1. **Conceptul de moștenire**

Conceptul de moștenire presupune transmiterea de la obiectul părinte la obiectul copil a proprietăților si metodelor pe care acesta le deține. Astfel o clasa care este derivata din alta clasa o sa conțină de asemeni proprietățile si metodele clasei din care a fost derivata.

Avantaje

Promovează reutilizarea codului

Facilitează crearea bibliotecilor

Face ca o subclasele sa urmeze o interfața standard

Dezavantaje:

Poate duce la utilizarea neeficienta a resurselor de calcul

Metodele claselor derivate vor rula relativ mai încet in timp

O implementare greșita a moștenirii poate duce la o soluție ineficienta

1. **Diferența dintre o clasa obișnuita si o clasa abstracta**

Clasele abstracte conțin metode care pot fi suprascrise de către clase care o deriveaza.

1. **Problema consecutiv**

Console.WriteLine("Dati numarul de elemente: ");

int nr = int.Parse(Console.ReadLine());

int[] arr = new int[nr];

Random rd = new Random();

for (int i = 0; i < nr; i++)

{

//for random elements

arr[i] = rd.Next(1, 100);

//for consecutive elements

//arr[i] = i + 1;

}

DisplayArr(arr);

if (isConsecutive(arr))

Console.WriteLine("Array-ul este consecuriv");

else

Console.WriteLine("Array-ul nu este consecuriv");

bool isConsecutive(int [] arr)

{

return isAscending(arr) | isDescending(arr);

}

bool isDescending(int[] arr)

{

bool good = true;

for (int i = 0; i < arr.GetLength(0)-1; i++)

{

if (arr[i] > arr[i+1])

{

good = false;

break;

}

}

return good;

}

bool isAscending(int[] arr)

{

bool good = true;

for (int i = 0; i < arr.GetLength(0)-1; i++)

{

if (arr[i] < arr[i + 1])

{

good = false;

break;

}

}

return good;

}

void DisplayArr(int[]arr)

{

Console.WriteLine("Array-ul generat: ");

for (int i = 0; i < arr.GetLength(0); i++)

{

Console.Write($"{arr[i]} ");

}

Console.WriteLine();

}

1. **Problema duplicate**

Console.WriteLine("Dati numarul de elemente: ");

int nr = int.Parse(Console.ReadLine());

//int[] arr = new int[nr];

int[] arr = { 3, 8, 7, 3, 3, 5, 3, 2, 1, 3, 3 }; //test case

//Random rd = new Random();

//for (int i = 0; i < nr; i++)

//{

// //for random elements

// arr[i] = rd.Next(1, 100);

// //for consecutive elements

// //arr[i] = i + 1;

//}

var found = from x in arr where arr.Count(c => c == x) > (arr.GetLength(0) / 2) select x;

if (found.Count()==0)

{

Console.WriteLine("Nu au fost gasite elementele");

return;

}

Console.WriteLine("Elementele duplicate care indepolinesc conditia problemei:");

foreach (var item in found.Distinct())

{

Console.WriteLine($"{item} ");

}

1. **Problema steluțe**

Console.WriteLine("Dati numarul de linii:");

int n = int.Parse(Console.ReadLine());

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

{

for (int j = 1; j <=n-i ; j++)

Console.Write(" ");

for (int j = 1; j <=2\*i-1; j++)

Console.Write("\*");

Console.WriteLine();

}