Documentation of Project "Battle-Field-4" Team "Fermium"

* Create class **Field** with the following fields (**size**, **matrixForField, randomGen- randomGenerator(the aim is to make easier testing of the method fillTheField with Reflection)** ) encapsulated with properties and two methods **PrintField** and **FillTheField**, responsible for generating the number of mines and filling the field
* Refactoring of the method **PrintField** is including:
* Removing empty lines
* Change variable names to more appropriate ones
* Adding comments and empty lines between logically different pieces
* Refactoring of the code from line 160 to 173 is including:
* Extracting it and put it in a method - **FillTheField** in the class Field
* Renaming some of the variable like **Proizwolni4isla** ->**randomGen**
* Extracting the complex expression from the line with mine number and put them in variables **minPercentOfMines** and **maxPercentOfMines**
* Adding empty lines between logically different pieces
* Removing unnecessary comments
* Create static class **MineType** with five static fields generating types of explosion
* Refactoring of the static class is including:
* Removing whitespaces
* Change variable names to more appropriate ones
* Create class **Mine** with the following field **type** capsulated with property and method **ExplodeType**, returning type of explosion
* Refactoring of the method **ExplodeType** is including:
* Extracted it from method **GrymOtQsnoNebe**
* Change variable names to more appropriate ones
* Add another case in the **switch**
* Removing unnecessary comments
* Use static fields from class **MineType**

|  |  |
| --- | --- |
|  |  |

* Create class **GameEngine** with the following field field encapsulated with property and methods **ExplodeMine, SetNextMinePosition, GenerateNumberOfMines, CreateGameField, Play, GetInitialSize** and **InitiateGame**, responsible for generating the game play
* Refactoring of the method **SetNextMinePosition** is including:
* Rename method **TimeToPlay** to **SetNextMinePosition**
* Change variable names to more appropriate ones
* Correct the logic for validation of input data using **Split** method
* Add two out parameters
* Put **{** and **}** after all conditionals and loops (when missing).

|  |  |
| --- | --- |
| public static int TimeToPlay(int[,] arr, int n)  {  int x = 0, y = 0;  bool cond = true;  while (cond) //check input  {  Console.Write("Please enter coordinates: ");  string s = Console.ReadLine();  if (s.Length > 2)  {  x = s.ElementAt(0) - '0';  y = s.ElementAt(2) - '0';  if (x < 0 || x > 9 || y < 0 || y > 9 || s.ElementAt(1) != ' ') Console.WriteLine("Invalid move!");  else  {  if (s.Length > 3)  {  if (s.ElementAt(3) != ') Console.WriteLine("Invalid move!");  else cond = false;  }  else cond = false;  }  }  else Console.WriteLine("Invalid move!");  if (cond == false)  if (arr[x, y] <= 0)  {  cond = true; Console.WriteLine("Invalid move!");  }  }  return GrymOtQsnoNebe(arr, n, x, y);  } | private void SetNextMinePosition(int [,] field, out int row, out int col)  {  row = 0;  col = 0;  bool isInvalid = true;  while (isInvalid)  {  Console.Write("Please enter coordinates: ");  string minePosition = Console.ReadLine();  string[] coordinates = minePosition.Split(new char[] { ' ' },StringSplitOptions.RemoveEmptyEntries);  if (coordinates.Length == 2)  {  row = int.Parse(coordinates[0]);  col = int.Parse(coordinates[1]);  if (row < 0 || row >= field.GetLength(0) || col < 0 || col >= field.GetLength(1))  {  Console.WriteLine("Invalid move!");  }  else  {  isInvalid = false;  }  }  else  {  Console.WriteLine("Invalid move!");  }  if (!isInvalid)  {  if (field[row, col] <= 0)  {  isInvalid = true;  Console.WriteLine("Invalid move!");  }  }  }  } |
|  |  |

* Refactoring of the method **ExplodeMines** including:
* Rename method **GrymOtQsnoNebe** to **ExplodeMine**
* Change variable names to more appropriate ones like **counter** -> **explodeMinesCount**
* Extract method **ExplodeType** and move in class **Mine**
* Call method **SetNextMinePosition**
* Put **{** and **}** after all conditionals and loops (when missing).
* Removing unnecessary comments

|  |  |
| --- | --- |
| public static int GrymOtQsnoNebe(int[,] arr, int n, int x, int y)  {  int[,] expl;  switch (arr[x, y]) // zadava ni koi vid bomba imame  {  case 1: expl = минаЕдно; break;  case 2: expl = минаДве; break;  case 3: expl = минаТри; break;  case 4: expl = минаЧетири; break;  default: expl = минаПет; break;  }  //gyrmi bombata  int counter = 0;  for (int i = -2; i < 3; i++)  {  for (int j = -2; j < 3; j++)  {  if (x + i >= 0 && x + i < n && y + j >= 0 && y + j < n)  {  if (expl[i + 2, j + 2] == 1)  {  if (arr[x + i, y + j] > 0) counter++;  arr[x + i, y + j] = -1;  }  }  }  }  return counter;  } | private int ExplodeMine(int[,] field)  {  int row = 0, col = 0;  SetNextMinePosition(field, out row, out col);  Mine mine = new Mine(field[row, col]);  int[,] explodeType = mine.ExplodeType();  int explodeMinesCount = 0;  for (int i = -2; i < 3; i++)  {  for (int j = -2; j < 3; j++)  {  if (row + i >= 0 && row + i < field.GetLength(0) && col + j >= 0 &&  col + j < field.GetLength(1))  {  if (explodeType[i + 2, j + 2] == 1)  {  if (field[row + i, col + j] > 0)  {  explodeMinesCount++;  }  field[row + i, col + j] = -1;  }  }  }  }  return explodeMinesCount;  } |

* Extract methods **CreateGameField, Play** and **GetInitialSize** from **InitiateGame**:
* Method **GetInitialSize** gets input data for size of the field
* Method **CreateGameField** creates instance of class **Field** than fills and prints the field
* Method **Play** calls **ExplodeMine** while the game over and prints the final result:
* Change variable names to more appropriate ones
* Add empty lines between logically different pieces
* Removing unnecessary comments
* Replace chinese text to appropriate Еnglish

|  |  |
| --- | --- |
| int 爆 = 0;  while (mineNumber > 0)  {  int tmp = TimeToPlay(arr, n);  mineNumber -= tmp;  PrintField(arr, n);  //Console.WriteLine("Mines Blowed this round: {0}",tmp);  爆++;  }  Console.WriteLine("游戏结束。引爆地雷：{0}", 爆); | private void Play()  {  int minesNumber = this.Field.NumberOfMines;  int turns = 0;  while (minesNumber > 0)  {  int explodedMines = ExplodeMine(this.Field.MatrixForField);  minesNumber -= explodedMines;  this.Field.PrintField();  turns++;  }  Console.WriteLine("Game over -> detonated mines: {0}", turns);  } |
|  |  |