**МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА**

**ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ ПО**

**ИНФОРМАТИКА**

**20 май 2024 г.**

**ПРОФИЛИРАНА ПОДГОТОВКА**

**ВАРИАНТ 1**

**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. Как да се изчисли произведението на две цели числа, записани в текстови полета с имена a и b?**

| **C#** |
|---|
| А) `int.Parse(a.Text)*int.Parse(b.Text)` |
| Б) `int.Parse(a)*int.Parse(b)` |
| В) `a * b` |
| Г) `a.Text * b.Text` |
| **Java** |
| А) `Integer.parseInt(a.getText())*Integer.parseInt(b.getText())` |
| Б) `Integer.parseInt(a)*Integer.parseInt(b)` |
| В) `a * b` |
| Г) `a.getText() * b.getText()` |

**2. Посочете вярното твърдение за статичен метод на клас:**

А) Статичният метод на клас е частен член на класа.

Б) Статичен метод може да се дефинира само в статичен клас.

В) Статичният метод принадлежи на класа, общ е за всички инстанции.

Г) Статичният метод има само декларация, без имплементация.

**3. Отбележете коя структура от данни трябва да се използва, за да отговаря на подхода за обработка на данни „пръв влязъл – пръв излязъл“ (FIFO)?**

А) масив

Б) опашка

В) списък

Г) стек

**4. В коя от дадените двойки обекти първият е във връзка от тип „е“ (is-a) с втория?**

А) мобилно устройство – GSM

Б) GSM – лаптоп

В) мобилно устройство – таблет

Г) лаптоп – мобилно устройство

**5. Какъв ще е резултатът при опит за изпълнение на дадения програмен фрагмент със стойност на променливата n = 17?**

**C#**

```
int n = double.Parse(Console.ReadLine());
double a = n / 2;
Console.WriteLine(a);
```

**Java**

```
Scanner scanner = new Scanner(System.in);
int n=Double.parseDouble(scanner.nextLine());
double a = n / 2;
System.out.println(a);
```

А) Ще възникне грешка по време на изпълнение.

Б) Ще възникне грешка при компилиране.

В) Програмата ще стартира и ще изведе 8.

Г) Програмата ще стартира и ще изведе 8.5.

**6. Какво ще се изведе на стандартния изход при изпълнението на следния програмен фрагмент?**

**C#**

```
string s = " six impossible things before breakfast ";
int i = s.IndexOf('i');
Console.Write(i + " ");
Console.WriteLine("\"" + s.Trim() + "\"");
s.Replace(' ', '|');
Console.WriteLine(s);
```

**Java**

```
String s = " six impossible things before breakfast ";
int i = s.indexOf('i');
System.out.print(i + " ");
System.out.println("\""+ s.trim() + "\"");
s.replace(' ', '|');
System.out.println(s);
```

А)

```
1 "six impossible things before breakfast"
 six impossible things before breakfast
```

Б)

```
2 "six impossible things before breakfast"
|six|impossible|things|before|breakfast|
```

В)

```
2 "six impossible things before breakfast"
 six impossible things before breakfast
```

Г)

```
2 "\six impossible things before breakfast\"
 six impossible things before breakfast
```

**7. Каква ще е стойността на променливата br след изпълнението на програмния код?**

**C#**

```
List<int> numbers = new List<int>();
for (int i = 1; i <= 1000; i++)
       numbers.Add(i);
int number = 23, br = 0;
int left = 0, right = numbers.Count - 1;
while (left <= right)
{
    br++;
    int mid = (left + right) / 2;
    if (numbers[mid] == number)
    {
        Console.WriteLine("Yes");
        break;
    }
    if (numbers[mid] < number) left = mid + 1;
    else right = mid - 1;
}
```

**Java**

```
List<Integer>numbers = new ArrayList<>();
for (int i = 1; i <= 1000; i++)
    numbers.add(i);
int number = 23, br = 0;
int left = 0, right = numbers.size() - 1;
while (left <= right)
{
    br++;
    int mid = (left + right) / 2;
    if (numbers.get(mid) == number)
    {
```

```
            System.out.println("Yes");
            break;
        }
        if (numbers.get(mid) < number) left = mid + 1;
        else right = mid - 1 ;
    }
```

А) 7 Б) 9 В) 10 Г) 23

**8. Нека класът B наследява класа A, а класът C наследява класа B. Кое от следните твърдения е вярно:**

А) Методите на класа **C** нямат никакъв достъп до компонентите на класа **A**.

Б) Методите на класа **C** имат достъп до protected и public компонентите на класа **B** и само до public компонентите на класа **A**.

В) Методите на класа **C** имат достъп до protected полетата на класа **A**.

Г) Методите на класа **C** имат достъп до всички компоненти на класа **B** и до protected и public компонентите на класа **A**.

**9. Какво ще се изведе на стандартния изход след изпълнението на дадения програмен фрагмент?**

**C#**

```
try
{
    int number = 2024;
    int result = number / (number % 2);
    Console.WriteLine("Message1");
}
catch (Exception e)
{
    Console.WriteLine("Message2");
}
finally
{
    Console.WriteLine("Message3");
}
Console.WriteLine("Message4");
```

**Java**

```
try {
   int number = 2024;
   int result = number / (number % 2);
   System.out.println("Message1");
}
catch (Exception e) {
   System.out.println("Message2");
}
```

```
finally {
   System.out.println("Message3");
}
System.out.println("Message4");
```

А)

```
Message1
Message4
```

Б)

```
Message1
Message3
Message4
```

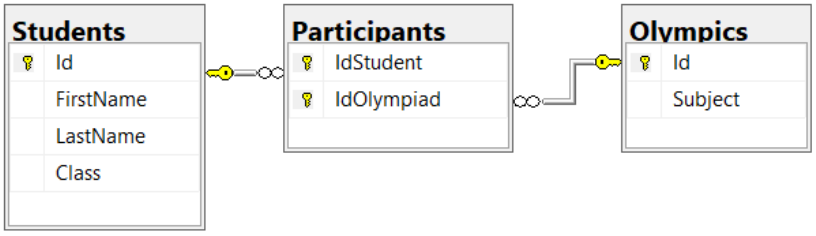
В)

```
Message2
Message3
Message4
```

Г)

```
Message2
Message3
```

**10. Каква е връзката между таблиците Students и Olympics в дадената диаграма на релационна база от данни?**




А) Едно към много

Б) Много към много

В) Много към едно

Г) Едно към едно

**11. Кое твърдение за модификаторите за достъп НЕ е вярно?**

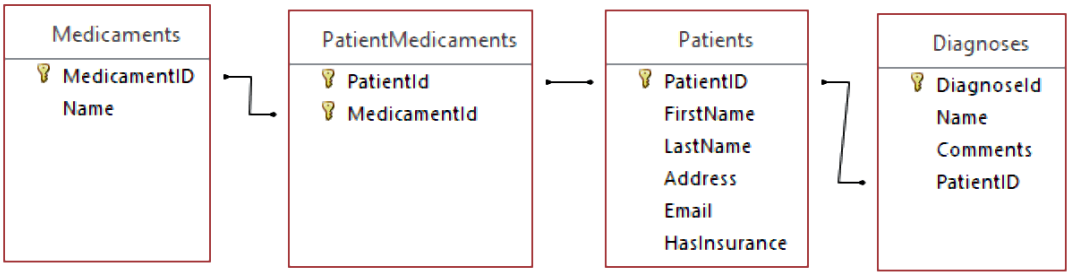
А) Интерфейсите **НЕ** съдържат модификатори за достъп

Б) Запазената дума `readonly` (за C#) / `final` (за Java) **НЕ Е** модификатор за достъп.

В) Модификаторите за достъп са основен начин за капсулиране на обект и скриване на данни от външния свят.

Г) Членове на базов клас с модификатор за достъп `private` могат да бъдат достъпни само от членове на класовете наследници.

**12. На диаграмата са представени таблици от база данни, съдържаща информация за лечението на пациенти в една болница.**




Кое от твърденията за отношенията между таблиците **НЕ** е вярно?

А) PatientId на таблицата Diagnoses е външен ключ.
Б) Връзката между таблиците Medicaments и Patients е много към много.
В) Връзката между таблиците Diagnoses и Patients е едно към едно.
Г) Таблицата PacientMedicaments има съставен първичен ключ.

**13. Какво ще се изведе на стандартния изход след изпълнение на програмния фрагмент?**

**C#**

```
int number = 2024;
Stack<int> stack = new Stack<int>();
stack.Push(1);
stack.Push(++number);
Console.Write(stack.Peek() + " ");
Console.Write(stack.Pop() + " ");
stack.Push((int)Math.Sqrt(number));
Console.Write(stack.Peek() + " ");
stack.Pop();
Console.Write(stack.Pop() + " ");
```

**Java**

```
int number = 2024;
Stack<Integer> stack = new Stack<>();
stack.push(1);
stack.push(++number);
System.out.print(stack.peek() + " ");
System.out.print(stack.pop() + " ");
stack.push((int)Math.sqrt(number));
System.out.print(stack.peek() + " ");
stack.pop();
System.out.print(stack.pop() + " ");
```

А) `1 45 2025 2025`
Б) `2025 2025 45 1`
В) `2024 2025 45 1`
Г) `1 45 2025 2024`

**14. Кое число ще върне като резултат извикването на дадения метод със стойност 7?**

```
static int RecFunc(int n)
    {   if (n == 0) return 0;
        return (n % 2) + RecFunc(n / 2);
    }
```

А) 0 Б) 1 В) 2 Г) 3

**15. Основните етапи на жизнения цикъл на информационна система са:**

А) анализ, проучване, разработване, документиране, обучение, тестване, експлоатация

Б) проучване, разработване, тестване, документиране, осигуряване на качеството, експлоатация и поддръжка

В) проучване, анализ, проектиране, разработване, тестване, експлоатация и поддръжка

Г) анализ, проучване, документиране, разработване, обучение, тестване и поддръжка

**16. Дадени са таблици Students и Profiles на релационна база от данни.**

**Students**

| Id | FirstName | LastName | Points | ProfileId |
|---|---|---|---|---|
| 1 | Иван | Иванов | 200 | 1 |
| 2 | Петър | Петров | 150 | 2 |
| 3 | Стефан | Стефанов | 280 | 3 |
| 4 | Симеон | Симеонов | 120 | 4 |
| 5 | Гергана | Георгиева | 250 | 3 |
| 6 | Таня | Тодорова | 180 | 1 |
| 7 | Петя | Попова | 300 | 2 |
| 8 | Ангел | Ангелов | 220 | 3 |
| 9 | Ани | Атанасова | 160 | 4 |
| 10 | Георги | Георгиев | 200 | 5 |

**Profiles**

| Id | Name |
|---|---|
| 1 | Математика |
| 2 | Математика и информатика |
| 3 | Информатика |
| 4 | Информатика и ИТ |
| 5 | Графичен дизайн |

**Колко записа ще изведе заявката:**

```
SELECT * FROM Students AS s
JOIN Profiles AS p ON s.ProfileId = p.Id
WHERE Name LIKE '%Информатика%'
```

А) 0 Б) 3 В) 5 Г) 7

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

17. В полетата за отговори срещу 1) и 2) запишете колко пъти ще се изпълни тялото на оператор за цикъл for, ако управляващата му променлива получава като стойност целите числа от интервала:

1) от 10 до 50 със стъпка 7

2) от **a** до **b** (a<b) със стъпка **step** (**step>0)**?

**18. В полето за отговор запишете какво ще се изведе на стандартния изход след изпълнението на дадения програмен фрагмент.**

**C#**

```
int n = 2143248129;
Stack<int> st = new Stack<int>();
Queue<int> q = new Queue<int>();
while (n != 0)
{
    int k = n % 10;
    if (k % 2 == 0)
       st.Push(k);
    else
       q.Enqueue(k);
    n = n / 10;
}
while (st.Count > 0)
{
    Console.Write(st.Pop());
}
Console.WriteLine();
while (q.Count > 0)
{
    Console.Write(q.Dequeue());
}
Console.WriteLine();
```

**Java**

```
int n = 2143248129;
Stack<Integer> st = new Stack<>();
Queue<Integer> q = new ArrayDeque<>();
while (n != 0) {
    int k = n % 10;
    if (k % 2 == 0) {
        st.push(k);
    } else {
        q.add(k);
    }
    n = n / 10;
}
while (!st.isEmpty()) {
    System.out.print(st.pop());
}
System.out.println();
while (!q.isEmpty()) {
```

```
    System.out.print(q.poll());
}
System.out.println();
```

**19. В полетата за отговори срещу (1), (2), (3) и (4) запишете какво ще се изведе на стандартния изход от редовете, означени със съответните номера, като резултат от изпълнението на следния програмен фрагмент?**

**C#**

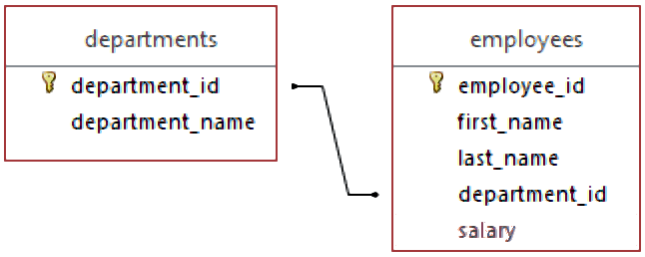
```
double[] arr1 = {3.14, 5.6, -7.2, 2.14, -2.7};
for (int i = 0; i < arr1.Length; i++)
{
    if (i % 2 == 0)
    {
        arr1[i] = Math.Floor(arr1[i]);
        Console.Write(arr1[i] + " "); // (1)
    }
}
Console.WriteLine();
for (int i = 0; i < arr1.Length; i++)
{
    if (i % 2 != 0)
    {
        arr1[i] = Math.Ceiling(arr1[i]);
        Console.Write(arr1[i] + " "); // (2)
    }
}
Console.WriteLine();
int[] arr2 = {-3, 0, -16, 100, 9, -1 };
Array.Sort(arr2);
Console.WriteLine("[" + string.Join(", ", arr2) + "]"); //
(3)
Console.WriteLine(Math.Sqrt(arr2[2])); // (4)
```

**Java**

```
double[] arr1 = {3.14, 5.6, -7.2, 2.14, -2.7};
for (int i = 0; i < arr1.length; i++)
{
    if (i % 2 == 0)
    {
        arr1[i] = Math.floor(arr1[i]);
        System.out.print(arr1[i] + " "); // (1)
    }
}
System.out.println();
for (int i = 0; i < arr1.length; i++)
{
    if (i % 2 != 0)
    {
        arr1[i] = Math.ceil(arr1[i]);
        System.out.print(arr1[i] + " "); // (2)
    }
```

```
}
System.out.println();
int[] arr2 = {-3, 0, -16, 100, 9, -1 };
Arrays.sort(arr2);
System.out.println(Arrays.toString(arr2)); // (3)
System.out.println(Math.sqrt(arr2[2])); // (4)
```

**20. Дадената диаграма представя база данни, съдържаща информация за служителите и отделите в една фирма.**




**В заявката по-долу три места са оставени празни, като са обозначени с (1), (2) и (3). В полетата за отговор срещу (1), (2) и (3) запишете пропуснатото в заявката, така, че тя да извежда employee_id, first_name, salary, department_name, сортирани по first_name в азбучен ред, като филтрира само служители със заплата по-голяма от 15000.**

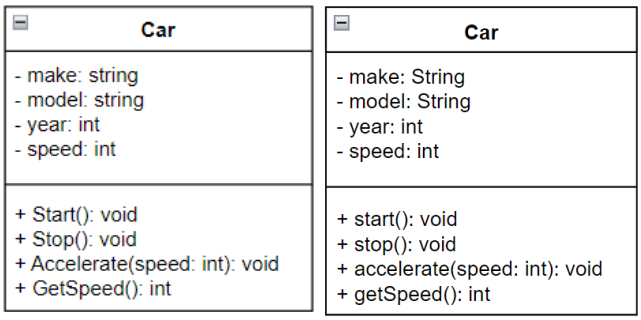
```
SELECT e.employee_id, e.first_name, e.salary, d.name AS deparment_name
FROM employees AS e
INNER JOIN departments AS d .....(1).....
.....(2)..... salary > 15000
.....(3)..... e.first_name;
```

**21. Дадена е UML диаграма на класа Car (C# и Java). В полетата за отговор срещу (1), (2) и (3) запишете:**

**(1) – декларация на полето make**

**(2) – заглавен ред на метода Accelerate (за C#) /accelerate (за Java)**

**(3) – заглавен ред на метода GetSpeed (за C#) / getSpeed (за Java)**




**22. В състезание участват 10 участници с номера от 1 до 10. Техните резултати са запазени в масива points. Даденият програмен фрагмент трябва да изведе на стандартния изход номерата и точките на участниците, чиито места в класирането съвпадат с техните номера. Класирането се извършва във възходящ ред по точките на участниците. На първо място се класира участникът с най-малко точки, а на последно с най-много точки.**

**В кода има пропуснати части, номерирани с (1), (2) и (3). В полетата за отговори срещу (1), (2) и (3) запишете пропуснатия код, така, че да се получи вярно решение.**

**C#**

```
int[] points = new int[] { 35, 28, 21, 16, 18, 11, 25, 14, 33, 17 };
int[] sortedPoints = new int[...(1)....Length];
Array.Copy(points, sortedPoints, points.Length);
Array.Sort(sortedPoints);
for (int i = 0; i < points.Length; i++)
{
      if (i == Array.BinarySearch(...(2)..., points[i]))
      {
           Console.WriteLine($"Number {...(3)...}: {points[i]}
points");
      }
}
```

**Java**

```
int[] points = {35, 28, 21, 16, 18, 11, 25, 14, 33, 17};
int[] sortedPoints = Arrays.copyOf(...(1)..., points.length);
Arrays.sort(sortedPoints);
for (int i = 0; i < points.length; i++)
 {
   if (i == Arrays.binarySearch(...(2)..., points[i]))
   {
        System.out.println(String.format("Number %d: %d points",
           ...(3)..., points[i]));
   }
}
```

**23. Методът LinearSearch (за C#) linearSearch (за Java) реализира линейно търсене в масив, Ако елемент target се съдържа в масива, методът връща позицията на елемента в масива, броена от 1. Ако елементът не се среща в масива, връща стойност 0.**

**В програмния код са допуснати грешки. Открийте ги и в полето за отговор запишете на кой ред са и след номера на реда запишете верния код, така че методът да работи коректно.**

**C#**

```
1  static int LinearSearch (int[] arr, int target) {
2      for (int i = 0; i < arr.Length(); i++) {
3          if (arr[i] == target)
4                 return i;
5      }
6      return 1;
7  }
```

**Java**

```
1  static int linearSearch (int[] arr, int target) {
2       for (int i = 0; i < arr.length(); i++) {
3           if (arr[i] == target)
4                   return i;
5       }
6       return 1;
7  }
```

**24. Даденият програмен фрагмент трябва да намира и извежда на стандартния изход най-малката стойност на елемент от масива arr. В кода са допуснати синтактични грешки. Открийте ги и в полето за отговор запишете номера на реда и верния код, така, че кодът да е синтактично коректен.**

**C#**

```
1  int[] arr = new [25, -3, 10, -1, 40, -12];
2  int minEl = arr;
3  for (int i, i < arr.Length, i++) {
4       if (arr[i] <= minEl)
5           minEl = arr[i];
6  }
7  Console.WriteLine(minEl);
```

**JAVA**

```
1  int [] arr = new [25, -3, 10, -1, 40, -12];
2  int minEl = arr;
3  for (int i, i < arr.length, i++){
4       if(arr[i] <= minEl)
5          minEl = arr[i];
6  }
7  System.out.println(minEl);
```

**25. ~~Дадено е съдържанието на~~ текстов файл input.txt, ~~който~~ съдържа думи, записани на отделни редове.**

**input.txt**

```
juice
chocolate
tea
coffee
milk
water
soda
```

В полето за отговор напишете какво ще е съдържанието на файла **output.txt**, след изпълнението на програмния фрагмент. Гарантирано е, че файлът **input.txt** съществува и се чете коректно.

**C#**

```
            StreamReader reader = new StreamReader("input.txt");
```

```
            StreamWriter writer = new StreamWriter("output.txt");
             try
             {
                 int num = 1;
                 while (true)
                 {
                     string inp = reader.ReadLine();
                     if (inp == null) break;
                     if (num % 2 == 0)
                     {
                         string newLine = ${num}. {inp}";
                         writer.WriteLine(newLine);
                     }
                     num++;
                 }
               reader.Close();
               writer.Close();
            }
              catch (Exception e)
              {
                  Console.WriteLine(e.Message);
              }
        }
    }
```

**JAVA**

```
try {
     Scanner reader = new Scanner (new FileInputStream ("input.txt"));
     PrintWriter writer = new PrintWriter ("output.txt");
     int num = 1;
     while (true) {
         if (!reader.hasNext ()) break;
         String inp = reader.nextLine ();
         if (num % 2 == 0) {
             String newLine = num + ". " + inp;
             writer.println(newLine);
         }
         num++;
     }
     reader.close ();
     writer.close ();
}
catch (Exception ex) {
     System.out.println (ex.getMessage ());
}
```

**МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА**

**ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ ПО**

**ИНФОРМАТИКА**

**20 май 2024 г.**

**ПРОФИЛИРАНА ПОДГОТОВКА**

**ВАРИАНТ 1**

**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!**

**26.** Дадени са две цели числа ***a*** и ***b*** ($0 < a < b < 10^6$). Обхождат се последователно всички цели числа от ***a*** до ***b*** включително. Коя цифра се среща най-много пъти в записа на числата и колко пъти се среща тя?

Създайте приложение с име ***Zad26***, което въвежда от стандартния вход две цели числа ***a*** и ***b***, намира и извежда на стандартния изход цифрата, която се среща най-много пъти в записа на числата от интервала [***a***, ***b***], както и броя срещания на цифрата. Ако има две и повече цифри, които се срещат максимален брой пъти, да се изведе най-малката от тях.

Числата ***a*** и ***b*** се въвеждат на отделни редове.

На стандартния изход се извежда търсената информация в следния формат:
Digit <*цифра*> - <*брой срещания*> times

**Забележка*: Приемат се и решения с графичен потребителски интерфейс (ГПИ), в които целите числа a и b се въвеждат в подходящи контроли, както и резултата се извежда в подходяща контрола.

**Пример:**

| **Вход 1:**<br>10<br>20 | **Изход 1:**<br>`Digit 1 - 11 times` |
|---|---|
| **Вход 2:**<br>2024<br>3027 | **Изход 2:**<br>`Digit 2 - 1280 times` |
| **Вход 3:**<br>454<br>455 | **Изход 3:**<br>`Digit 4 - 3 times` |

*Прикачете в изпитната система архив с име **inf_20.05.2024_ zad26.zip**, съдържащ файловете с Вашите решения.*

**27.** Създайте база от данни с име **Geo**, която съхранява данни за планини и върхове.

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

| Име на колоната | Тип на данните | Ограничения |
|---|---|---|
| **ID** | **цяло число** | автоматично се увеличава с 1 |
| **MountainName** | **текст на кирилица** до **20** символа | задължително се посочва стойност |
| **CountryCode** | **текст на латиница** с **точно 3** символа | задължително се посочва стойност |
| **Country** | **текст на кирилица** до **20** символа | |

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

| Име на колоната | Тип на данните | Ограничения |
|---|---|---|
| **ID** | **цяло число** | автоматично се увеличава с 1 |
| **PeakName** | **текст на кирилица** до **20** символа | задължително се посочва стойност |
| **Elevation** | **цяло число** | задължително се посочва стойност, проверка дали е положително число |
| **MountainId** | **цяло число** | задължително се посочва стойност |

В) Напишете заявка за добавяне в таблица **Mountains** на следните кортежи (данните за таблицата са в ресурсен файл с име resources.txt):

| ID | MountainName | CountryCode | Country |
|---|---|---|---|
| 1 | Рила | BUL | България |
| 2 | Пирин | BUL | България |
| 3 | Стара планина | BUL | България |
| 4 | Анди | ARG | Аржентина |
| 5 | Анди | CHL | Чили |
| 6 | Хималаи | NPL | Непал |
| 7 | Алпи | SUI | Швейцария |
| 8 | Алпи | ITA | Италия |
| 9 | Алпи | AUT | Австрия |
| 10 | Алпи | FRA | Франция |
| 11 | Елбрус | RUS | Русия |
| 12 | Елбрус | GEO | Грузия |

Г) Напишете заявка за добавяне в таблица **Peaks** на следните кортежи (данните за таблицата са в ресурсен файл с име resources.txt):

| ID | PeakName | Elevation | MountainId |
|---|---|---|---|
| 1 | Аконкагуа | 6962 | 4 |
| 2 | Ботев | 2376 | 3 |
| 3 | Мусала | 2925 | 1 |
| 4 | Еверест | 8849 | 6 |
| 5 | Вихрен | 2914 | 2 |
| 6 | Мальовица | 2729 | 1 |
| 7 | Монблан | 4809 | 10 |
| 8 | Матерхорн | 4478 | 8 |
| 9 | Дюфур | 4634 | 7 |
| 10 | Елбрус | 5642 | 11 |
| 11 | Ком | 2015 | 3 |
| 12 | Манаслу | 2729 | 6 |
| 13 | Дено | 2790 | 1 |

Д) Точната височина на връх Ком е 2015,8 метра. Напишете заявка, която променя данните в колона Elevation от 2015 на 2016 в кортежа за връх Ком.

Е) Напишете заявка, която намира и извежда средната височина на върховете в планината с ID 1.

Ж) Напишете заявка, която намира и извежда броя на върховете с височина между 5000 и 9000 метра включително.

З) Напишете заявка, която извежда за върховете с височина над 2900 метра следните данни: *име* (PeakName), *височина* (Elevation), *планина* (MountainName), *страна* (CountryName), подредени по името на страната в азбучен ред, а при една и съща страна – по височина в намаляващ ред.

И) Напишете заявка, която намира и извежда за всяка държава броя на планините в нея. Кортежите в резултатната таблица да са подредени по броя на планините в намаляващ ред, а при еднакъв брой – по азбучен ред на имената на държавите.

*Прикачете в изпитната система файл с име **inf_20.05.2024_ zad27.zip**, съдържащ създадената от Вас база от данни и написаните заявки.*

**28.** Създайте приложение с име **Zad28**, което имплементира следните класове:

А) Клас **Person** със следните полета:

**- firstName** – собствено име (символен низ)

**- lastName** – фамилно име (символен низ)

**- id** – уникален идентификатор (символен низ с дължина 10 знака)

Класът трябва да притежава следните методи:

- **конструктор** с параметри, който задава стойности на всички полета.

Класът трябва да предоставя възможност за валидиране на полето **id**. Ако за **id** се зададе низ с дължина различна от 10 символа, да се генерира изключение със съобщение: „<*собствено име*> <*фамилно име*> - invalid identifier!".

Б) Клас **Kid** – наследник на Person със собствени полета:

- **age** – възраст (цяло число)
- **group** – група, в която се записва детето според възрастта му (низ)
- **parentLastName** – фамилия на родител (низ)
- **parentGSM** – телефон на родител (низ)

Класът да притежава следните методи:

- **конструктор** с параметри, който задава стойности на всички характеристики (собсвено име, фамилно име, идентификационен номер, възраст, фамилия на родителя и телефон на родител).

Групата, в която се записва детето, НЕ се въвежда, а се определя според възрастта му: I група – 3 години, II група – 4 години, III група – 5 години и IV група – 6 години.

– предефиниран метод **ToString()** (за C#) / **toString()** (за Java), който да връща низа: "<*собствено име*> <*фамилно име*>, <*възраст*>, <*телефон на родител*> (<*фамилия на родител*>)"

Класът трябва да предоставя възможност за валидиране на полето **age** и ако НЕ е от 3 до 6 години включително, да се генерира изключение със съобщението: „The child <*собствено име*> <*фамилия*> age is invalid - <*възраст*>".

В) Клас **KinderGarden** със следните полета:

- **kidList** – списък с обекти от клас **Kid**, съдържащ децата в детската градина

Класът трябва да притежава следните методи:

- **конструктор** за създаване на празен списък на децата в детската градина

– метод **EnrollKid** (за C#) / **enrollKid** (за Java) за записване на дете в градината, като добавя обект от клас **Kid** в списъка и извежда на стандартния изход съобщението: "The child <*собствено име*> <*фамилно име*> is enrolled."

– метод **ReleaseKid** (за C#) / **releaseKid** (за Java) премахва от списъка на децата в детската градина дете с даден идентификатор, ако такова има, и извежда на стандартния изход съобщението: „The child <*собствено име*> <*фамилно име*> has been unsubscribed." Ако няма дете с такъв идентификатор, извежда на стандартния изход съобщението: „Unsubscribe failed - invalid identifier <*идентификатор*>."

– метод **GroupInfo** (за C#)/ **groupInfo** (за Java) извежда информация за дадена група във формат:

На първия ред се извежда: <*групата*> group - <*брой деца*> children.

На следващите редове се извежда информация за всяко записано дете от групата, подредени в азбучен ред по име и фамилия. Информацията за всяко дете се извежда на отделен ред във вида:
<*собствено име*> <*фамилно име*>, <*възраст*>, <*телефон на родител*> (<*фамилия на родител*>)

Г) Приложението трябва да прочете от текстов файл **data.txt** поредица от команди и да изведе на стандартния изход резултата от изпълнението им. Всеки ред на файла представлява една от следните команди:

- команда за **записване** на дете в детската градина, в следния формат:

enrollment <*собствено име*> <*фамилно име*> <*идентификатор*> <*възраст*> <*фамилно име на родител*> <*телефон на родител*>

- команда за **отписване** на дете от детската градина, в следния формат:

unsubscribe <*идентификатор*>

- команда за **извеждане** на информация за посочена група, в следния формат:

information <*група*>

- команда за край на информацията, в следния формат:

END

Ако във файла има и други команди, различни от посочените, да се изведе съобщение: „<*прочетената команда*> - invalid command."

Приложението трябва да прихваща и обработва предизвиканите изключения.

При приключване работата на приложението на стандартния изход да се изведе съобщение: „Have a nice day!"

*Пример с използване на ресурсния файл с име data.txt:*

**Изход:**

```
The child Iva Teneva is enrolled.
The child Maria Petrova is enrolled.
The child Petar Yanev is enrolled.
The child Anna Nikolova is enrolled.
The child Georgi Stoev is enrolled.
The child Marta Dobreva is enrolled.
The child Nikola Ivanov is enrolled.
Nikolay Dimitrov - invalid identifier!
The child Mila Popova age is invalid - 7
The child Ilia Nikolov is enrolled.
The child Stela Petkova is enrolled.
I group - 3 children
Anna Nikolova, 3, 0888333444 (Veleva)
Nikola Ivanov, 3, 0888999111 (Ivanova)
Stela Petkova, 3, 0888666777 (Yankova)
II group - 3 children
Ilia Nikolov, 4, 0888444555 (Nikolov)
Marta Dobreva, 4, 0888777999 (Dobrev)
Petar Yanev, 4, 0888111222 (Yanev)
```

```
Unsubscribe failed - invalid identifier 2102232455.
The child Stela Petkova has been unsubscribed.
I group - 2 children
Anna Nikolova, 3, 0888333444 (Veleva)
Nikola Ivanov, 3, 0888999111 (Ivanova)
Info - invalid command
Have a nice day!
```

*Прикачете в изпитната система архив с име inf_20.05.2024_* ***zad28.zip****, съдържащ файловете с Вашите решения.*

**МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА**

# ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ ПО ИНФОРМАТИКА

**20 май 2024 г.**

**ПРОФИЛИРАНА ПОДГОТОВКА**

**ВАРИАНТ 1**

Задача от 1. до 16. Ключ с верните отговори

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

**Задача 17. – 2 точки**

А) `6`

Б) `(b-a)/step+1`

**Задача 18. – 2 точки**

```
242482
```

```
9131
```

**Задача 19. – 2 точки**

```
(1)     3 -8 -3 или
   3.0 -8.0 -3.0
```

(2) 6 3 или

6.0 3.0

(3) [-16, -3, -1, 0, 9, 100]

(4) NaN или друг отговор в смисъл, че „не може да се намери квадратен корен от отрицателно число“ или да се приложи Math.Sqrt() върху отрицателен аргумент.

**Задача 20. – 3 точки**

```
(1) ON d.department_id = e.department_id
(2) WHERE
(3) ORDER BY
```

**Задача 21. – 3 точки**

```
(1) private string make (за C#) / private String make (за Java)
(2) public void Accelerate (int speed)   (за C#)
      public void accelerate (int speed)   (за Java)
(3) public int GetSpeed () (за C#) / public int getSpeed () (за
Java)
```

**Задача 22. – 3 точки**

```
(1) points
(2) sortedPoints
(3) i + 1
```

**Задача 23. – 3 точки**

```
ред 2 for (int i = 0; i < arr.Length; i++) или arr.Length (за C#)
      for (int i = 0; i < arr.length; i++) или arr.length (за Java)
ред 4 return i+1; или i+1
ред 5 return 0; или 0
```

**Задача 24. – 3 точки**

```
1  int [] arr = {25, -3, 10, -1, 40, -12};

2  int maxEl = arr[0];
```

Признава се и задаването на който и да е друг индекс на елемент от масива. Не се признава за вярна редакция, ако `arr` се замени с конкретна стойност

```
3  for (int i = 1 ; i < arr.Length; i++) (за C#)

   for (int i = 1 ; i < arr.length; i++) (за Java)
```

```
   4      if(arr[i] >= maxEl)
```

**Задача 25. – 3 точки**

```
2. chocolate
4. coffee
6. water
```

**Задача 26 – 15 точки**

Примерно решение

**C#**

```
using System;

namespace DZI
{
    class Zad26
    {
        static void Main(string[] args)
        {
            int a = int.Parse(Console.ReadLine());
            int b = int.Parse(Console.ReadLine());
            int[] count = new int[10];
            for (int i = a; i <= b; i++)
            {
                int x = i;
                while (x != 0)
                {
                    int c = x % 10;
                    count[c]++;
                    x = x / 10;
                }
            }
            int max = count[0];
            int digit = 0;
            for (int i = 1; i < 10; i++)
            {
                if (count[i] > max)
                {
                    max = count[i];
                    digit = i;
                }
            }
            Console.WriteLine($"Digit {digit} - {max} times");
        }
    }
}
```

**Java**

```
package DZI;

import java.util.Scanner;

public class Zad26 {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int a = scanner.nextInt();
        int b = scanner.nextInt();
        int[] count = new int[10];

        for (int i = a; i <= b; i++) {
            int x = i;
            while (x != 0) {
                int c = x % 10;
                count[c]++;
                x = x / 10;
            }
        }

        int max = count[0];
        int digit = 0;
        for (int i = 1; i < 10; i++) {
            if (count[i] > max) {
                max = count[i];
                digit = i;
            }
        }
        System.out.printf("Digit %d - %d times%n", digit, max);
    }
}
```

**Задача 27 – 20 точки**

Примерно решение

```
CREATE DATABASE Geo;
USE Geo;
--- A)
CREATE TABLE Mountains (
ID INT NOT NULL IDENTITY PRIMARY KEY,
```

```
MountainName NVARCHAR(20) NOT NULL,
CountryCode CHAR(3) NOT NULL,
Country NVARCHAR(20));

--- Б)
CREATE TABLE Peaks (
ID INT NOT NULL IDENTITY PRIMARY KEY,
PeakName NVARCHAR(20) NOT NULL,
Elevation INT NOT NULL,
MountainId INT NOT NULL REFERENCES Mountains(ID),
CHECK (Elevation > 0));

--- В)
INSERT INTO Mountains (MountainName,CountryCode, Country) VALUES
('Рила','BUL','България'),
('Пирин','BUL','България'),
('Стара планина','BUL','България'),
('Анди','ARG','Аржентина'),
('Анди','CHL','Чили'),
('Хималаи','NPL','Непал'),
('Алпи','SUI','Швейцария'),
('Алпи','ITA','Италия'),
('Алпи','AUT','Австрия'),
('Алпи','FRA','Франция'),
('Елбрус','RUS','Русия'),
('Елбрус','GEO','Грузия');

--- Г)
INSERT INTO Peaks VALUES
('Аконкагуа',6962,4),
('Ботев',2376,3),
('Мусала',2925,1),
('Еверест',8849,6),
('Вихрен',2914,2),
('Мальовица',2729,1),
('Монблан',4809,10),
('Матерхорн',4478,8),
('Дюфур',4634,7),
('Елбрус',5642,11),
('Ком',2015,3),
('Манаслу',2729,6),
('Дено',2790,1);

--- Д)
UPDATE Peaks
SET Elevation = 2016
WHERE PeakName = 'Ком';
```

```
--- Е)
SELECT AVG(Elevation) FROM Peaks
WHERE MountainId = 1;

--- Ж)
SELECT COUNT(ID) FROM Peaks
WHERE Elevation BETWEEN 5000 AND 9000;

--- З)
SELECT p.PeakName, p.Elevation, m.MountainName, m.Country FROM Peaks AS p
INNER JOIN Mountains AS m ON p.MountainId=m.ID
WHERE Elevation > 2900
ORDER BY Country, Elevation DESC;

--- И)
SELECT Country, COUNT(*) AS 'Count' FROM Mountains
GROUP BY Country
ORDER BY COUNT(*) DESC, Country;
```

**Задача 28 – 25 точки**

Примерно решение

**C#**

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

namespace Zad28
{
    class Zad28
    {
        static void Main(string[] args)
        {
            using(StreamReader reader = new StreamReader("data.txt"))
            {
                KinderGarden kinderGarden = new KinderGarden();
                string cmd;
                while ((cmd = reader.ReadLine()) != "END")
                {
                    string[] cmdArgs = cmd.Split();
                    Kid kid;
                    switch (cmdArgs[0])
                    {
                        case "enrollment":
                            try
                            {
```

```
                                kid = new Kid(cmdArgs[1], cmdArgs[2],
cmdArgs[3], int.Parse(cmdArgs[4]), cmdArgs[5], cmdArgs[6]);
                                kinderGarden.EnrollKid(kid);
                            }
                            catch (Exception e)
                            {
                                Console.WriteLine(e.Message);
                            }
                            break;
                        case "unsubscribe":
                            kinderGarden.ReleaseKid(cmdArgs[1]);
                            break;
                        case "information":
                            kinderGarden.GroupInfo(cmdArgs[1]);
                            break;
                        default:
                            Console.WriteLine($"{cmdArgs[0]} - invalid
command");
                            break;
                    }
                }
                Console.WriteLine("Have a nice day!");
            }
        }
    }

class Person
    {
        private string id;
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string ID
        {
            get { return id; }
            set
            {
                if (value.Length != 10)
                {
                    throw new Exception($"{FirstName} {LastName} - invalid
identifier!");
                }
                id = value;
            }
        }
        public Person (string firstName, string lastName, string id)
        {
            FirstName = firstName;
            LastName = lastName;
            ID = id;
        }
    }

class Kid : Person
    {
        private int age;
```

```
        public int Age
        {
            get { return age; }
            set
            {
                if (value < 3 || value > 6)
                {
                    throw new Exception($"The child {FirstName} {LastName} age
is invalid - {value}");
                }
                age = value;
            }
        }
        public string Group { get; set; }
        public string ParentLastName { get; set; }
        public string ParentGSM { get; set; }
        public Kid(string firstName, string lastName, string id, int age, string
parentLastName, string parentGSM)
            : base(firstName, lastName, id)
        {
            Age = age;
            switch (age)
            {
                case 3:
                    Group = "I";
                    break;
                case 4:
                    Group = "II";
                    break;
                case 5:
                    Group = "III";
                    break;
                case 6:
                    Group = "IV";
                    break;
            }
            ParentLastName = parentLastName;
            ParentGSM = parentGSM;
        }
        public override string ToString()
        {
            return $"{FirstName} {LastName}, {Age}г., {ParentGSM}
({ParentLastName})";
        }

    }

class KinderGarden
    {
        private List<Kid> kidList;
        public KinderGarden()
        {
            kidList = new List<Kid>();
        }
        public void EnrollKid(Kid kid)
```

```
        {
            kidList.Add(kid);
            Console.WriteLine($"The child {kid.FirstName} {kid.LastName} is
enrolled.");
        }

        public void ReleaseKid(string id)
        {
            Kid kid = kidList.Find(k => k.ID == id);
            if (kid != null)
            {
                kidList.Remove(kid);
                Console.WriteLine($"The child {kid.FirstName} {kid.LastName} has
been unsubscribed.");
            }
            else Console.WriteLine($"Unsubscribe failed - invalid identifier
{id}.");
        }
        public void GroupInfo(string group)
        {
            int count = kidList.Count(k => k.Group == group);
            Console.WriteLine($"{group} group - {count} children");
            foreach (var kid in kidList.OrderBy(k => k.FirstName).ThenBy(k =>
k.LastName))
            {
                if (kid.Group == group)
                {
                    Console.WriteLine(kid);
                }
            }
        }

    }
}
```

**Java**

```
package dzi;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.*;

public class Zad28 {
    public static void main(String[] args) throws FileNotFoundException {
        try(Scanner reader = new Scanner(new FileInputStream("data.txt"))){
            KinderGarden kinderGarden = new KinderGarden();
            String cmd;
            while(reader.hasNext()){
                cmd = reader.nextLine();
                if(cmd.equals("END")) {
                    break;
                }
                String[] cmdArg = cmd.split(" ");
```

```
                switch (cmdArg[0]){
                    case "enrollment":
                        try{
                            Kid kid = new Kid(cmdArg[1], cmdArg[2], cmdArg[3],
Integer.parseInt(cmdArg[4]), cmdArg[5], cmdArg[6]);
                            kinderGarden.enrollKid(kid);
                        }
                        catch (Exception e){
                            System.out.println(e.getMessage());
                        }
                        break;
                    case "unsubscribe":
                        kinderGarden.releaseKid(cmdArg[1]);
                        break;
                    case "information":
                        kinderGarden.groupInfo(cmdArg[1]);
                        break;
                    default:
                        System.out.println(String.format("%s - invalid command
", cmdArg[0]));
                        break;
                }
            }
            System.out.println("Have a nice day!");
        }
    }
}

public class Person {

    private String firstName;
    private String lastName;
    private String id;

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getId() {
        return id;
    }
```

```
    public void setId(String id) throws Exception {
        if (id.length() != 10) {
            throw new Exception(String.format("%s %s - invalid identifier!",
getFirstName(), getLastName()));
        } else {
            this.id = id;
        }
    }

    public Person(String firstName, String lastName, String id) throws Exception
{
        this.firstName = firstName;
        this.lastName = lastName;
        this.setId(id);
    }
}

public class Kid extends Person {

    private int age;
    private String group;
    private String parentLastName;
    private String parentGSM;

    public int getAge() {
        return age;
    }

    public void setAge(int age) throws Exception {
        if (3 <= age && age <= 6) {
            this.age = age;
        } else {
            throw new Exception(String.format("The child %s %s age is invalid -
%d", getFirstName(), getLastName(), age));
        }
    }

    public String getGroup() {
        return group;
    }

    public void setGroup() {
        switch (getAge()) {
            case 3:
                this.group = "I";
                break;
            case 4:
                this.group = "II";
                break;
            case 5:
                this.group = "III";
                break;
            case 6:
                this.group = "IV";
                break;
```

```
        }
    }

    public String getParentLastName() {
        return parentLastName;
    }

    public void setParentLastName(String parentLastName) {
        this.parentLastName = parentLastName;
    }

    public String getParentGSM() {
        return parentGSM;
    }

    public void setParentGSM(String parentGSM) {
        this.parentGSM = parentGSM;
    }

    public Kid(String firstName, String lastName, String id, int age, String
parentLastName, String parentGSM) throws Exception {
        super(firstName, lastName, id);
        this.setAge(age);
        this.parentLastName = parentLastName;
        this.parentGSM = parentGSM;
        this.setGroup();
    }

    @Override
    public String toString() {
        return String.format("%s %s, %d, %s (%s)", getFirstName(),
getLastName(), getAge(), getParentGSM(), getParentLastName());
    }

}

public class KinderGarden {

    private List<Kid> kidList;

    public KinderGarden() {
        this.kidList = new ArrayList<>();
    }

    public void enrollKid(Kid kid) {
        kidList.add(kid);
        System.out.println(String.format("The child %s %s is enrolled.",
kid.getFirstName(), kid.getLastName()));
    }

    public void releaseKid(String id) {
        int i = 0;
        while ( i < kidList.size() && !kidList.get(i).getId().equals(id)) {
            i++;
        }
```

```
        if (i == kidList.size()) {
            System.out.println("Unsubscribe failed - invalid identifier " + id +
".");
        } else {
            System.out.println(String.format("The child %s %s has been
unsubscribed.", kidList.get(i).getFirstName(), kidList.get(i).getLastName()));
            kidList.remove(i);
        }
    }
    Comparator<Kid> compareByName = new Comparator<Kid>() {
        @Override
        public int compare(Kid a, Kid b) {
            if (a.getFirstName().equals(b.getFirstName())) {
                return a.getLastName().compareTo(b.getLastName());
            } else {
                return a.getFirstName().compareTo(b.getFirstName());
            }
        }
    };

    public void groupInfo(String group) {
        int cnt = 0;
        for (Kid el : kidList) {
            if (el.getGroup().equals(group)) {
                cnt++;
            }
        }
        Collections.sort(kidList, compareByName);
        System.out.println(String.format("%s group - %d children", group, cnt));
        for (Kid el : kidList) {
            if (el.getGroup().equals(group)) {
                System.out.println(el.toString());
            }
        }
    }
}
```