1. Create class Field with the following fields (size, matrixForField) encapsulated with properties and two methods PrintField and FillTheField, responsible for generating the number of mines and filling the field
2. Refactoring of the method PrintField is including:
   1. Removing empty lines
   2. Change variable names to more appropriate ones
   3. Adding comments and empty lines between logically different pieces

|  |  |
| --- | --- |
| public static void PrintField(int[,] arr, int n)  {  Console.Write(" ");  for (int i = 0; i < n; i++)  {  Console.Write(" {0}", i);  }  Console.WriteLine();  Console.Write(" ");  for (int i = 0; i < n \* 2; i++)  {        Console.Write("-");  }  Console.WriteLine();  for (int i = 0; i < n; i++)  { Console.Write("{0}|", i);  for (int j = 0; j < n; j++)  {  char c;  switch (arr[i, j])  {  case 0: c = '-'; break;  case -1: c = 'X'; break;  default: c = (char)('0'+ arr[i, j]);  break;  }  Console.Write("{0} ", c);  }  Console.WriteLine();  }  } | public void Print()  {  int size = this.Size;  int[,] arr = this.MatrixForField;  //Print the numeration of cols  Console.Write(" ");  for (int col = 0; col < size; col++)  {  Console.Write(" {0}", col);  }  Console.WriteLine();  Console.Write(" ");  for (int col = 0; col < size \* 2; col++)  {  Console.Write("-");  }  Console.WriteLine();  for (int row = 0; row < size; row++)  {  Console.Write("{0}|", row);  for (int col = 0; col < size; col++)  {  char cellValue;  switch (arr[row, col])  {  case 0: cellValue = '-'; break;  //when it is already exploded  case -1: cellValue = 'X'; break;  default: cellValue = (char)('0' + arr[row, col]);  break;  }  Console.Write("{0} ", cellValue);  }  Console.WriteLine();  }  } |

1. Refactoring of the code from line 160 to 173 is including:
   1. Extracting it and put it in a method - FillTheField in the class Field
   2. Renaming some of the variable like Proizwolni4isla ->randomGen
   3. Extracting the complex expression from the line with mine number and put them in variables minPercentOfMines and maxPercentOfMines
   4. Adding empty lines between logically different pieces
   5. Removing unnecessary comments

|  |  |
| --- | --- |
| Random ProizvolniChisla = new Random(); //vhoid i inicializaciq na n i matricata;  int mineNumber = ProizvolniChisla.Next(  15 \* n \* n / 100, 30 \* n \* n / 100 + 1  );  for (int i = 0; i < mineNumber; i++) // randomizirane na minite i postavqneto im iz poleto  {  int x = ProizvolniChisla.Next(0, n);  int y = ProizvolniChisla.Next(0, n);  while (arr[x, y] != 0)  {  x = ProizvolniChisla.Next(0, n);  y = ProizvolniChisla.Next(0, n);  }  arr[x, y] = ProizvolniChisla.Next(1, 6);  } | public void FillTheField()  {  Random randomGen = new Random();  int size = this.Size;  int[,] arr = this.MatrixForField;  int minPercentOfMines = 15 \* size \* size / 100;  int maxPercentOfMines = 30 \* size \* size / 100;  int numberOfMines =  randomGen.Next(minPercentOfMines, maxPercentOfMines + 1);  for (int i = 0; i < numberOfMines; i++)  {  int row = randomGen.Next(0, size);  int col = randomGen.Next(0, size);  while (arr[row, col] != 0)  {  row = randomGen.Next(0, size);  col = randomGen.Next(0, size);  }  arr[row, col] = randomGen.Next(1, 6);  }  } |