МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ ПО

ИНФОРМАТИКА

23 май 2025 г.

ПРОФИЛИРАНА ПОДГОТОВКА

ВАРИАНТ 2

ЧАСТ 1 (Време за работа: 90 минути)

Внимание! Полетата за отговори НЕ трябва да съдържат текстове или символи, които могат да доведат до нарушаване на анонимността на изпитната Ви работа!

За задачите от 1. до 16. включително изберете точно един от отговорите!

1. Даден е следния програмен фрагмент:

C#

```
class Engine
{
    private int power;
    public Engine(int power)
    {
        this.power = power;
    }
}
class Vehicle
{
    private Engine engine;
    public Vehicle(Engine engine)
    {
        this.engine = engine;
    }
}
class Car : Vehicle
{
    public Car(Engine engine) : base(engine) { }
}
```

Java

```
class Engine {
    private int power;
    public Engine(int power) {
        this.power = power;
    }
}
class Vehicle {
    private Engine engine;
    public Vehicle(Engine engine) {
        this.engine = engine;
    }
}
class Car extends Vehicle {
    public Car(Engine engine) {
        super(engine);
    }
}
```

Кое от следните твърдения е вярно за връзките между класовете в дадения код?

А) `Car` "има" (`has a`) `Vehicle` и `Vehicle` "има" (`has a`) `Engine`

Б) `Car` "е" (`is a`) `Vehicle` и `Vehicle` "е" (`is a`) `Engine`

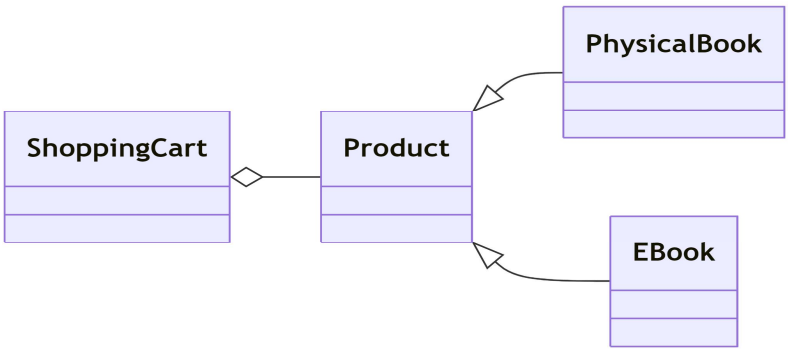
В) `Car` "е" (`is a`) `Vehicle` и `Vehicle` "има" (`has a`) `Engine`

Г) `Car` "има" (`has a`) `Engine` и `Vehicle` "има" (`has a`) `Engine`

2. Дадена е UML диаграма на система за онлайн книжарница, където:

- `Product` е базов клас за всички продукти
- `PhysicalBook` и `EBook` са видове продукти
- `ShoppingCart` е количка за пазаруване, която може да съдържа различни продукти

Кое от следните твърдения правилно описва използването на принципите на наследяване и композиция/агрегация в системата?



А) ShoppingCart наследява Product, а PhysicalBook и EBook са композиция на количката за пазаруване ShoppingCart

Б) PhysicalBook и EBook са наследници на Product, а ShoppingCart използва агрегация, за да съхранява списък от продукти

В) ShoppingCart е наследник на Product, който може да съдържа само един вид продукт (или само PhysicalBook, или само EBook)

Г) PhysicalBook наследява EBook, а ShoppingCart създава нови продукти чрез наследяване

3. Кой от изброените модификатори за достъп позволява най-широк достъп до съответния член на класа?

C#	**JAVA**
А) protected	**А)** protected
Б) private	**Б)** private
В) internal	**В)** без (default)
Г) public	**Г)** public

4. Кое от следните твърдения за компонента TextBox (за C#) / TextField (за Java) **е вярно?**

А) Свойството Text (за C#)/ методът getText() (за Java) връща въведения текст като масив от символи

Б) При настройка ReadOnly = true (за C#)/ setEditable(false) (за Java) потребителят не може да вижда текста

В) Свойството Text (за C#)/ методът getText() (за Java) връща въведения текст като символен низ

Г) TextBox (за C#) / TextField (за Java) се използва само за визуализиране на текст, но не и за въвеждане

5. Посочете вярното твърдение за статичен метод на клас:

А) В дефиницията на статичен метод на клас може да се използват както статични, така и нестатични членове на класа

Б) В дефиницията на статичен метод на клас не може да се използват нестатични членове на класа

В) Статичен метод на клас може да се дефинира само в статичен клас

Г) Статичните методи на клас са винаги частни и не може да се достъпват от други класове

6. За дадения програмен фрагмент, кое от твърденията е вярно?

C#

```
string text = "exam";
for (int i = 0; i <= text.Length; i++)
{
    Console.Write(text[i]);
}
```

Java

```
String text = "exam";
for (int i = 0; i <= text.length(); i++){
    System.out.print(text.charAt(i));
}
```

А) Програмата няма да се компилира, защото **`Length`** (за C#) / **`length()`** (за Java) се използва неправилно

Б) Програмата няма да се компилира, защото достъпът до символ в низ е некоректен

В) Програмата ще работи правилно и ще изведе "**`exam`**"

Г) Програмата ще генерира грешка при изпълнение

7. Какво ще се изведе на стандартния изход след извикването на метода **`PrintNumbers(3)`** (**за** C#) / **`printNumbers(3)`** (**за** Java)?

C#

```
public static void PrintNumbers(int n)
{
    if (n > 0)
    {
        PrintNumbers(n - 1);
        Console.Write(" " + n);
    }
}
```

JAVA

```
public static void printNumbers(int n)
 {
    if (n > 0)
    {
       printNumbers(n - 1);
       System.out.print(" " + n);
    }
 }
```

А) `3 2 1`

Б) `1 2 3`

В) `3 1 2`

Г) `1 3 2`

8. Какво ще се изведе на стандартния изход след изпълнение на следния програмен фрагмент:

C#

```
string text = "Barbie without Ken";
text.Replace("without", "meets");
Console.WriteLine(text);
text = text.Replace("without", "meets");
Console.WriteLine(text);
```

Java

```
String text = "Barbie without Ken";
text.replace("without", "meets");
System.out.println(text);
text = text.replace("without", "meets");
System.out.println(text);
```

А)

Barbie without Ken

Barbie without Ken

Б)

Barbie meets Ken

Barbie meets Ken

В)

Barbie without Ken

Barbie meets Ken

Г)

Barbie meets Ken

Barbie without Ken

9. Даден е масив int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

Посочете вярното твърдение:

А) Елементите на масива се разполагат в последователни клетки в статичната памет (стек, stack).

Б) Елементите на масива се разполагат в последователни клетки в динамичната памет (хийп, heap).

В) Елементите на масива се разполагат в произволни клетки в динамичната памет (хийп, heap).

Г) Елементите на масива представляват референции, сочещи клетките в паметта, където са разположени стойностите.

10. Каква е основната разлика между линейно и двоично търсене?

А) Линейното търсене изисква сортиран масив, а двоичното не

Б) Двоичното търсене работи само с числа, а линейното – с всякакви данни

В) Линейното търсене проверява последователно елементите, докато двоичното разделя структурата на части

Г) Линейното търсене винаги е по-бързо от двоичното

11. Какво ще се изведе на стандартния изход, след изпълнение на програмния фрагмент?

C#

```
List<int> nums = new List<int> { 12, 4, -7, 11, 10 };

nums.Insert(2, 11);
nums.RemoveAt(4);
nums.Sort();

for (int i = 0; i < nums.Count; i++)
{
    Console.Write(nums[i] + " ");
}
```

Java

```
List<Integer> nums = new ArrayList<>(Arrays.asList(12, 4, -7, 11, 10));

nums.add(2, 11);
nums.remove(4);
nums.sort(null);

for (int i = 0; i < nums.size(); i++) {
    System.out.print(nums.get(i) + " ");
}
```

А) -7 4 10 11 12

Б) -7 10 11 11 12

В) -7 2 4 11 11 12

Г) -7 2 11 11 12

12. Какво ще се изведе на стандартния изход след изпълнение на следната програма?

C#

```
using System;
public class Test
{
   public static void Main(string[] args)
  {
      try
```

```
      {
          P();
          Console.WriteLine("After the method call");
      }
      catch (FormatException)
      {
          Console.WriteLine("Invalid number format");
      }
      catch (DivideByZeroException)
      {
          Console.WriteLine("Runtime Exception");
      }
      catch (Exception)
      {
          Console.WriteLine("General exception");
      }
  }
  static void P()
  {
      string s = "5.6";
      int num = int.Parse(s);
      int i = 0;
      int y = 2 / i;
      Console.WriteLine("Welcome");
  }
}
```

Java

```
public class Test {
    public static void main(String[] args) {
    try {
      p();
      System.out.println("After the method call");
    }
    catch (NumberFormatException ex) {
     System.out.println("Invalid number format");
    }
    catch (ArithmeticException ex) {
     System.out.println("Runtime Exception");
    }
    catch (Exception ex) {
      System.out.println("General exception");
    }
  }
  static void p() {
    String s = "5.6";
    int num=Integer.parseInt(s);
    int i = 0;
    int y = 2 / i;
    System.out.println("Welcome");
  }
}
```

А) Invalid number format

Б) Welcome

After the method call

В) Runtime Exception

Г) Welcome

General exception

13. Дадена е следната таблица в релационна база от данни:

ID	FirstName	LastName	Grade	Class
1	Иван	Петров	5.50	12А
2	Мария	Иванова	6.00	12А
3	Петър	Димитров	5.75	12Б
4	Елена	Георгиева	5.50	12Б
5	Димитър	Тодоров	5.50	12А

Какъв ще бъде резултатът от изпълнението на следната SQL заявка:

```
SELECT FirstName, LastName, Grade
FROM Students
WHERE Class = '12A'
ORDER BY Grade DESC, LastName ASC;
```

А)

Мария Иванова 6.00

Димитър Тодоров 5.50

Иван Петров 5.50

Б)

Мария Иванова 6.00

Иван Петров 5.50

Димитър Тодоров 5.50

В)

Димитър Тодоров 5.50

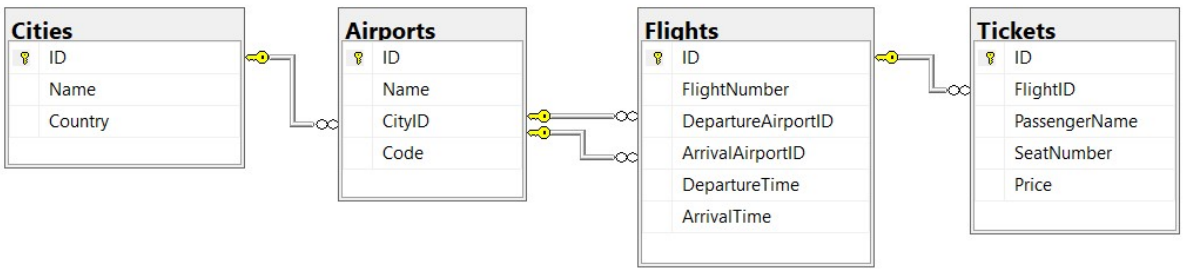
Иван Петров 5.50

Мария	Иванова	6.00

Г)

Мария	Иванова	6.00
Петър	Димитров	5.75
Иван	Петров	5.50

14. Трябва да се изведе информация за полетите, включваща: град на излитане, град на кацане, брой закупени билети за полета. В справката трябва да бъде включена информация и за тези полети, за които все още няма закупени билети. Колко от таблиците от дадената диаграма ще участват в заявка, която връща исканата информация?



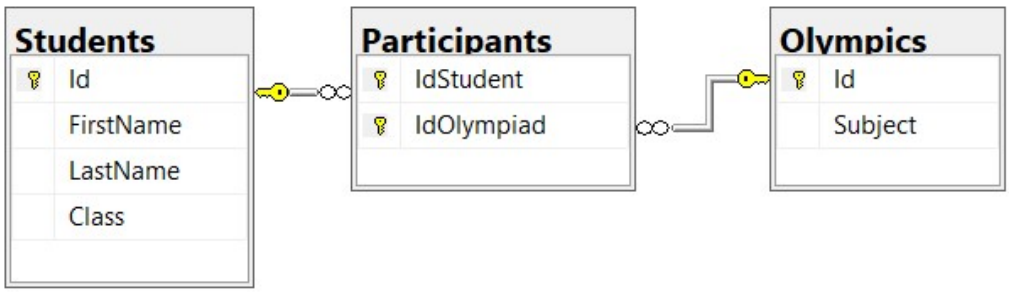
А) 1

Б) 2

В) 3

Г) 4

15. Какъв тип е връзката между таблиците **Students** и **Olympics** от дадената релационна база от данни.



А) Нямат връзка

Б) От тип „едно към едно“

В) От тип „много към много“

Г) От тип „едно към много“

16. Поредността на етапите в жизнения цикъл на информационна система е:

1. Проектиране
2. _____
3. Тестване
4. Използване
5. Поддръжка

Кой е етапът, липсващ в поредицата?

А) Разработване

Б) Анализ

В) Планиране

Г) Реализиране

Отговорите на задачите от 17. до 25. включително запишете в полетата за отговори под задачата!

17. В полето за отговор запишете какво ще се изведе на стандартния изход след изпълнение на програмния фрагмент.

C#

```
int sum = 0, br = 0;
for (int i = 1; i <= 3; i++)
{
    int j = 1;

    while (j <= 2)
    {
        br++;
        sum += i * j;
        j++;
    }
}
Console.WriteLine(br);
Console.WriteLine(sum);
```

Java

```
int sum = 0, br = 0;
for (int i = 1; i <= 3; i++) {

    int j = 1;

    while (j <= 2) {
        br++;
        sum += i * j;
        j++;
```

```
    }
}
System.out.println(br);
System.out.println(sum);
```

18. Запишете в полето за отговор резултата, който ще се изведе на стандартния изход, или думата „Грешка“, ако ще възникне грешка при изпълнението на дадената програма.

C#

```
class MyClass
{
    static void Main(string[] args)
    {
       int[] arr = { 7, 12, 4, 2, 11, 6 };
       int i = 0, sum = 0;
       while (i <= 6)
       {
           sum += ++arr[i];
           if (sum >= 25) break;
          ++i;
      }
      Console.WriteLine("sum = " + sum);
      Console.WriteLine("element = " + arr[i]);
   }
}
```

Java

```
public class MyClass {
  public static void main(String args[]) {
        int [] arr = {7, 12, 4, 2, 11, 6};
        int i=0, sum=0;
        while (i<=6) {
            sum += ++arr[i];
            if (sum>=25) break;
            ++i;
        }
        System.out.println("sum = " + sum);
        System.out.println("element = " + arr[i]);
    }
}
```

19. Чрез дадения програмен фрагмент се цели да се намери най-малкият елемент на едномерния масив и неговият пореден номер. В кода има допуснати грешки и той не дава верен резултат. В полето за отговор запишете номерата на редовете с грешките и след номера на реда запишете верния код. (Приема се, че за C# библиотеката System е включена.)

C#

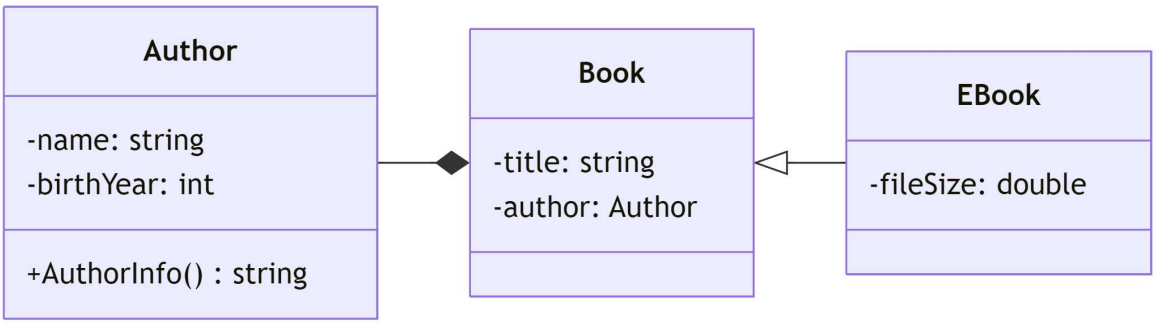
```
1     static void Main(string[] args)
2     {
3         int[] array = { 10, 120, 14, 165, 18 };
4         int index = 0;
5         int min = 0;
6         for(int i = 0; i <= 5; i++)
7         {
8            if (array[i] < min)
9            {
10              min = array[i];
11              index = i;
12           }
13        }
14        Console.WriteLine($"min -> {min} - {index}");
15    }
```

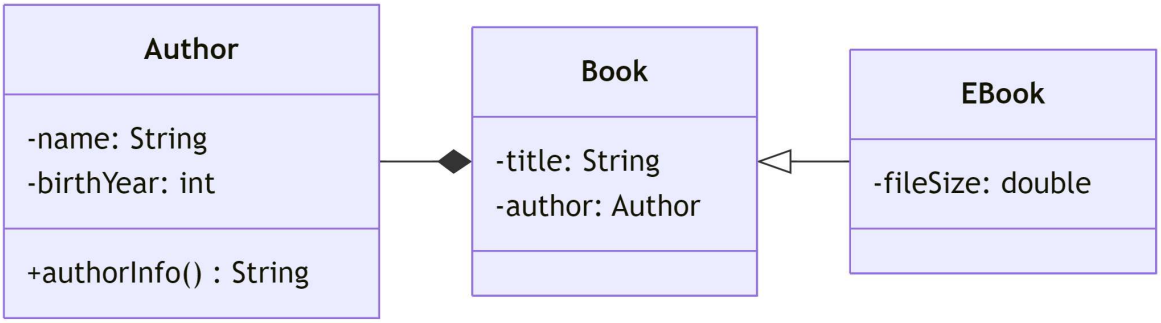
Java

```
 public class Main {
1     public static void main(String[] args) {
2
3         int[] array = { 10, 120, 14, 165, 18 };
4         int index = 0;
5         int min = 0 ;
6          for (int i = 0; i <= 5; i++) {
7
8             if (array[i] < min) {
9
10                min = array[i];
11                 index = i;
12            }
13        }
14         System.out.println("min -> " + min + " - " + index);
15     }
```

20. Дадена е UML диаграма и програмен код, в който има пропуснати части, отбелязани с номера (1), (2) и (3).

C#





C#

```
class Author
{
    private string name;
    private int birthYear;

    public ……(1)……
    {
        string s = "{name}, born in {birthYear}";
        return s;
    }
}
class Book
{
    private string title;
    ……(2)……
}
class ……(3)……
{
    private double fileSize;
}
```

Java

```
class Author {
    private String name;
    private int birthYear;

    public ……(1)……{
        return name + ", born in " + birthYear;
    }
}

class Book {
    private String title;
    ……(2)……
}

class ……(3)……{
    private double fileSize;
}
```

В полетата за отговори (1), (2) и (3) запишете пропуснатите части от кода, така че да се получи програмен код, съответстващ на дадената UML диаграма.

21. С дадения код се четат произволен брой цели числа от файла **Numbers.txt**, написани на отделни редове. Програмният фрагмент пресмята и записва във файла **Result.txt** сумата на прочетените числа. В кода има пропуснати части, отбелязани с (1), (2) и (3).

C#

```
int sum = 0;
StreamReader reader = new StreamReader("Numbers.txt");
using (reader)
{
    string line;
    while ((line = __(1)__) != null)
    {
          int number = int.Parse(line);
          sum += number;
    }
}
StreamWriter writer = __(2)__ ("Result.txt");
using (writer)
{
    __(3)__ ($"Сума: {sum}");
}
```

Java

```
int sum = 0;
try (Scanner reader = new Scanner(new File("Numbers.txt"))) {
   while (reader.hasNextLine()) {
      String line = __(1)__  ;
      int number = Integer.parseInt(line);
      sum += number;
   }
}

try (PrintWriter writer = __(2)__ ("Result.txt")) {
    __(3)__ ("Сума: " + sum);
}
```

В полетата за отговори, за съответния номер, напишете липсващия код, така че да се получи вярно решение. ***Приема се, че нужните библиотеки са включени.***

22. С дадения код, от стандартния вход се прочитат произволен брой цели числа, въведени на един ред, разделени с точно един интервал, представляващи точките на състезатели, участващи в състезание. След това се определя и извежда номера

на победителя и печалбата му. Печалбата е случайно число измежду най-малкия постигнат резултат и общия брой точки, включително. В кода има пропуснати части, отбелязани с (1), (2), (3) и (4).

C#

```
string[] line = Console.ReadLine().__(1)__ (" ");
int[] numbers = new int[line.Length];
for (int i = 0; i < line.Length; i++)
{
     numbers[i] = int.Parse(line[i]);
}
int max = numbers[0], min = numbers[0], sum = 0, ind = -1;
for (int i = 0; i < line.Length; i++)
{
     if (numbers[i] > max)
     {
          max = numbers[i];
          ind = i;
     }
     min = __(2)__.Min(min, numbers[i]);
     sum = sum + numbers[i];
}
__(3)__ rnd = new __(4)__ ();
int profit = rnd.Next(min, sum + 1);
Console.WriteLine($"Winner is number {ind + 1} with profit {profit}");
```

Java

```
Scanner scanner = new Scanner(System.in);
String[]  line = scanner.nextLine().__(1)__ (" ");
int[]numbers = new int[line.length];
for (int i = 0; i < line.length; i++){
    numbers[i] = Integer.parseInt(line[i]);
}
int max = numbers[0], min = numbers[0], sum = 0, ind = -1;
for (int i = 0; i < line.length; i++){
    if ( numbers[i] > max){
        max = numbers[i];
        ind = i;
    }
    min = __(2)__.min(min, numbers[i]);
    sum = sum + numbers[i];
}
__(3)__ rnd = new __(4)__ ();
int profit= rnd.nextInt(min, sum+1);
System.out.println("Winner is number " + (ind+1) + " with profit " + profit );
```

В полетата за отговори, за съответния номер, напишете името на липсващия метод или клас.

23. В дадения програмен фрагмент три реда от кода са заменени с номера (1), (2) и (3). В дясната колона с букви са означени заменените редове.

След изпълнението на програмния код на стандартния изход трябва да се изведе: *Monday Tuesday Wednesday Thursday Friday Saturday Sunday .*

В полетата за отговори за съответния номер на ред, запишете буквата, която съответства на пропуснатия код.

C#	

`string[] days = { "Monday", "Thursday", "Friday" };`
`List<string> listDays = new List<string>();`
`listDays.AddRange(days);`
 	
`listDays.Add("Weekday");`
`    (1)`
`    (2)`
`listDays[5] = "Saturday";`
`    (3)`
 	
`Console.WriteLine(string.Join(" ", listDays));`
	`A) listDays.Insert(1, "Tuesday");`
	`Б) listDays.Add("Sunday");`
	`В) listDays.Insert(1, "Wednesday");`

Java	

`String[] days = {"Monday", "Thursday", "Friday"};`
`ArrayList<String> listDays = new ArrayList<>(Arrays.asList(days));`
 	
`listDays.add( "Weekday");`
`     (1)`
`     (2)`
`listDays.set(5, "Saturday");`
`     (3)`
 	
`System.out.println(String.join(" ", listDays));`
	`A) listDays.add(1, "Tuesday");`
	`Б) listDays.add("Sunday");`
	`В) listDays.add(1, "Wednesday");`

24. Даденият фрагмент от програма работи със списък от имена. Програмата трябва да сортира имената по азбучен ред, да провери дали дадено име присъства в списъка и да намери даденото име на коя позиция е след сортирането. Попълнете пропуснатите части на кода, означени с (1), (2) и (3), така че програмата да работи коректно.

C#

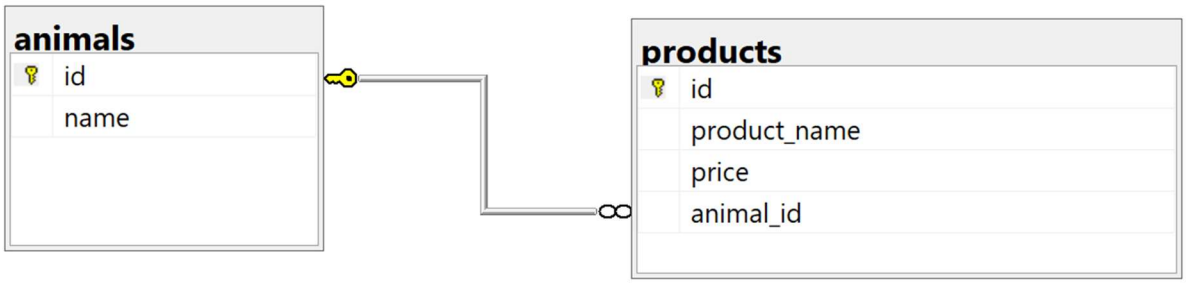
`List<string> students = new List<string>() { "Петър", "Анна", "Мария", "Георги", "Димитър" };`
 	
`students.____1____;`
 	
`string searchName = "Мария";`

```
bool found = students.____2____(searchName);

int position = students.____3____(searchName);

Console.WriteLine($"Списък след сортиране: {string.Join(", ", students)}");
Console.WriteLine($"Намерен ли е {searchName}? {found}");
Console.WriteLine($"Позиция в сортирания списък: {position}");
```

Java

```
ArrayList<String> students = new ArrayList<>(Arrays.asList("Петър", "Анна",
"Мария", "Георги", "Димитър"));

Collections.____1____(students);

String searchName = "Мария";
boolean found = students.____2____(searchName);

int position = students.____3____(searchName);

System.out.println("Списък след сортиране: " + String.join(", ", students));
System.out.println("Намерен ли е " + searchName + "? " + found);
System.out.println("Позиция в сортирания списък: " + position);
```

25. Дадената диаграма представя база данни за зоомагазин и съдържа информация за продуктите, които се продават, и вида на животните.



В заявката по-долу три места са оставени празни, като са обозначени с (1), (2) и (3). В полетата за отговор за (1), (2) и (3) запишете пропуснатото в заявката, така че тя да извежда името на всяко животно и броя продукти, предназначени за него. Трябва да се покажат само животните, за които има повече от един продукт и резултатите да са подредени по азбучен ред на имената.

```
SELECT a.name AS animal_name, COUNT(p.id) AS product_count
FROM products AS p
JOIN animals AS a .......(1).......
....(2).... a.name
HAVING .....(3).....
ORDER BY a.name;
```

МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ ПО

ИНФОРМАТИКА

23 май 2025 г.

ПРОФИЛИРАНА ПОДГОТОВКА

ВАРИАНТ 2

ЧАСТ 2 (Време за работа: 150 минути)

Файловете с отговорите на задачите от 26. до 28. включително, качете в изпитната система като спазите указанията в условието на задачата!

Внимание! Имената на работните файлове, които прикачвате в изпитната система НЕ трябва да съдържат текстове или символи, които могат да доведат до нарушаване на анонимността на изпитната Ви работа!

Задача 26. В нумерологията всеки човек има лично число, което се получава като се съберат цифрите от датата му на раждане. Цифрите на получената сума се сумират до получава на едноцифрено число или-на 11 или 22. Смята се, че личното число определя характерни способности на човека.

Създайте приложение с име **Zad26**, което при зададено цяло число ***n*** и ***n*** на брой рождени дати, намира кое е най-често срещаното лично число и коя е способността, която то определя. Ако най-често срещаното лично число не е единствено, да се изведе това, което е най-малко.

ВХОД:

На първия ред на стандартния вход се въвежда едно цяло положително число **n** (**1 ≤ n ≤ 1000).**

На следващите **n** реда се въвеждат низове в следния формат "**ден-месец-година**", като денят и месецът са зададени с 2 цифри, а годината – с 4 цифри.

Забележка: *Гарантирано е, че входните данни са коректни и не е необходима проверка за валидност на датите или на числото n.*

ИЗХОД:

На **първия** ред на стандартния изход се извежда съобщението: *The most common personal number is* <най-често срещаното число> - < брой срещания> *times*.

На **втория** ред на стандартния изход се извежда "*Characteristic quality:* <характерното качество>". Съответствията на лично число и качество са:

```
1 - Independence
2 - Diplomacy
3 - Natural talent
4 - Organizational skills
5 - Free spirit
6 - Caring and protection
7 - Philosophical skills
8 - Professionals
9 - Tolerance and humanity
11 - Visionaries with ideas
22 - Confidence and intuition
```

Пример:

Вход:

```
10
25-12-1998
01-04-2002
14-03-2000
29-08-1983
23-05-1977
07-06-1998
13-02-1981
05-10-2002
04-08-1994
06-07-2011
```

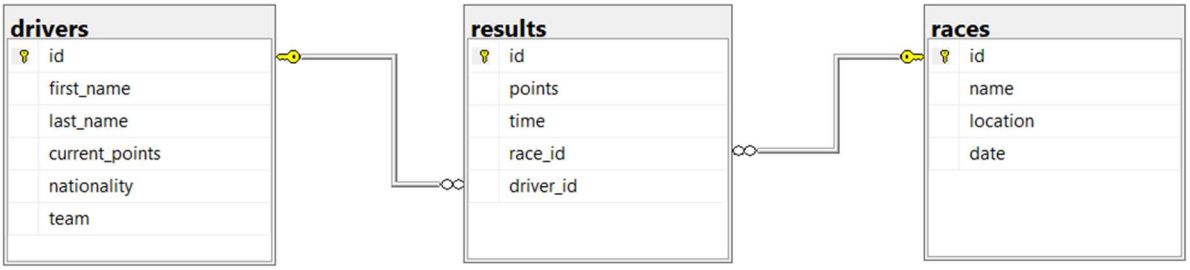
Изход:

```
The most common personal number is 1 - 3 times.
Characteristic quality: Independence
```

*Прикачете в изпитната система архив с име **inf_23.05.2025_zad26.zip**, съдържащ файловете с Вашето решение.*

Приемат се и решения с графичен потребителски интерфейс (ГПИ), в които данните се въвеждат в подходящи контроли, а резултатът се извежда в етикет или нередактируемо текстово поле.

Задача 27. Дадена е диаграма на релационна база от данни **`formula1`**, която съхранява информация за резултатите от състезанията на пилотите от Формула 1. Създайте релационна база от данни **`formula1.`**



А) Създайте в базата данни таблица **`drivers`** със следните атрибути:

Име на колоната	Тип на данните	Ограничения
`id`	цяло число	автоматично се увеличава с 1, първичен ключ
`first_name`	текст до 50 символа	задължително **съдържа** стойност
`last_name`	текст до 50 символа	задължително **съдържа** стойност
`current_points`	реално число с 2 цифри след десетичния разделител	стойност по подразбиране нула
`nationality`	текст до 50 символа	
`team`	текст до 50 символа	задължително **съдържа** стойност

Б) Създайте в базата данни таблица **`races`** със следните атрибути:

Име на колоната	Тип на данните	Ограничения
`id`	цяло число	автоматично се увеличава с 1, първичен ключ
`name`	текст до 80 символа	задължително **съдържа** стойност
`location`	текст до 100 символа	задължително **съдържа** стойност
`date`	дата	задължително **съдържа** стойност

В) Създайте в базата данни таблица **`results`** със следните атрибути:

Име на колоната	Тип на данните	Ограничения
`id`	цяло число	автоматично се увеличава с 1, първичен ключ

`points`	реално число с 2 цифри след десетичния разделител	задължително **съдържа** стойност
`time`	време	
`race_id`	цяло число	задължително **съдържа** стойност, външен ключ към таблицата **`races`**
`driver_id`	цяло число	задължително **съдържа** стойност, външен ключ към таблицата **`drivers`**

Г) За всяка от таблиците в базата от данни добавете дадените кортежи (данните за таблиците са в ресурсния файл с име `resources.txt`):

За таблицата **`drivers`**:

1	Fernando	Alonso	87.00	Spanish	Aston Martin
2	Lewis	Hamilton	120.00	British	Mercedes
3	Max	Verstappen	130.50	Dutch	Red Bull Racing
4	Charles	Leclerc	148.00	Monegasque	Ferrari
5	George	Russell	78.50	British	Mercedes
6	Sergio	Perez	85.00	Mexican	Red Bull Racing

За таблицата **`races`**:

1	Bahrain Grand Prix	Sakhir	2025-03-02
2	Saudi Arabian Grand Prix	Jeddah	2025-03-16
3	Australian Grand Prix	Melbourne	2025-03-30
4	Monaco Grand Prix	Monaco	2025-05-26
5	British Grand Prix	Silverstone	2025-07-07
6	Japanese Grand Prix	Suzuka	2025-10-13

За таблицата **`results`**:

1	25.00	01:33:45	1	1
2	18.00	01:34:02	1	2
3	25.00	01:27:12	2	2
4	18.00	01:28:45	2	3
5	25.00	01:35:10	3	3
6	15.00	01:36:30	3	4
7	25.00	01:38:45	4	5
8	18.00	01:39:20	4	6

Д) Напишете заявка, която променя локацията (`location`) на състезанието `Monaco Grand Prix` така, че след промяната да бъде `Monte Carlo`.

Е) Напишете заявка, която да изведе име, фамилия и текущи точки на пилотите, които имат повече от 100 текущи точки. Резултатът да бъде подреден в низходящ ред на точките. При вярно написана заявка се получава резултатът, представен в таблица Резултат Е.

first_name	last_name	current_points
Charles	Leclerc	148.00
Max	Verstappen	130.50
Lewis	Hamilton	120.00

Резултат Е

Ж) Напишете заявка, която да изведе средния брой текущи точки на пилотите от националност (`nationality`), започваща с буква М.
При вярно написана заявка се получава резултатът, представен в таблица Резултат Ж.

average_points
116.500000

Резултат Ж

З) Напишете заявка, която да изведе отборите (`team`) и броя на пилотите от всеки отбор.
При вярно написана заявка се получава резултатът, представен в таблица Резултат З.

team	number_of_drivers
Aston Martin	1
Ferrari	1
Mercedes	2
Red Bull Racing	2

Резултат З

И) Напишете заявка, която да изведе име и фамилия на пилотите, които участват в дадените състезания, името на състезанието и броя точки, които получават от тях.
При вярно написана заявка се получава резултатът, представен в таблица Резултат И.

first_name	last_name	race_name	points
Lewis	Hamilton	Bahrain Grand Prix	25.00
Max	Verstappen	Bahrain Grand Prix	18.00
Max	Verstappen	Saudi Arabian Grand Prix	25.00
Charles	Leclerc	Saudi Arabian Grand Prix	18.00
Charles	Leclerc	Australian Grand Prix	25.00
George	Russell	Australian Grand Prix	15.00
Sergio	Perez	Monaco Grand Prix	25.00
Fernando	Alonso	Monaco Grand Prix	18.00

Резултат И

Прикачете в изпитната система файл с име inf_23.05.2025_zad27.zip, съдържащ създадената база от данни (при работа с MS Access) или заявките за създаването на база от данни (при работа със стандартен език за заявки) и написаните от Вас заявки към базата данни.

Задача 28. Създайте приложение **`MusicSchoolApp`** (Музикална школа), което имплементира базов клас **`SchoolMember`** (участник в школата) и негови наследници **`Student`** (ученик), **`Teacher`** (преподавател) и **`Manager`** (мениджър) със следното описание:

А) Абстрактен базов клас: `SchoolMember`

Полета:

- **`name`** (низ) – име на участник в школата.
- **`age`** (цяло число) – възраст.

Методи:

– Конструктор, който задава стойности за всички полета.

– Абстрактен метод **`Info()`** (за C#) / **`info`**() (за Java), който извежда на стандартния изход информация за съответния участник в школата.

Данните да се валидират в съответствие със следните ограничения:

– Името не може да бъде празен низ. При невалидно име се генерира изключение със съобщение "*Name cannot be an empty string!*"

– Възрастта трябва да бъде число, по-голямо от 5. При невалидна възраст се генерира изключение със съобщение "*Age must be greater than 5 years!*"

Б) Клас наследник: `Student`

Полета:

– **`instrument`** (низ) – музикалният инструмент, който ученикът изучава.

– **`practiceHours`** (цяло число) – брой часове практика седмично.

Методи:

– **Конструктор**, който задава стойности за всички полета (включително от базовия клас).

– **Предефиниран** метод **`Info`**() (за C#) / **`info`**() (за Java), който извежда информация за ученик във формат:

– `Student: <име>, Age: <възраст> years`

– `Instrument: <инструмент>`

– `Practice hours: <часове практика> per week`

Данните да се валидират в съответствие със следните ограничения:

– Стойността на полето `instrument` не може да бъде празен низ. При невалидна стойност се генерира изключение със съобщение "*Instrument cannot be an empty string!*"

– Стойността на полето `practiceHours` трябва да бъде положително число. При невалидна стойност се генерира изключение със съобщение "*Practice hours must be a positive number!*"

В) Клас наследник: Teacher

Полета:

– **`specialty`** (низ) – специалността на учителя – някоя от думите **`Piano`** (пиано), **`Violin`** (цигулка) или **`Guitar`** (китара).

– **`studentsCount`** (цяло число) – брой ученици, които учителят обучава.

– **`salary`** (реално число) – заплата.

– **`bonus`** (реално число) – бонусът се изчислява като процент от заплатата на базата на броя ученици (2% на ученик).

Методи:

– **Конструктор**, който задава стойности за всички полета (включително от базовия клас).

– **Предефиниран** метод **`Info()`** (за C#) или **`info()`** (за Java), който извежда информация за преподавателя във формат:

– `Teacher: <име>, Age: <възраст> years`

– `Specialty: <специалност>`

– `Students count: <брой ученици>`

– `Salary: <заплата, форматирана до втория знак> lv.`

– `Bonus: <бонус, форматиран до втория знак> lv.`

Данните да се валидират в съответствие със следните ограничения:

– Стойността на полето `specialty` не може да бъде празен низ. При невалидна стойност се генерира изключение със съобщение "*Specialty cannot be an empty string!*"

– Стойността на полето `studentsCount` трябва да бъде неотрицателно число. При невалидна стойност се генерира изключение със съобщение "*Students count cannot be negative*!"

– Стойността на полето `salary` трябва да бъде неотрицателно число. При невалидна стойност се генерира изключение със съобщение "*Salary cannot be a negative number!*"

Г) Клас наследник: Manager

Полета:

– **`budget`** (реално число) – бюджетът на школата.

– **`yearsInService`** (цяло число) **–** години на работа, като мениджър.

Методи:

– **Конструктор**, който задава стойности за всички полета (включително от базовия клас).

– **Предефиниран** метод **`Info()`** (за C#) / **`info()`** (за Java), който извежда информация във формат:

– `Manager: <име>, Age: <възраст> years`

– `Budget: <бюджет> lv`

– `Years in service: <стаж> years`

Данните да се валидират в съответствие със следните ограничения:

– Стойността на полето `budget` трябва да бъде положително число. При невалидна стойност се генерира изключение със съобщение "*Budget must be a positive number!*"

– Стойността на полето `yearsInService` трябва да бъде неотрицателно число. При невалидна стойност се генерира изключение със съобщение "*Years in service cannot be negative!*"

Д) Приложението трябва да реализира следните функционалности:

1. Данните за членовете на музикалната школа се четат от текстов файл **`input.txt`**. Всеки ред от файла съдържа данните за един член на школата в следния формат:

– За ученик: **`Student`**`;<име>;<възраст>;<инструмент>;<часове практика>`

– За преподавател: **`Teacher`**`;<име>;<възраст>;<специалност>;<брой ученици>;<заплата>`

– За мениджър: **`Manager`**`;<име>;<възраст>;<бюджет>;<стаж>`

2. На стандартния изход се извежда информация за всички членове на школата, подредени по възраст в нарастващ ред. За разделител между всеки от тях се използва ред с 10 тирета.

3. Прихващане и обработка на изключения – при невалидни данни в някой от редовете на файла `input.txt` на стандартния изход се извежда съобщение "*Invalid data on line* **<X>: <детайли за грешката>**", където *<X>* е номерът на реда в `input.txt` файла, а <детайли за грешката> са съобщенията от валидациите. След като приложението обработи възникналата грешка, продължава работата с останалите редове.

4. Накрая, **ако има въведени учители**, се извежда информация във формат:

"*The teacher with the most students is* <име> *with* <брой ученици> *students.*" за преподавателя с най-много ученици. Ако има повече от един учител с максимален брой ученици, то да се изведе информацията за първия срещнат, такъв учител.

Прикачете в изпитната система архив с име ***inf_23.05.2025_zad28.zip****, съдържащ файловете с Вашите решения.*

Изход с използване на ресурсния файл с име input.txt:

```
Invalid data on line 3: Name cannot be an empty string!
Invalid data on line 7: Instrument cannot be an empty string!
Invalid data on line 11: Age must be greater than 5 years!
Student: Ivan Ivanov, Age: 12 years
Instrument: Guitar
Practice hours: 5 per week
----------
Student: Ivaylo Todorov, Age: 12 years
Instrument: Piano
Practice hours: 10 per week
----------
Student: Maria Petrova, Age: 15 years
Instrument: Piano
Practice hours: 8 per week
----------
Student: Peter Dimitrov, Age: 17 years
Instrument: Violin
Practice hours: 10 per week
----------
Teacher: Sofia Vasileva, Age: 28 years
Specialty: Violin
Students count: 15
Salary: 2800,00 lv
Bonus: 840,00 lv
----------
Teacher: Anna Kircheva, Age: 35 years
Specialty: Piano
Students count: 12
Salary: 2500,00 lv
Bonus: 600,00 lv
----------
Manager: Elena Dimitrova, Age: 38 years
Budget: 120000,00 lv
Years in service: 5 years
----------
Teacher: Diana Petrova, Age: 40 years
Specialty: Violin
Students count: 10
Salary: 2000,00 lv
Bonus: 400,00 lv
----------
Teacher: Georgi Nikolov, Age: 42 years
```

```
Specialty: Guitar
Students count: 8
Salary: 2200,00 lv
Bonus: 352,00 lv
----------
Manager: Vasil Yordanov, Age: 45 years
Budget: 150000,00 lv
Years in service: 8 years
----------
The teacher with the most students is Sofia Vasileva with 15 students.
```

МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ ПО

ИНФОРМАТИКА

23 май 2025 г.

ПРОФИЛИРАНА ПОДГОТОВКА

ВАРИАНТ 2

Ключ с верните отговори на задачи от 1. до 16.

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

Верни отговори, примерни решения и критерии за оценяване на задачи от 17. до 25.

17. – 2 точки

6
18

18. – 2 точки

sum = 26
element = 5

19. – 2 точки

ред 5

min = array[0] или

min = int.MaxValue (за C#) / min = Integer.MAX_VALUE (за Java)

ред 6

for (int i = 1; i <5; i++) i<5 или i< array.Length (за C#) / i < array.length (за Java)

20. – 3 точки

(1) string AuthorInfo() (за C#) / String authorInfo() (за Java)
(2) private Author author; (за C# може и само: Author author;)
(3) EBook: Book (за C#) / EBook extends Book (за Java)

21. – 3 точки

(1) reader.ReadLine() (за C#) / reader.nextLine() (за Java)
(2) new StreamWriter (за C#) / new PrintWriter (за Java)
(3) writer.WriteLine (за C#) / writer.write (за Java)

22. – 3 точки

(1) Split (за C#) / split (за Java)
(2) Math
(3) Random
(4) Random

23. – 3 точки

(1)	В
(2)	А
(3)	Б

24. – 3 точки

За C#:

(1) Sort()
(2) Contains
(3) IndexOf

За Java:

(1) sort
(2) contains
(3) indexOf

25. – 3 точки

(1) ON p.animal_id = a.id
(2) GROUP BY
(3) COUNT(p.id) > 1

26. – 15 точки

Примерно решение:

C#

```
using System;
using System.Linq;
namespace Zad26
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            int[] pNumbers = new int[25];
            for (int i = 1; i <= n; i++)
            {
                string birthDate = Console.ReadLine();
                int num = Calculate(birthDate);
                pNumbers[num]++;
            }
            int max = pNumbers[0], ind = 0;
            for (int i = 0; i < n; i++)
            {
                if (pNumbers[i] > max)
                {
                    max = pNumbers[i];
                    ind = i;
                }
            }
            Console.WriteLine($"The most common personal number is {ind} -
{max} times.");
            Console.Write("Characteristic quality: ");
            if (ind == 1) Console.WriteLine("Independence");
            if (ind == 2) Console.WriteLine("Diplomacy");
            if (ind == 3) Console.WriteLine("Natural talent");
            if (ind == 4) Console.WriteLine("Organizational skills");
            if (ind == 5) Console.WriteLine("Free spirit");
            if (ind == 6) Console.WriteLine("Caring and protection");
            if (ind == 7) Console.WriteLine("Philosophical skills");
            if (ind == 8) Console.WriteLine("Professionals");
            if (ind == 9) Console.WriteLine("Tolerance and humanity");
            if (ind == 11) Console.WriteLine("Visionaries with ideas");
            if (ind == 22) Console.WriteLine("Confidence and intuition");
        }

        private static int Calculate(string birthDate)
        {
            int[] parts = birthDate.Split('-').Select(int.Parse).ToArray();
            int d = parts[0];
            int m = parts[1];
            int y = parts[2];
            int sum = d / 10 + d % 10 + m / 10 + m % 10 + y / 1000 + y / 100
% 10 + y / 10 % 10 + y % 10;
            while (sum >= 10 && sum != 11 && sum != 22)
```

```
            {
                sum = sum / 10 + sum % 10;
            }
            return sum;
        }
    }
}
```

JAVA

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = Integer.parseInt(scanner.nextLine());
        int[] pNumbers = new int[25];
        for (int i = 1; i <= n; i++) {
            String birthDate = scanner.nextLine();
            int num = calculate(birthDate);
            pNumbers[num]++;
        }
        int max = pNumbers[0], ind = 0;
        for (int i = 0; i < n; i++) {
            if (pNumbers[i] > max) {
                max = pNumbers[i];
                ind = i;
            }
        }
        System.out.println("The most common personal number is " + ind + " -
" + max + " times.");
        System.out.print("Characteristic quality: ");
        if (ind == 1) System.out.println("Independence");
        if (ind == 2) System.out.println("Diplomacy");
        if (ind == 3) System.out.println("Natural talent");
        if (ind == 4) System.out.println("Organizational skills");
        if (ind == 5) System.out.println("Free spirit");
        if (ind == 6) System.out.println("Caring and protection");
        if (ind == 7) System.out.println("Philosophical skills");
        if (ind == 8) System.out.println("Professionals");
        if (ind == 9) System.out.println("Tolerance and humanity");
        if (ind == 11) System.out.println("Visionaries with ideas");
        if (ind == 22) System.out.println("Confidence and intuition");
    }
    private static int calculate(String birthDate) {
        String[] parts = birthDate.split("-");
        int d = Integer.parseInt(parts[0]);
        int m = Integer.parseInt(parts[1]);
        int y = Integer.parseInt(parts[2]);
        int sum = d / 10 + d % 10 + m / 10 + m % 10 + y / 1000 + y / 100 %
10 + y / 10 % 10 + y % 10;
        while (sum >= 10 && sum != 11 && sum != 22) {
            sum = sum / 10 + sum % 10;
        }
        return sum;
    }
}
```

27. – 20 точки

Примерно решение:

```
create database formula1;
use formula1;

CREATE TABLE drivers (
    id INT IDENTITY(1,1) PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    current_points DECIMAL(10, 2) DEFAULT 0,
    nationality VARCHAR(50),
    team VARCHAR(50) NOT NULL
);

CREATE TABLE races (
    id INT IDENTITY(1,1) PRIMARY KEY,
    name NVARCHAR(80) NOT NULL,
    location NVARCHAR(100) NOT NULL,
    date DATE NOT NULL
);

CREATE TABLE results (
    id INT IDENTITY(1,1) PRIMARY KEY,
    points DECIMAL(6, 2) NOT NULL,
    time TIME,
    race_id INT NOT NULL,
    driver_id INT NOT NULL,
    FOREIGN KEY (driver_id) REFERENCES Drivers(id),
    FOREIGN KEY (race_id) REFERENCES Races(id)
);

INSERT INTO drivers (first_name, last_name, current_points,nationality,
team)
VALUES
('Lewis', 'Hamilton',120.00, 'British', 'Mercedes'),
('Max', 'Verstappen',130.50, 'Dutch', 'Red Bull Racing'),
('Charles', 'Leclerc', 148.00,'Monegasque', 'Ferrari'),
('George', 'Russell', 78.50, 'British', 'Mercedes'),
('Sergio', 'Perez', 85.00,'Mexican', 'Red Bull Racing'),
('Fernando', 'Alonso',87.00, 'Spanish', 'Aston Martin');

INSERT INTO races (name, location, date)
VALUES
('Bahrain Grand Prix', 'Sakhir', '2025-03-02'),
('Saudi Arabian Grand Prix', 'Jeddah', '2025-03-16'),
('Australian Grand Prix', 'Melbourne', '2025-03-30'),
('Monaco Grand Prix', 'Monaco', '2025-05-26'),
('British Grand Prix', 'Silverstone', '2025-07-07'),
('Japanese Grand Prix', 'Suzuka', '2025-10-13');

INSERT INTO results (points, time, race_id, driver_id)
VALUES
(25.00, '01:33:45', 1, 1),
(18.00, '01:34:02', 1, 2),
```

```
(25.00, '01:27:12', 2, 2),
(18.00, '01:28:45', 2, 3),
(25.00, '01:35:10', 3, 3),
(15.00, '01:36:30', 3, 4),
(25.00, '01:38:45', 4, 5),
(18.00, '01:39:20', 4, 6);

-- Д)
UPDATE races
SET location = 'Monte Carlo'
WHERE name = 'Monaco Grand Prix';

-- Е)
SELECT first_name, last_name, current_points
FROM drivers
WHERE current_points > 100
ORDER BY current_points DESC;

-- Ж)
SELECT AVG(current_points) AS average_points
FROM drivers
WHERE nationality LIKE 'M%';

-- З)
SELECT team, COUNT(*) AS number_of_drivers
FROM drivers
GROUP BY team;

-- И)
SELECT d.first_name, d.last_name, r.name AS race_name, res.points
FROM results res
JOIN drivers d ON res.driver_id = d.id
JOIN races r ON res.race_id = r.id;
```

28. – 25 точки

Примерно решение:

C#

```
using System;
namespace Zad28
{
    public abstract class SchoolMember
    {
      private string name;
      private int age;

      public string Name
      {
        get => name;
        set
        {
            if (string.IsNullOrEmpty(value))
                throw new ArgumentException("Name cannot be an empty
string!");
            name = value;
```

```
        }
      }

    public int Age
    {
        get => age;
        set
        {
            if (value <= 5)
                throw new ArgumentException("Age must be greater than 5
years!");
            age = value;
        }
    }

    public SchoolMember(string name, int age)
    {
        Name = name;
        Age = age;
    }

    public abstract void Info();
}

class Student : SchoolMember
{
    private string instrument;
    private int practiceHours;

    public string Instrument
    {
        get => instrument;
        set
        {
            if (string.IsNullOrEmpty(value))
                throw new ArgumentException("Instrument cannot be an empty
string!");
            instrument = value;
        }
    }

    public int PracticeHours
    {
        get => practiceHours;
        set
        {
            if (value <= 0)
                throw new ArgumentException("Practice hours must be a
positive number!");
            practiceHours = value;
        }
    }

    public Student(string name, int age, string instrument, int
practiceHours)
        : base(name, age)
    {
        Instrument = instrument;
```

```
        PracticeHours = practiceHours;
    }

    public override void Info()
    {
        Console.WriteLine($"Student: {Name}, Age: {Age} years");
        Console.WriteLine($"Instrument: {Instrument}");
        Console.WriteLine($"Practice hours: {PracticeHours} per week");
    }
}

class Teacher : SchoolMember
{
    private string specialty;
    private int studentsCount;
    private double salary;
    private double bonus;

    public string Specialty
    {
        get => specialty;
        set
        {
            if (string.IsNullOrEmpty(value))
                throw new ArgumentException("Specialty cannot be an empty
string!");
            specialty = value;
        }
    }

    public int StudentsCount
    {
        get => studentsCount;
        set
        {
            if (value < 0)
                throw new ArgumentException("Students count cannot be
negative!");
            studentsCount = value;
            CalculateBonus();
        }
    }

    public double Salary
    {
        get => salary;
        set
        {
            if (value < 0)
                throw new ArgumentException("Salary cannot be a negative
number!");
            salary = value;
            CalculateBonus();
        }
    }

    public double Bonus => bonus;
```

```
    private void CalculateBonus()
    {
        bonus = salary * (studentsCount * 0.02);
    }

    public Teacher(string name, int age, string specialty, int
studentsCount, double salary)
        : base(name, age)
    {
        Specialty = specialty;
        Salary = salary;
        StudentsCount = studentsCount;
    }

    public override void Info()
    {
        Console.WriteLine($"Teacher: {Name}, Age: {Age} years");
        Console.WriteLine($"Specialty: {Specialty}");
        Console.WriteLine($"Students count: {StudentsCount}");
        Console.WriteLine($"Salary: {Salary:F2} lv");
        Console.WriteLine($"Bonus: {Bonus:F2} lv");
    }
}

class Manager : SchoolMember
{
    private double budget;
    private int yearsInService;

    public double Budget
    {
        get => budget;
        set
        {
            if (value <= 0)
                throw new ArgumentException("Budget must be a positive
number!");
            budget = value;
        }
    }

    public int YearsInService
    {
        get => yearsInService;
        set
        {
            if (value < 0)
                throw new ArgumentException("Years in service cannot be
negative!");
            yearsInService = value;
        }
    }

    public Manager(string name, int age, double budget, int yearsInService)
        : base(name, age)
    {
        Budget = budget;
        YearsInService = yearsInService;
```

```
    }

    public override void Info()
    {
        Console.WriteLine($"Manager: {Name}, Age: {Age} years");
        Console.WriteLine($"Budget: {Budget:F2} lv");
        Console.WriteLine($"Years in service: {YearsInService} years");
    }
}
internal class Zad28
{
    static void Main(string[] args)
    {
        List<SchoolMember> members = new List<SchoolMember>();
        int lineNumber = 0;
        Teacher bestTeacher = null;
        using (StreamReader reader = new StreamReader("input.txt"))
        {
            string line;
            while ((line = reader.ReadLine()) != null)
            {
                lineNumber++;
                try
                {
                    string[] data = line.Split(';');
                    switch (data[0])
                    {
                        case "Student":
                            members.Add(new Student(data[1],
int.Parse(data[2]), data[3], int.Parse(data[4])));
                            break;
                        case "Teacher":
                            Teacher t = new Teacher(data[1],
int.Parse(data[2]),
                                data[3], int.Parse(data[4]),
double.Parse(data[5]));
                            members.Add(t);
                            // намиране на учителя с най-много ученици
                            if (bestTeacher == null || t.StudentsCount >
bestTeacher.StudentsCount)
                            {
                                bestTeacher = t;
                            }
                            break;

                        case "Manager":
                            members.Add(new Manager(data[1],
int.Parse(data[2]),
                                double.Parse(data[3]), int.Parse(data[4])));
                            break;
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"Invalid data on line {lineNumber}:
{ex.Message}");
                }
            }
```

```
        }

        var sortedMembers = members.OrderBy(m => m.Age);
        foreach (var member in sortedMembers)
        {
            member.Info();
            Console.WriteLine("----------");
        }
        if (bestTeacher != null)
        {
            Console.WriteLine($"The teacher with the most students is
{bestTeacher.Name} with" +
                              $" {bestTeacher.StudentsCount} students.");
        }

    }
  }
}
```

JAVA

```
import java.io.FileInputStream;
import java.util.*;
abstract class SchoolMember {
    private String name;
    private int age;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        if (name == null || name.trim().isEmpty()) {
            throw new IllegalArgumentException("Name cannot be an empty
string!");
        }
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        if (age <= 5) {
            throw new IllegalArgumentException("Age must be greater than 5
years!");
        }
        this.age = age;
    }

    public SchoolMember(String name, int age) {
        setName(name);
        setAge(age);
```

```
    }

    public abstract void info();
}

class Student extends SchoolMember {
    private String instrument;
    private int practiceHours;

    public String getInstrument() {
        return instrument;
    }

    public void setInstrument(String instrument) {
        if (instrument == null || instrument.trim().isEmpty()) {
            throw new IllegalArgumentException("Instrument cannot be an
empty string!");
        }
        this.instrument = instrument;
    }

    public int getPracticeHours() {
        return practiceHours;
    }

    public void setPracticeHours(int practiceHours) {
        if (practiceHours <= 0) {
            throw new IllegalArgumentException("Practice hours must be a
positive number!");
        }
        this.practiceHours = practiceHours;
    }

    public Student(String name, int age, String instrument, int
practiceHours) {
        super(name, age);
        setInstrument(instrument);
        setPracticeHours(practiceHours);
    }

    @Override
    public void info() {
        System.out.println("Student: " + getName() + ", Age: " + getAge() +
" years");
        System.out.println("Instrument: " + instrument);
        System.out.println("Practice hours: " + practiceHours + " per
week");
    }
}

class Teacher extends SchoolMember {
```

```
    private String specialty;
    private int studentsCount;
    private double salary;
    private double bonus;

    public String getSpecialty() {
        return specialty;
    }

    public void setSpecialty(String specialty) {
        if (specialty == null || specialty.trim().isEmpty()) {
            throw new IllegalArgumentException("Specialty cannot be an empty
string!");
        }
        this.specialty = specialty;
    }

    public int getStudentsCount() {
        return studentsCount;
    }

    public void setStudentsCount(int studentsCount) {
        if (studentsCount < 0) {
            throw new IllegalArgumentException("Students count cannot be
negative!");
        }
        this.studentsCount = studentsCount;
        calculateBonus();
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        if (salary < 0) {
            throw new IllegalArgumentException("Salary cannot be a negative
number!");
        }
        this.salary = salary;
        calculateBonus();
    }

    public double getBonus() {
        return bonus;
    }

    private void calculateBonus() {
        this.bonus = salary * (studentsCount * 0.02);
    }
```

```
    public Teacher(String name, int age, String specialty, int
studentsCount, double salary) {
        super(name, age);
        setSpecialty(specialty);
        setSalary(salary);
        setStudentsCount(studentsCount);
    }

    @Override
    public void info() {
        System.out.println("Teacher: " + getName() + ", Age: " + getAge() +
" years");
        System.out.println("Specialty: " + specialty);
        System.out.println("Students count: " + studentsCount);
        System.out.printf("Salary: %.2f lv%n", salary);
        System.out.printf("Bonus: %.2f lv%n", bonus);
    }
}

class Manager extends SchoolMember {
    private double budget;
    private int yearsInService;

    public double getBudget() {
        return budget;
    }

    public void setBudget(double budget) {
        if (budget <= 0) {
            throw new IllegalArgumentException("Budget must be a positive
number!");
        }
        this.budget = budget;
    }

    public int getYearsInService() {
        return yearsInService;
    }

    public void setYearsInService(int yearsInService) {
        if (yearsInService < 0) {
            throw new IllegalArgumentException("Years in service cannot be
negative!");
        }
        this.yearsInService = yearsInService;
    }

    public Manager(String name, int age, double budget, int yearsInService)
{
        super(name, age);
        setBudget(budget);
```

```
        setYearsInService(yearsInService);
    }

    @Override
    public void info() {
        System.out.println("Manager: " + getName() + ", Age: " + getAge() +
" years");
        System.out.printf("Budget: %.2f lv%n", budget);
        System.out.println("Years in service: " + yearsInService + "
years");
    }
}

public class Zad28 {
    public static void main(String[] args) {
        List<SchoolMember> members = new ArrayList<>();
        int lineNumber = 0;
        Teacher maxStudentTeacher = null;
        try (Scanner reader = new Scanner(new
FileInputStream("input1.txt"))) {
            while (reader.hasNextLine()) {
                lineNumber++;
                String line = reader.nextLine();
                try {
                    String[] data = line.split(";");
                    switch (data[0]) {
                        case "Student":
                            members.add(new Student(data[1],
Integer.parseInt(data[2]),
                                    data[3], Integer.parseInt(data[4])));
                            break;
                        case "Teacher":
                            Teacher t = new Teacher(data[1],
Integer.parseInt(data[2]),
                                        data[3], Integer.parseInt(data[4]),
Double.parseDouble(data[5]));
                            members.add(t);
// намиране на учителя с най-много ученици
                            if ((maxStudentTeacher == null) ||
(maxStudentTeacher.getStudentsCount() < t.getStudentsCount())) {
                                maxStudentTeacher = t;
                            }
                            break;

                        case "Manager":
                            members.add(new Manager(data[1],
Integer.parseInt(data[2]),
                                    Double.parseDouble(data[3]),
Integer.parseInt(data[4])));
                            break;
                    }
```

```
                } catch (Exception ex) {
                    System.out.println("Invalid data on line " + lineNumber
+ ": " + ex.getMessage());
                }
            }
        } catch (Exception ex) {
            System.out.println("Error reading file: " + ex.getMessage());
            return;
        }

        members.stream()
            .sorted(Comparator.comparing(SchoolMember::getAge))
            .forEach(member -> {
                member.info();
                System.out.println("----------");
            });

        if (maxStudentTeacher != null) {
            System.out.println("The teacher with the most students is " +
                    maxStudentTeacher.getName() + " with " +
maxStudentTeacher.getStudentsCount() + " students.");
        }
    }
}
```