МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ ПО ИНФОРМАТИКА ПРОФИЛИРАНА ПОДГОТОВКА

23 август 2024 г.

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. Композиция (`composition`) в обектно ориентираното програмиране означава, че:

   А) полетата на класа трябва да бъдат декларирани като частни (`private`)
   Б) един клас наследява друг клас
   В) променлива от родителския клас се отнася до обект от наследяващия клас
   Г) даден клас съдържа поле, което е обект от друг клас

2. Коя от представените UML диаграми илюстрира класа, дефиниран с дадения код?

C#

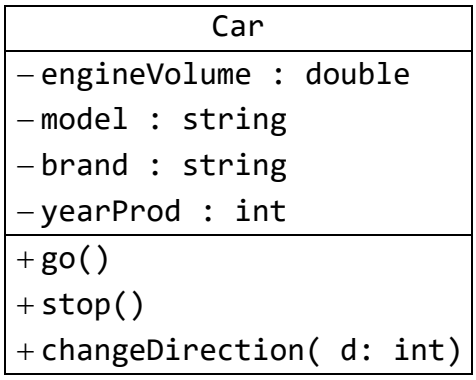
```
class Car {
    private int engineVolume;
    private int yearProd;
    private string model;
    private string brand;

    public void go() {}
    public void stop() {}
    public void changeDirection(int d){}
}
```

JAVA

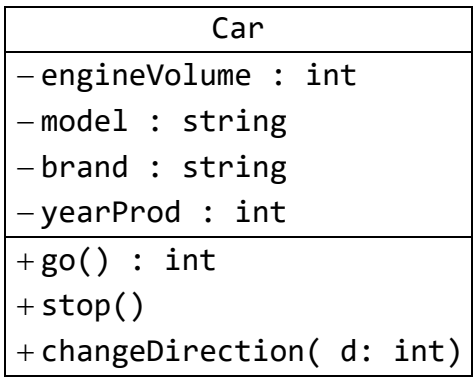
```
class Car {
    private int engineVolume;
    private int yearProd;
    private String model;
    private String brand;
```

```
    public void go() {}
    public void stop() {}
    public void changeDirection(int d){}
}
```

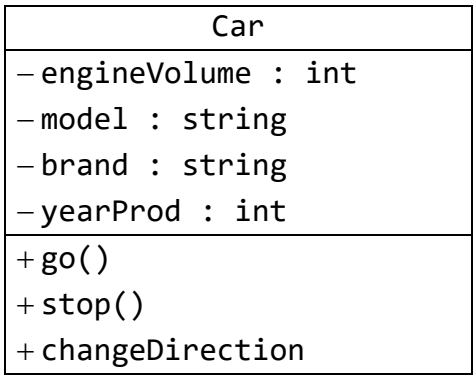
А)



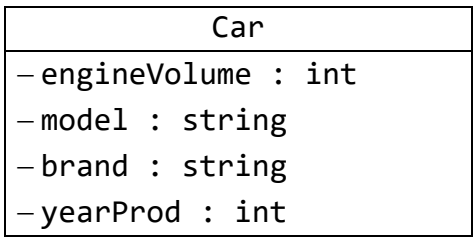
Б)



В)



Г)



+ go() + stop() + changeDirection(d: int)

3. Колко пъти ще се изпълни следващия цикъл и кое е последното изведено число?

C#	**JAVA**

int i = 5;
while (i <= 5)
{
Console.WriteLine(i);
i += 2;
}
	int i = 5;
	while (i <= 5) {
	System.out.println(i);
	i += 2;
	}

А) Цикълът е безкраен

Б) 1 път, последното изведено число е 7

В) 0 пъти, не се извежда нищо

Г) 1 път, последното изведено число е 5

4. Какъв ще бъде резултатът след изпълнение на програмния код?

C#

```
public class Student
{
    static int studentG;
    private string name;
    private int group;

    public Student(string name, int group)
    {
        this.name = name;
        this.group = group;
        studentG++;
    }
    public static void Main()
    {
        Student studentA = new Student("George", 3);
        Student studentB = new Student("Miranda", 4);
        Student studentC = new Student("Alex", 1);
        Student studentD = new Student("Dalia", 2);

        Console.WriteLine(Student.studentG);
    }
}
```

Java

```
public class Student {
    static int studentG;
    private String name ;
    private int group;
```

```
    public Student(String name, int group) {
        this.name = name;
        this.group = group;

        studentG++;
    }

    public static void main(String[] args) {
        Student studentA = new Student("George", 3);
        Student studentB = new Student("Miranda", 4);
        Student studentC = new Student("Alex", 1);
        Student studentD = new Student("Dalia", 2);

        System.out.println(Student.studentG);
    }
}
```

А) 1 Б) 2 В) 3 Г) 4

5. На кой ред запазените думи са само модификатори за достъп?

А) `public, class, final`

Б) `public, static, protected`

В) `private, protected, public`

Г) `private, overload, public`

6. Даденият програмен код трябва да извежда средноаритметичното на стойностите на масива. На кой ред от програмния фрагмент е допусната грешка?

C#

```
1 int[] numbers = { 5, -10, 17, -20, 25 };
2 int sum = 0;
3 foreach (int number in numbers) {
4     sum += number;
5 }
6 double average = sum / numbers.Length;
7 Console.WriteLine("Average: {0:F2}", average);
```

Java

```
1 int[] numbers = { 5, -10, 17, -20, 25 };
2 int sum = 0;
3 for (int number : numbers) {
4     sum += number;
5 }
6 double average = sum / numbers.length;
7 System.out.println(String.format("Average: %.2f", average));
```

А) На ред 6
Б) На ред 7
В) Няма грешка в кода
Г) На ред 3

7. Коя от фигурите ще бъде изведена на стандартния изход след извикването на метода `Fig`(3) (за C#) или `fig`(3) (за Java)?

C#

```
static void Row (int number)
{
   if (number > 0)
   {
      Console.Write(number + " ");
      Row(number - 1);
   }
}
static void Fig(int number)
{
   if (number > 0)
   {
      Fig(number - 1);
      Row(number);
      Console.WriteLine();
   }
}
```

Java

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

public static void fig (int number){
   if (number > 0){
      fig (number - 1);
      row (number);
      System.out.println();
   }
}
```

А) 1 2 3
   1 2
   1

Б) 3 2 1
   2 1
   1
В) 1
   1 2
   1 2 3
Г) 1
   2 1
   3 2 1

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

C#

```
string str = " C# Programming Course";
string sub = str.Substring(4, 11);
int index1 = str.IndexOf("C#");
int index2 = str.IndexOf("r", 7);
Console.WriteLine(sub + " " + index1 + " " + index2);
```

Java

```
String str = " C# Programming Course";
String sub = str.substring(4, 15);
int index1 = str.indexOf("C#");
int index2 = str.indexOf("r", 7);
System.out.println(sub + " " + index1 + " " + index2);
```

А) Programming 0 6
Б) Programming 1 8
В) Programming 0 7
Г) C# Programming 1 7

9. Кой алгоритъм е представен в програмния фрагмент, ако се знае, че масивът array е подреден във възходящ ред?

C#

```
static int Function (int[] array, int left, int right, int num)
{
    if (left > right) return -1;
    int middle = (left + right) / 2;
    if (num == array[middle]) return middle;
    if (num < array[middle])
        return Function (array, left, middle - 1, num);
    else
        return Function (array, middle + 1, right, num);
 }
```

Java

```
static int function(int[] array, int left, int right, int num) {
```

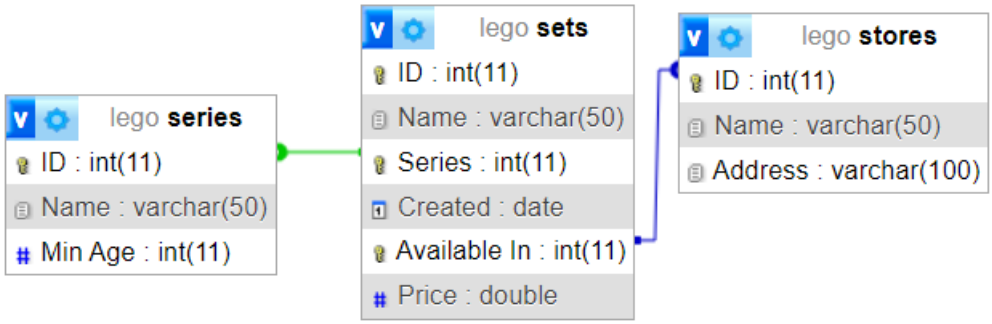
```
    if (left > right) return -1;
    int middle = (left + right) / 2;
    if (num == array[middle]) return middle;
    if (num < array[middle])
       return function(array, left, middle - 1, num);
    else
       return function(array, middle + 1, right, num);
}
```

А) Сливане на сортирани редици
Б) Бързо сортиране в низходящ ред
В) Двоично търсене
Г) Комбинации

10. През кой етап от жизнения цикъл на една информационна система се разработва времеви график и план за изпълнение на проекта

А) Проектиране
Б) Разработка
В) Експлоатация
Г) Анализ

11. Кое е вярното твърдение за дадената диаграма?



А) Връзката между таблиците `lego stores - lego sets` е **1 : n** (едно към много).
Б) Връзката между таблиците `lego stores - lego sets` е **n : n** (много към много).
В) Атрибутът `Series` на таблицата `lego sets` е външен ключ за таблицата `lego series`.
Г) Атрибутът `ID` на таблицата `lego stores` е външен ключ за таблицата `lego sets`.

12. Даден е текстовия файл Numbers.txt, съдържащ произволен брой цели числа, всяко на нов ред. Какъв ще бъде резултатът от изпълнението на следния програмен фрагмент?

C#

```
static void Main(string[] args)
{
```

```
    int[] array = { 1, 2, 15 };
    foreach (int el in array)
    {
       Console.WriteLine(el);
    }
    StreamReader reader = new StreamReader("Numbers");
    string line = reader.ReadLine();
    while (line != null)
    {
        int num = Convert.ToInt32(line);
        Console.WriteLine(num);
        line = reader.ReadLine();
    }
}
```

Java

```
public static void main(String[] args) throws FileNotFoundException {
    int[] array = {1, 2, 15};
    for (int el : array) {
        System.out.println(el);
    }
    Scanner scanner = new Scanner(new File("Numbers"));
    while (scanner.hasNext()) {
        int num = scanner.nextInt();
        System.out.println(num);
    }
}
```

А) Програмата няма да стартира, ще даде грешка – грешно име на файл.

Б) Програмата ще стартира, но ще прекъсне със съобщение за грешка, че файлът не може да бъде намерен.

В) Програмата ще стартира и ще отвори файл Numbers.txt за четене.

Г) Програмата ще стартира, ще създаде файл Numbers и ще го отвори за четене.

13. В база от данни в таблицата `meals` (дадена по-долу) са въведени следните записи:

Id	Name	Category	Quantity	Price
1	chicken	Protein	200	12,50
2	rice	Grain	150	4,50
3	broccoli	Vegetable	200	3,80
4	pasta	Grain	230	7,50
5	potato	Grain	360	5,50
6	chocolate	Dessert	100	4,50
7	pork	Protein	250	15,60
8	fish	Protein	200	10,10
9	peas	Vegetable	150	2,50
10	apple	Dessert	80	1,50
11	tofu	Protein	200	9,70

12	milk	Beverage	200	3,00

Какъв ще бъде резултатът от изпълнението на дадената заявка?

```
SELECT Category, COUNT(Id) AS Count_of_meals FROM meals
GROUP BY Category
HAVING COUNT(Id)>=2
ORDER BY Count_of_meals, Category DESC
```

А)

Category	Count_of_meals
Vegetable	2
Dessert	2
Grain	3
Protein	4

Б)

Category	Count_of_meals
Protein	4
Grain	3
Vegetable	2
Dessert	2

В)

Category	Count_of_meals
Grain	3
Protein	4

Г)

Category	Count_of_meals
Protein	4
Grain	3

14. Посочете правилната последователност на описаните абстрактни структури от данни.

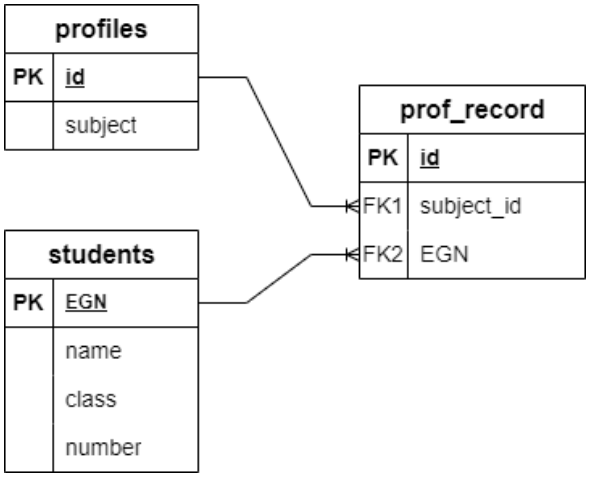
Линейната абстрактна структура от данни …….. е структура от данни, която се използва за съхранение на последователност от еднотипни елементи, като елементите се добавят и премахват на място в структурата, където е необходимо.

Линейната абстрактна структура от данни …….. е структура от данни, която се използва за съхранение на последователност от еднотипни елементи, като елементите се добавят и премахват в съответствие с принципа "първи влязъл, първи излязъл" (First In, First Out).

Линейната абстрактна структура от данни …….. е структура от данни, която се използва за съхранение на последователност от еднотипни елементи, като елементите се добавят и премахват в съответствие с принципа „последен влязъл, първи излязъл“ (Last In, First Out).

А) стек, опашка, списък
Б) опашка, списък, стек
В) списък, опашка, стек
Г) опашка, стек, списък

15. Дадена е следната диаграма на релационна база от данни:



Кои от твърденията са верни за връзките между таблиците в базата данни?

1. Връзката между таблица students и таблица prof_record е едно към много.
2. Връзката между таблица students и таблица prof_record е едно към едно.
3. Връзката между таблица profiles и таблица prof_record е едно към много
4. Връзката между таблица profiles и таблица prof_record е много към много.
5. Няма връзка между таблица students и таблица profiles.
6. Връзката между таблица students и таблица profiles е много към много.

А) 1, 4, 5
Б) 1, 3, 6
В) 2, 3, 5
Г) 2, 4, 6

16. Добавянето на кои два реда, на празното място в дадения програмен фрагмент, ще доведе до извеждане на стандартния изход думата „nation”, като резултат от изпълнението на фрагмента?

C#

```
var str = new StringBuilder();
.........................
.........................
Console.WriteLine(str);
```

Java

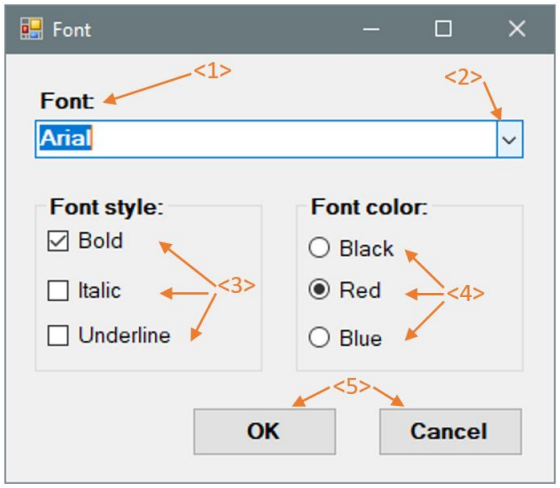
```
var str = new StringBuilder();
.........................
.........................
System.out.println(str);
```

C#	**Java**
А) `str.Add("examination");` `str.Remove(0, 5);` Б) `str.Add("examination");`	А) `str.add("examination");` `str.delete(0, 5);` Б) `str.add("examination");`

`   str.Remove(0, 4);`
`В) str.Append("examination");`
`   str.Remove(0, 5);`
`Г) str.Append("examination");`
`   str.Remove(0, 4);`
	`   str.delete(0, 4);`
	`В) str.append("examination");`
	`   str.delete(0, 5);`
	`Г) str.append("examination");`
	`   str.delete(0, 4);`

Отговорите на задачите от 17. до 24. включително запишете в полетата за отговори под задачата!

17. В полетата за отговори срещу всеки номер напишете визуален компонент, отговарящ на номера върху даденото изображение, който може да бъде използван за изграждането на представения на изображението потребителски интерфейс.



18. Довършете изреченията, като в полетата за отговори срещу А) и Б) запишете типа връзка `is-a` (е), `has-a` (има) или няма връзка.

А) Когато класа А наследява класа В, казваме, че класовете А и В са свързани с връзка от тип ……

Б) Когато поле на класа А е препратка (референция) към клас В, казваме, че класовете А и В са свързани с връзка от тип ……

19. В празна опашка последователно са постъпили елементите 1, 20, 1, 5, 0, 10, а в празен стек – 0 и 6. От опашката се изваждат два елемента, първият от които (ако е различен от 0) се добавя отново в опашката, а вторият от тях се добавя в стека. Действието се повтаря до изваждането на елемент със стойност 0 от опашката (няма значение дали е първи или втори от двойката).

В полетата за отговори срещу (1), (2), (3) и (4) запишете:

(1) Колко са елементите в стека?

(2) Колко е броят на елементите в опашката?

(3) Кои са елементите на стека?

(4) Кои са елементите на опашката?

Забележка: Винаги се изваждат два елемента от опашката, дори ако единият от тях е 0.

20. Даденият програмен фрагмент трябва да извежда на стандартния изход всички прости делители (без повторения) на естествено число, въведено от стандартния вход, но има изпуснати части, отбелязани с номера <1>, <2>, <3>, <4>, <5> и <6>.

Срещу номерата <1>, <2>, <3>, <4>, <5> и <6> запишете пропуснатите части от кода, така че да се получи верен програмен код.

C#

```
int number = ...<1>...;
for (int i = 2; ...<2>...; ...<3>...)
{
    if ( ...<4>...)
        Console.Write(i + " ");
    while ( ...<5>...)
    {
       ...<6>...;
    }
}
```

Java

```
Scanner scr = new Scanner(System.in);
int number = ...<1>...;
for (int i = 2; ...<2>...; ...<3>...) {
   if( ...<4>...)
      System.out.print(i + " ");
   while ( ...<5>...) {
      ...<6>...;
   }
}
```

21. В полето за отговори запишете какво ще се изведе на стандартния изход след изпълнението на програмния фрагмент?

C#

```
public class Test
{
    public static void Main(string[] args)
    {
        try
        {
            Console.WriteLine("Welcome to Java");
            int i = 0;
            int y = 2 / i;
            Console.WriteLine("Welcome to C#");
        }
```

```
        catch (DivideByZeroException ex)
        {
            Console.WriteLine("Welcome to Java and C#");
        }
        finally
        {
            Console.WriteLine("End of the block");
        }
    }
}
```

JAVA

```
public class Test {
   public static void main(String[] args) {
      try {
         System.out.println("Welcome to Java");
         int i = 0;
         int y = 2 / i;
         System.out.println("Welcome to C#");
      }
      catch (RuntimeException ex) {
         System.out.println("Welcome to Java and C#");
      }
      finally {
         System.out.println("End of the block");
      }
   }
}
```

22. Програмният фрагмент трябва да въвежда от стандартния вход размера на квадратен масив от цели числа с неповече от 20 реда и 20 колони, както и неговите елементи. От стандартния вход коректно се въвеждат на първи ред – броят *n* на редовете/колоните, а следващите *n* реда съдържат по *n* цели числа, не по-големи от 1000000 по абсолютна стойност, разделени с интервал. Фрагментът трябва да намира и извежда на стандартния изход сумата от нечетните стойности на елементите на вторичния диагонал на двумерния масив. В кода има допуснати грешки. В полето за отговор запишете на кои редове има грешка и срещу номера на реда запишете верния код.

Забележка: Вторичен диагонал – диагоналът на матрицата, започващ от горен десен елемент и стигащ до долен ляв елемент.

C#

```
1  int n = Convert.ToInt32(Console.ReadLine());
2  int sum = 0;
3  int[,] arr = new int[20, 20];
4  for (int i = 0; i < n; i++) {
5       string[] line = Console.ReadLine().Split();
6       for (int i = 0; i < n; i++) {
7            arr[i, j] = Convert.ToInt32(line[j]);
8       }
9  }
10 for (int i = 0; i < n; i++) {
11      if (arr[i, i] % 2 == 1) {
12          sum += arr[i, i];
13       }
14 }
15 Console.WriteLine(sum);
```

Java

```
1  Scanner scr = new Scanner(System.in);
2  int n = scr.nextInt();
3  int sum = 0;
4  int[][] arr = new int[20][20];
5   for (int i = 0; i< n; i++) {
6       for (int i = 0; i< n; i++) {
7           arr[i][j] = scr.nextInt();
8       }
9  }
10 for (int i = 0; i < n; i++) {
11      if(arr[i][i] % 2 == 1) {
12          sum += arr[i][i];
13      }
14 }
15 System.out.println(sum);
```

23. В полето за отговор запишете по какво трябва да се различават методите на даден клас, за да ги наричаме претоварени (overload) методи?

(1) Брой на аргументите

(2) Имената на методите

(3) Подредбата (позицията) на аргументите, които са от различен тип

(4) Тип на аргументите

(5) Тип на връщания от метода резултат

24. За всеки клас от представените в полетата за отговори, обозначени с <1>, <2> и <3> запишете метода, който добавя елемент след последния към създадената инстанция.

C#

```
List<String> list = new List<String>();
list. ...<1>... ("Maria");

Stack<String> stack = new Stack<String>();
stack. ...<2>... ("Maria");

Queue<String> queue = new Queue<String>();
queue. ...<3>... ("Maria");
```

Java

```
List<String> list = new ArrayList<>();
list. ...<1>... ("Maria");

Stack<String> stack = new Stack<>();
stack. ...<2>... ("Maria");

Queue<String> queue = new ArrayDeque<>();
queue. ...<3>... ("Maria");
```

25. На изображението са дадени таблиците `Students` и `Profiles` от релационна база с данни и резултат (таблицата `Query result`) от дадената по-долу заявка:

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	5
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

Query result

ProfileId	Max_Points
1	200
2	300
3	280
4	160
5	250

```
SELECT ProfileId, MAX(Points) AS Max_Points
FROM Students
GROUP BY ProfileId
```

В полето за отговори напишете модификация на дадената заявка, така че като резултат да се добави още една колона, в която се визуализира името на профила.

МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ ПО ИНФОРМАТИКА ПРОФИЛИРАНА ПОДГОТОВКА

23 август 2024 г.

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!

26. Управител на IT компания получава доклад за печалбата ѝ за n ($n \leq 100$) месеца назад в хиляди лева. Той иска да разбере най-големия брой последователни месеци, през които компанията е имала печалба не по-ниска от тази на предходния месец и с колко процента най-ниската печалба за периода на доклада е по-малка от печалбата за следващия месец.

Създайте приложение с име ***Zad26***, което чете от стандартния вход два реда: от първия ред се чете едно естествено число *n* – броят на месеците в доклада и от втория ред - *n* естествени числа, разделени с точно един интервал – печалбите (в хиляди лева). Приложението намира най-големия брой последователни месеци, през които компанията е имала печалба не по-малка от тази на предходния месец и с колко процента най-ниската печалба за периода на доклада е по-малка от печалбата за следващия месец. Ако най-ниската печалба се достига в последния за доклада месец, то тя се сравнява с предпоследния месец от доклада. Гарантирано е, че има само един месец с най-ниска печалба и входните данни се подават коректно.

Приложението извежда на стандартния изход търсената информация в следния формат:

На първия ред се извежда: The longest period with bigger profit is <брой последователни месеци> mounths.

На втория ред се извежда: Smaller with <процент> %.

Дробните числа да се форматират до втория знак след десетичната запетая.

Примерен вход:

10

5 3 4 6 7 1 2 3 4 5

Изход:

```
Longest period with bigger profit is 5 mounths.
Smaller with 50.00%
```

**Забележки*: 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.

Прикачете в изпитната система архив с име inf_23.08.2024_ zad26.zip, съдържащ файловете с Вашите решения.

27. Създайте база от данни с име **Travel**, която съхранява данни за екскурзии до различни дестинации.

А) Напишете заявка, чрез която в базата данни се създава таблицата **Countries** със следните атрибути:

- **ID** – цяло число, автоматично увеличаващо се с 1, първичен ключ
- **Name** – текст до 30 знака.

Б) Напишете заявка, чрез която в базата данни се създава таблицата **Destinations** със следните атрибути:

- **ID** – цяло число, автоматично увеличаващо се с 1, първичен ключ
- **Town** – текст до 30 знака.
- **Distance** – цяло число;
- **Duration** – цяло число;
- **Price** – реално число до 10 знака с 2 знака след десетичната запетаята;
- **CountryId** – цяло число, външен ключ към таблица Countries.

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

ID	Name
1	France

2	Germany
3	Italy
4	Spain
5	Austria

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

ID	Town	Distance	Duration	Price	CountryId
1	Paris	2169	4	1800.00	1
2	Berlin	4006	6	2100.00	2
3	Rome	1666	3	1500.00	3
4	Madrid	2966	7	1800.00	4
5	Milan	1408	4	1900.00	3
6	Venice	1152	3	1200.00	3
7	Barcelona	2375	7	2800.00	4

Д) Напишете заявка, която променя цената на пътуването до Paris на 1700.

Е) Напишете заявка, която изтрива от таблицата Destinations всички записи, чийто имена на градове започват с ‘Ва‘

Ж) Напишете заявка, която връща името на града и разстоянието, за тези дестинации, които са на разстояние между 1500 и 2500 (включително).

З) Напишете заявка, която връща името на града и цената на най-скъпата дестинация.

И) Напишете заявка, която намира за всяка страна броя на дестинациите в нея, подредени в намаляващ ред по брой пътувания. При равен брой на пътувания, данните да се подредят по име на държавата в нарастващ ред.

Прикачете в изпитната система файл с име inf_23.08.2024_ 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 търсени заявки.

28. Във футболен клуб работят директори, треньори и футболисти.

Създайте приложение с име **Zad28**, което имплементира базов клас ClubMember и негови наследници FootballPlayer (футболист), Director (директор) и Coach (треньор). Класовете трябва да съдържат следните характеристики и свойства:

А) Класът **ClubMember** с полета:

- firstName (име) - низ
- lastName (фамилия) - низ
- age (възраст)– цяло число
- salary (заплата) – реално число

Класът трябва да притежава следните методи:

- **конструктор**, който задава стойности на всички характеристики;
- **абстрактен метод** Info (за C#) и info (за Java), който извежда на стандартния изход информацията за съответния служител;

Данните за класа ClubMember да се валидират, съгласно следните ограничения:

- Имента на членовете на клуба не могат да са правзни низове иначе да се генерира изключение със съобщението: „The name can’t be an empty string!“
- Възрастта трябва да число по-голямо от 17, иначе да се генерира изключение със съобщението: „Age must be greater than 17 years!“
- Заплатата трябва да е положително число, иначе да се генерира изключение със съобщението: „Salary must be a positive number!“

Б) Класът **Director** наследява ClubMember и съдържа собствени полета:

- directorType (вид директор) – низ – една от думите executive, technical или sports

Класът трябва да притежава следните методи:

- **конструктор**, който задава всички стойности на характеристиките;
- **предефиниран метода** Info (за C#) и info (за Java), така че да се извежда на стандартния изход информацията в следния формат:

  На първия ред: <вид директор> director: <име> <фамилия>

  На втория ред: salary: <заплата> lv

  На третия ред: age: <възраст> years

Заплатата да бъде форматирана до втория знак след десетичния разделител.

В) Класът **Coach** наследява ClubMember и съдържасобствени характеристики:

- coachType (вид треньор) – низ – една от думите: head, assistant или goalkeeper
- contractLength (продължителност на договора) – цяло число

Класът трябва да притежава следните методи:

- **конструктор**, който задава всички стойности на характеристиките;
- **предефиниран метода** Info (за C#) и info (за Java), така че да се извежда на стандартния изход информацията в следния формат:

  На първия ред: <вид треньор> coach: <име> <фамилия>

  На втория ред: salary: <заплата> lv

  На третия ред: age: <възраст> years

Заплатата да бъде форматирана до втория знак след десетичния разделител.

Г) Класът **FootballPlayer** наследява ClubMember и съдържа собствени характеристики:

- position (позиция)– низ – една от думите: forward, midfielder, defender, goalkeeper
- contractLength (продължителност на договора) – цяло число
- matches (брой мачове) – цяло число
- goals (брой голове) – цяло число
- assist (брой асистенции) – цяло число

Класът трябва да притежава следните методи:

- **конструктор**, който задава всички стойности на характеристиките;
- **предефиниран метода** Info (за C#) и info (заJava), така че да се изведа на стандартния изход информацията в следния формат:

  На първия ред: <име> <фамилия> - <позиция>

  На втория ред: salary: <заплата> lv

  На третия ред: age: <възраст> years

  На четвъртия ред: <брой голове> goals and <брой асистенции> assists in <брой мачове> matches

Д) Приложението трябва да прочете от текстов файл **input.txt** данните за членовете на футболен клуб, като информацията за всеки член на клуба е записана на един ред. Приложението трябва да извежда на стандартният изход информацията за всеки един от членовете на клуба, сортирани по години в нарастващ ред, като информацията за всеки член на клуба се отделя с един ред, съдържащ двадесет звездички. Накрая на отделни редове да се изведат следните две съобщения:

The person with the highest salary in the club is <име> <фамилия> with <заплата> lv salary.

The team's top scorer is <име> <фамилия> with <брой голове> goals.

Приложението трябва да прихваща и обработва изключенията.

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

Изход:

```
Petar Petrov - forward
salary: 9000.00 lv
age: 18 years
3 goals and 0 assists in 10 matches
********************
Ivan Ivanov - forward
salary: 10000.00 lv
age: 22 years
25 goals and 10 assists in 50 matches
********************
Kaloyan Kaloyanov - defender
salary: 6000.00 lv
age: 24 years
5 goals and 2 assists in 80 matches
********************
Georgi Georgiev - midfielder
salary: 11000.00 lv
age: 27 years
19 goals and 33 assists in 100 matches
********************
Dimitar Dimitrov - goalkeeper
salary: 7500.00 lv
age: 35 years
0 goals and 0 assists in 300 matches
********************
```

```
assistant coach: Plamen Kirov
salary: 3000.00 lv
age: 41 years
**********
sports director: Biser Biserov
salary: 18000.00 lv
age: 48 years
**********
head coach: Gari Plamenov
salary: 4000.00 lv
age: 51 years
**********
executive director: Daniel Danielov
salary: 25000.00 lv
age: 55 years
**********
technical director: Grigor Grigorov
salary: 20000.00 lv
age: 61 years
**********
The person with the hightes salary in the club is Daniel Danielov with 25000.00 lv salary.
The team's top scorer is Ivan Ivanov with 25 goals.
```

Прикачете в изпитната система архив с име inf_23.08.2024_ zad28.zip, съдържащ файловете с Вашите решения.

МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ ПО

ИНФОРМАТИКА

23 август 2024 г.

ПРОФИЛИРАНА ПОДГОТОВКА

ВАРИАНТ 2

Задача от 1. до 16. Ключ с верните отговори

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

Задача 17. – 2 точки

<1> етикет / Label

<2> комбинирана кутия /Combo Box

<3> поле за отметка / Check Box

<4> радио бутон / Radio Button

<5> бутон / Button

Задача 18. – 2 точки

А) is-a / е

Б) has-a / има

Задача 19. – 2 точки

(1) 4

(2) 2

(3) 5, 20, 6, 0 или 0, 6, 20, 5

(4) 1, 1

Задача 20. – 3 точки

<1> **За C#:** `Convert.ToInt32(Console.ReadLine())`

За Java: `scr.nextInt()`

<2> `i <= number`

<3> `i++ или ++i или i+=1 или i= i+1`

<4> `number % i == 0`

<5> `number % i == 0`

<6> `number /= i или number = number / i`

Задача 21. – 3 точки

```
Welcome to Java
Welcome to Java and C#
End of the block
```

Задача 22. – 3 точки

C#	Java
Ред 6 for (int j = 0; j < n; j++) Ред 11 if(arr[i][n-1-i] % 2 != 0) Ред 12 sum += arr[i][n-1-i];	Ред 6 for (int j = 0; j< n; j++) Ред 11 if(arr[i][n-1-i] % 2 != 0) Ред 12 sum += arr[i][n-1-i];

Задача 23. – 3 точки

(1) Брой на аргументите
(3) Подредбата (позицията) на аргументите, които са от различен тип
(4) Тип на аргументите
Признава се и ако са записани само номерата или само думите, като не се взема под внимание реда на изброяването.

Задача 24. – 3 точки

ред	C#	Java
< 1 >	Add	add
< 2 >	Push	push / add
< 3 >	Enqueue	offer / add

Задача 25. – 3 точки

```
SELECT s.ProfileId, p.Name,  MAX(s.Points) AS [Max Points]
FROM Students  as s
JOIN Profiles as p ON s.ProfileId = p.Id
GROUP BY s.ProfileId, p.Name
```

Задача 26 – 15 точки

Примерно решение

C#

```
using System;

namespace DZI
{
    class Zad26
    {
        static void Main(string[] args)
        {
            int n = Convert.ToInt32(Console.ReadLine());
            int[] profit = new int[n];

            string[] input = Console.ReadLine().Split(' '); ;
            for (int i = 0; i < n; i++)
            {
                profit[i] = Convert.ToInt32(input[i]);
            }

            int longestStreak = 1;
            int currentStreak = 1;
            for (int i = 1; i < n; ++i)
            {
                if (profit[i] <= profit[i - 1])
                {
                    if (currentStreak > longestStreak)
                    {
                        longestStreak = currentStreak;
                    }
                    currentStreak = 0;
                }
                ++currentStreak;
            }
```

```
            if (currentStreak > longestStreak)
            {
                longestStreak = currentStreak;
            }
            Console.WriteLine("The longest period with bigger profit is "
+ longestStreak + " mounths.");

            int smallestProfit = profit[0];
            int targetPeriod = 0;
            for (int i = 1; i < n; i++)
            {
                if (profit[i] < smallestProfit)
                {
                    smallestProfit = profit[i];
                    targetPeriod = i;
                }
            }
            int closestProfit;
            if (targetPeriod == n - 1)
            {
                closestProfit = profit[n - 2];
            }
            else
            {
                closestProfit = profit[targetPeriod + 1];
            }
            string str = "%.";
            double smallerPercentage = (closestProfit - smallestProfit) /
(double)closestProfit * 100;
            Console.WriteLine($"Smaller with
{smallerPercentage:F2}{str}");
        }
    }
}
```

Java

```
package DZI;

import java.util.Scanner;

public class Zad26 {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] profit = new int[n];

        for (int i = 0; i < n; i++) {
```

```
            profit[i] = scanner.nextInt();
        }

        int longestStreak = 1;
        int currentStreak = 1;
        for (int i = 1; i < n; ++i) {
            if (profit[i] <= profit[i - 1]) {
                if (currentStreak > longestStreak) {
                    longestStreak = currentStreak;
                }
                currentStreak = 0;
            }
            ++currentStreak;
        }
        if (currentStreak > longestStreak) {
            longestStreak = currentStreak;
        }
        System.out.println("The longest period with bigger profit is " +
longestStreak + " mounths.");

        int smallestProfit = profit[0];
        int targetPeriod = 0;
        for (int i = 1; i < n; i++) {
            if (profit[i] < smallestProfit) {
                smallestProfit = profit[i];
                targetPeriod = i;
            }
        }
        int closestProfit;
        if (targetPeriod == n - 1) {
            closestProfit = profit[n - 2];
        } else {
            closestProfit = profit[targetPeriod + 1];
        }
        String str ="%.";
        double smallerPercentage = (closestProfit - smallestProfit) /
(double) closestProfit * 100;
        System.out.println(String.format("Smaller with %.2f%s",
smallerPercentage, str));
    }
}
```

Задача 27 – 20 точки

Примерно решение

```
CREATE DATABASE Travel;
USE Travel;
```

-- А

MySql	SQL
CREATE TABLE Countries (ID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(30));	CREATE TABLE Countries (ID INT Identity PRIMARY KEY, "Name" NVARCHAR(30))

-- Б

MySql	SQL
CREATE TABLE Destinations (ID INT PRIMARY KEY AUTO_INCREMENT, Town VARCHAR(30), Distance INT, Duration INT, Price DECIMAL(10, 2), CountryId INT, FOREIGN KEY (CountryId) REFERENCES Countries (ID));	CREATE TABLE Destinations (ID INT Identity PRIMARY KEY, Town NVARCHAR(30), Distance INT, Duration INT, Price DECIMAL(10, 2), CountryId INT FOREIGN KEY REFERENCES Countries(ID))

```
-- В
INSERT INTO Countries (Name)
VALUES ('France'), ('Germany'), ('Italy'), ('Spain'), ('Austria');

-- Г
INSERT INTO Destinations (Town, Distance, Duration, Price, CountryId)
VALUES
  ('Paris', 2169 , 4, 1800.00, 1),
  ('Berlin', 4006, 6, 2100.00, 2),
  ('Rome', 1666, 3, 1500.00, 3),
  ('Madrid', 2966, 7, 1800.00, 4),
  ('Milan', 1408, 4, 1900.00, 3),
  ('Venice', 1152 , 3, 1200.00, 3),
  ('Barcelona', 2375, 7, 2800.00, 4);

-- Д
UPDATE Destinations
SET Price = 1700
WHERE Town = 'Paris';
```

```
-- Е
DELETE FROM Destinations
WHERE Town LIKE 'Ba%';

-- Ж
SELECT Town, Distance
FROM Destinations
WHERE Distance BETWEEN 1500 AND 2500;

-- З
```

MySql	SQL
SELECT Town, Price FROM Destinations ORDER BY Price DESC LIMIT 1;	SELECT TOP 1 Town, Price FROM Destinations ORDER BY Price DESC

```
-- И
SELECT c.Name, COUNT(d.CountryId)
FROM Countries AS c
JOIN Destinations AS d ON d.CountryId = c.ID
GROUP BY c.Name
ORDER BY COUNT(d.CountryId) DESC, c.Name;
```

Задача 28 – 25 точки

Примерно решение

C#

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

namespace Zad28
{
    class Zad28
    {
        public static void Main(string[] args)
        {
            var clubMembers = new List<ClubMember>();
            using (StreamReader reader = new StreamReader("input.txt"))
            {
                string input;
                while ((input = reader.ReadLine()) != null)
                {
                    string[] line = input.Split(' ');
                    int len = line.Length;
                    switch (len)
                    {
                        case 9:
                            try
```

```
                            {
                                FootballPlayer newPlayer = new
FootballPlayer(line[0], line[1],
                                        int.Parse(line[2]),
double.Parse(line[3]),
                                        line[4], int.Parse(line[5]),
int.Parse(line[6]),
                                        int.Parse(line[7]),
int.Parse(line[8]));
                                clubMembers.Add(newPlayer);
                            }
                            catch (Exception e)
                            {
                                Console.WriteLine(e.Message);
                            }
                            break;
                        case 6:
                            try
                            {
                                Coach coach = new Coach(line[0], line[1],
int.Parse(line[2]),
                                        double.Parse(line[3]), line[4],
int.Parse(line[5]));
                                clubMembers.Add(coach);
                            }
                            catch (Exception e)
                            {
                                Console.WriteLine(e.Message);
                            }
                            break;
                        case 5:
                            try
                            {
                                Director dirctor = new Director(line[0],
line[1],
                                        int.Parse(line[2]),
double.Parse(line[3]), line[4]);
                                clubMembers.Add(dirctor);
                            }
                            catch (Exception e)
                            {
                                Console.WriteLine(e.Message);
                            }
                            break;
                    }
                }
            }
            clubMembers.Sort((x, y) => x.Age.CompareTo(y.Age));
            foreach (ClubMember member in clubMembers)
            {
                member.Info();
                Console.WriteLine("********************");
            }
            double maxSalary = clubMembers.Max(cm => cm.Salary);
```

```
            ClubMember memberMaxSalery = clubMembers.FirstOrDefault(cm =>
cm.Salary == maxSalary); ;

            Console.WriteLine($"The person with the highest salary in the club
is {memberMaxSalery.FirstName} " +
                $"{memberMaxSalery.LastName} with {memberMaxSalery.Salary:f2}
lv salary.");

            FootballPlayer goalMaster = clubMembers
                .Where(x => x.GetType() == typeof(FootballPlayer))
                .Select(x => x as FootballPlayer)
                .OrderByDescending(x => x.Goals)
                .FirstOrDefault();

            Console.WriteLine($"The team's top scorer is {goalMaster.FirstName}
" +
                $"{goalMaster.LastName} with {goalMaster.Goals} goals.");
        }
    }

abstract class ClubMember
    {
        string firstName, lastName;
        int age;
        double salary;

        public string FirstName
        {
            get
            {
                return this.firstName;
            }
            set
            {
                if (value == "") throw new Exception("The name can’t be an
empty string!");
                else this.firstName = value;
            }
        }

        public string LastName
        {
            get
            {
                return this.lastName;
            }
            set
            {
                if (value == "") throw new Exception("The name can’t be an
empty string!");
                else this.lastName = value;
            }
        }
        public int Age
```

```
        {
            get
            {
                return this.age;
            }
            set
            {
                if (value <= 17) throw new Exception("Age must be greater than
17 years!");
                else this.age = value;
            }
        }

        public double Salary
        {
            get
            {
                return this.salary;
            }
            set
            {
                if (value <= 0) throw new Exception("Salary must be a positive
number!");
                else this.salary = value;
            }
        }

        public ClubMember(string firstName, string lastName, int age, double
salary)
        {
            FirstName = firstName;
            LastName = lastName;
            Age = age;
            Salary = salary;
        }

        public abstract void Info();
    }

class Director : ClubMember
    {
        public Director(string firstName, string lastName, int age, double
salary, string directorType) : base(firstName, lastName, age, salary)
        {
            DirectorType = directorType;
        }

        public string DirectorType { get; set; }

    public override void Info()
        {
            Console.WriteLine($"{this.DirectorType} director: {this.FirstName}
{this.LastName}");
            Console.WriteLine($"salary: {this.Salary:F2} lv");
```

```
            Console.WriteLine($"age: {this.Age} years");
        }
    }

class Coach : ClubMember
    {
        public string CoachType { get; set; }

        public int ContractLength { get; set; }

        public Coach(string firstName, string lastName, int age, double salary,
string coachType, int contractLength) : base(firstName, lastName, age, salary)
        {
            CoachType = coachType;
            ContractLength = contractLength;
        }

    public override void Info()
        {
            Console.WriteLine($"{this.CoachType} coach: {this.FirstName}
{this.LastName}");
            Console.WriteLine($"salary: {this.Salary:f2} lv");
            Console.WriteLine($"age: {this.Age} years");
        }
    }
class FootballPlayer : ClubMember
    {
        public string Position { get; set; }

        public int ContractLength { get; set; }

        public int Matches { get; set; }

        public int Goals { get; set; }

        public int Assist { get; set; }

        public FootballPlayer(string firstName, string lastName, int age,
double salary, string position, int contractLength, int matches, int goals, int
assist) : base(firstName, lastName, age, salary)
        {
            Position = position;
            ContractLength = contractLength;
            Matches = matches;
            Goals = goals;
            Assist = assist;
        }

    public override void Info()
        {
            Console.WriteLine($"{FirstName} {LastName} - {Position}");
            Console.WriteLine($"salary: {Salary:f2} lv");
            Console.WriteLine($"age: {Age} years");
```

```
            Console.WriteLine($"{Goals} goals and {Assist} assists in {Matches}
matches");
        }
    }
}
```

Java

```
package DZI;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;

public class Zad28 {

    public static void main(String[] args) throws FileNotFoundException {
        List<ClubMember> clubMembers = new ArrayList<>();
        try ( Scanner reader = new Scanner(new FileInputStream("input.txt"))) {
            while (reader.hasNext()) {
                String[] line = reader.nextLine().split(" ");
                int len = line.length;
                switch (len) {
                    case 9:
                        try {
                        FootballPlayer player = new FootballPlayer(line[0],
line[1],
                                Integer.parseInt(line[2]),
Double.parseDouble(line[3]),
                                line[4], Integer.parseInt(line[5]),
Integer.parseInt(line[6]),
                                Integer.parseInt(line[7]),
Integer.parseInt(line[8]));
                        clubMembers.add(player);
                    } catch (Exception e) {
                        System.out.println(e.getMessage());
                    }
                    break;
                    case 6:
                        try {
                        Coach coach = new Coach(line[0], line[1],
Integer.parseInt(line[2]),
                                Double.parseDouble(line[3]), line[4],
Integer.parseInt(line[5]));
                        clubMembers.add(coach);
                    } catch (Exception e) {
                        System.out.println(e.getMessage());
                    }
                    break;
                    case 5:
                        try {
                        Director dirctor = new Director(line[0], line[1],
                                Integer.parseInt(line[2]),
Double.parseDouble(line[3]), line[4]);
                        clubMembers.add(dirctor);
                    } catch (Exception e) {
                        System.out.println(e.getMessage());
```

```
                    }
                    break;
                }
            }
        }
        Collections.sort(clubMembers, (p1, p2) -> Integer.compare(p1.age,
p2.age));
        for (ClubMember member : clubMembers) {
            member.info();
            System.out.println("********************");
        }
        ClubMember maxSalery = clubMembers.get(0);
        for (ClubMember member : clubMembers) {
            if(maxSalery.getSalary() < member.getSalary()) maxSalery = member;
        }
        System.out.println(String.format("The person with the highest salary in
the club is "
                + "%s %s with %.2f lv salary.", maxSalery.getFirstName(),
maxSalery.getLastName(),
                maxSalery.getSalary()));

        FootballPlayer player, goalMaster = null;
        for (ClubMember member : clubMembers) {
            if(member instanceof FootballPlayer){
                player = (FootballPlayer) member;
                if(goalMaster == null || player.getGoals() >
goalMaster.getGoals())
                    goalMaster = player;
            }
        }
        System.out.println(String.format("The team's top scorer is %s %s with
%d goals.",
                goalMaster.getFirstName(), goalMaster.getLastName(),
goalMaster.getGoals()));

    }
}

abstract class ClubMember {
    String firstName, lastName;
    int age;
    double salary;

    public void setFirstName(String firstName) throws Exception{
        if(firstName.equals("")) throw new Exception("The name can’t be an
empty string!");
        else this.firstName = firstName;
    }

    public void setLastName(String lastName) throws Exception{
        if(lastName.equals("")) throw new Exception("The name can’t be an empty
string!");
        else this.lastName = lastName;
    }
```

```
    public void setAge(int age) throws Exception{
        if(age <= 17) throw new Exception("Age must be greater than 17
years!");
        else this.age = age;
    }

    public void setSalary(double salary) throws Exception{
        if(salary <= 0) throw new Exception("Salary must be a positive
number!");
        else this.salary = salary;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public int getAge() {
        return age;
    }

    public double getSalary() {
        return salary;
    }

    public ClubMember(String firstName, String lastName, int age, double
salary) throws Exception{
        setFirstName(firstName);
        setLastName(lastName);
        setAge(age);
        setSalary(salary);
    }

    public abstract void info();
}
public class Director extends ClubMember{
    String directorType;

    public Director(String firstName, String lastName, int age, double salary,
String directorType) throws Exception {
        super(firstName, lastName, age, salary);
        this.directorType = directorType;
    }

    public String getDirectorType() {
        return directorType;
    }

    public void setDirectorType(String directorType) {
        this.directorType = directorType;
```

```
    }

    @Override
    public void info() {
        System.out.println(String.format("%s director: %s %s", directorType,
firstName, lastName));
        System.out.println(String.format("salary: %.2f lv", salary));
        System.out.println(String.format("age: %d years", age));
    }
}

public class Coach extends ClubMember{
    String coachType;
    int contractLength;

    public String getCoachType() {
        return coachType;
    }

    public void setCoachType(String coachType) {
        this.coachType = coachType;
    }

    public int getContractLength() {
        return contractLength;
    }

    public void setContractLength(int contractLength) {
        this.contractLength = contractLength;
    }

    public Coach(String firstName, String lastName, int age, double salary,
String coachType, int contractLength) throws Exception {
        super(firstName, lastName, age, salary);
        this.coachType = coachType;
        this.contractLength = contractLength;
    }

    @Override
    public void info() {
        System.out.println(String.format("%s coach: %s %s", coachType ,
firstName, lastName));
        System.out.println(String.format("salary: %.2f lv", salary));
        System.out.println(String.format("age: %d years", age));
    }
}

public class FootballPlayer extends ClubMember {

    String position;
    int contractLength, matches, goals, assist;

    public String getPosition() {
        return position;
```

```
    }

    public void setPosition(String position) {
        this.position = position;
    }

    public int getContractLength() {
        return contractLength;
    }

    public void setContractLength(int contractLength) {
        this.contractLength = contractLength;
    }

    public int getMatches() {
        return matches;
    }

    public void setMatches(int matches) {
        this.matches = matches;
    }

    public int getGoals() {
        return goals;
    }

    public void setGoals(int goals) {
        this.goals = goals;
    }

    public int getAssist() {
        return assist;
    }

    public void setAssist(int assist) {
        this.assist = assist;
    }

    public FootballPlayer(String firstName, String lastName, int age, double
salary, String position, int contractLength, int matches, int goals, int
assist) throws Exception {
        super(firstName, lastName, age, salary);
        this.position = position;
        this.contractLength = contractLength;
        this.matches = matches;
        this.goals = goals;
        this.assist = assist;
    }

    @Override
    public void info() {
        System.out.println(String.format("%s %s - %s", firstName, lastName,
position));
        System.out.println(String.format("salary: %.2f lv", salary));
```

```
        System.out.println(String.format("age: %d years", age));
        System.out.println(String.format("%d goals and %d assists in %d
matches", goals , assist, matches));
    }
}
```