Product product : myStore.Products) {
            if (product.ID == selectedProductID) {
                selectedProduct = product;
                break;
            }
        }
        if (selectedProduct != null) {
            customer.placeOrder(selectedProduct);
            myStore.displayCustomerPurchasedProducts(customer);
        } else {
            System.out.println("Invalid product ID.");
        }
    }
}

class Product {
    int ID;
    String Name;
    double Price;
    int StockQuantity;
    public Product(int ID, String Name, double Price, int
StockQuantity) {
        this.ID = ID;
        this.Name = Name;
        this.Price = Price;
        this.StockQuantity = StockQuantity;
    }
    void displayInfo() {
        System.out.println(ID + ". " + Name + " - $" + Price + "
- " + StockQuantity + " in stock");
    }
}

class Customer {
    int ID;
    String Name;
    String Address;
    List<Product> PurchasedProducts = new ArrayList<>();
    void placeOrder(Product product) {
        if (product.StockQuantity > 0) {
            PurchasedProducts.add(product);
            product.StockQuantity--;
            System.out.println("Order placed successfully. Total
amount: $" + product.Price);
        } else {
            System.out.println("Product out of stock.");
        }
    }
}

```

```

    }
}
class Order {
    int ID;
    String OrderDate;
    Customer Customer;
    List<Product> OrderedProducts = new ArrayList<>();
}
class OnlineStore {
    List<Product> Products = new ArrayList<>();
    List<Customer> Customers = new ArrayList<>();
    List<Order> Orders = new ArrayList<>();
    void displayProductList() {
        System.out.println("Available products:");
        for (Product product : Products) {
            product.displayInfo();
        }
    }
    void displayCustomerPurchasedProducts(Customer customer) {
        System.out.println("Customer's Purchased Products:");
        for (Product purchasedProduct : customer.
PurchasedProducts) {
            System.out.println(purchasedProduct.Name);
        }
    }
}
}

```

## 28. Примерно решение

**C#**

```
using System;
using System.Collections.Generic;
class Course
{
    private string title;
    private int duration;
    private decimal price;
    public Course(string title, int duration, decimal price)
    {
        Title = title;
        Duration = duration;
        Price = price;
    }
    public string Title
    {
        get { return title; }
        set
        {
            if (string.IsNullOrEmpty(value))
            {
                throw new ArgumentException("Invalid course
title!");
            }
            title = value;
        }
    }
    public int Duration
    {
        get { return duration; }
        set
        {
            if (value <= 0)
            {
                throw new ArgumentException("Duration must be a
positive number!");
            }
            duration = value;
        }
    }
    public decimal Price
    {
        get { return price; }
        set
        {
```

```

        if (value <= 0)
        {
            throw new ArgumentException("Price must be a
positive number!");
        }
        price = value;
    }
}
class Student
{
    private string name;
    private List<Course> courses;
    public Student(string name)
    {
        Name = name;
        courses = new List<Course>();
    }
    public string Name
    {
        get { return name; }
        set
        {
            if (string.IsNullOrEmpty(value))
            {
                throw new ArgumentException("Invalid student
name!");
            }
            name = value;
        }
    }
    public List<Course> Courses { get { return courses; } }
    public void Enroll(Course course)
    {
        courses.Add(course);
    }
}
class University
{
    private List<Course> courses;
    private List<Student> students;
    public University()
    {
        courses = new List<Course>();
        students = new List<Student>();
    }
}

```

```

public List<Course> Courses { get { return courses; } }
public List<Student> Students { get { return students; } }
public void AddCourse(Course course)
{
    courses.Add(course);
}
public void RegisterStudent(Student student)
{
    students.Add(student);
}
public void DisplayCourses()
{
    Console.WriteLine("All Courses:");
    foreach (var course in courses)
    {
        Console.WriteLine($"Title: {course.Title}, Duration:
{course.Duration} hours, Price: {course.Price:C}");
    }
}
public void DisplayStudents()
{
    Console.WriteLine("All Students:");
    foreach (var student in students)
    {
        Console.WriteLine($"Name: {student.Name}");
        Console.WriteLine("Enrolled Courses:");
        foreach (var course in student.Courses)
        {
            Console.WriteLine($"- {course.Title}");
        }
    }
}
}
class Program
{
    static void Main()
    {
        University university = new University();
        // Adding courses
        university.AddCourse(new Course("Java Programming", 40,
150));
        university.AddCourse(new Course("Web Development", 30,
120));
        // Registering students
        Student student1 = new Student("Todor");
        student1.Enroll(university.Courses[0]);
    }
}

```

```

        student1.Enroll(university.Courses[1]);
        Student student2 = new Student("Ivan");
        student2.Enroll(university.Courses[1]);
        university.RegisterStudent(student1);
        university.RegisterStudent(student2);
        // Displaying information about courses and students
        university.DisplayCourses();
        university.DisplayStudents();
    }
}

```

#### Java

```

import java.util.ArrayList;
import java.util.List;
class Course {
    private String title;
    private int duration;
    private double price;
    public Course(String title, int duration, double price) {
        setTitle(title);
        setDuration(duration);
        setPrice(price);
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        if (title == null || title.trim().isEmpty()) {
            throw new IllegalArgumentException("Invalid course
title!");
        }
        this.title = title;
    }
    public int getDuration() {
        return duration;
    }
    public void setDuration(int duration) {
        if (duration <= 0) {
            throw new IllegalArgumentException("Duration must be
a positive number!");
        }
        this.duration = duration;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {

```

```

        if (price <= 0) {
            throw new IllegalArgumentException("Price must be a
positive number!");
        }
        this.price = price;
    }
}
class Student {
    private String name;
    private List<Course> courses;
    public Student(String name) {
        setName(name);
        courses = new ArrayList<>();
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        if (name == null || name.trim().isEmpty()) {
            throw new IllegalArgumentException("Invalid student
name!");
        }
        this.name = name;
    }
    public List<Course> getCourses() {
        return courses;
    }
    public void enroll(Course course) {
        courses.add(course);
    }
}
class University {
    private List<Course> courses;
    private List<Student> students;
    public University() {
        courses = new ArrayList<>();
        students = new ArrayList<>();
    }
    public List<Course> getCourses() {
        return courses;
    }
    public List<Student> getStudents() {
        return students;
    }
    public void addCourse(Course course) {
        courses.add(course);
    }
}

```

```

        public void registerStudent(Student student) {
            students.add(student);
        }
        public void displayCourses() {
            System.out.println("All Courses:");
            for (Course course : courses) {
                System.out.printf("Title: %s, Duration: %d hours,
Price: %.2f%n",
                                course.getTitle(), course.getDuration(),
course.getPrice());
            }
        }
        public void displayStudents() {
            System.out.println("All Students:");
            for (Student student : students) {
                System.out.println("Name: " + student.getName());
                System.out.println("Enrolled Courses:");
                for (Course course : student.getCourses()) {
                    System.out.println("- " + course.getTitle());
                }
            }
        }
    }
}

class Main {
    public static void main(String[] args) {
        University university = new University();
        // Adding courses
        university.addCourse(new Course("Java Programming", 40,
150.0));
        university.addCourse(new Course("Web Development", 30,
120.0));
        // Registering students
        Student student1 = new Student("Todor");
        student1.enroll(university.getCourses().get(0));
        student1.enroll(university.getCourses().get(1));
        Student student2 = new Student("Ivan");
        student2.enroll(university.getCourses().get(1));
        university.registerStudent(student1);
        university.registerStudent(student2);
        // Displaying information about courses and students
        university.displayCourses();
        university.displayStudents();
    }
}

```