using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TurtleGraphics

{

class Program

{

static void Main(string[] args)

{

try

{ doWork(); }

catch (Exception Ex)

{ Console.WriteLine(Ex.Message); }

}

public static void doWork()

{

#region DEFINITIONS

Turtle player = new Turtle('E', 0, 0, true);

int[,] floor = new int[50, 50];

#region INITIALIZATION FORS

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

{

for (int j = 0; j < floor.GetLength(1); j++)

{

floor[i, j] = 0;

}

}

#endregion

#endregion

Console.Clear();

player.Print(floor);

Turtle.menu(player, floor);

}

}

}

namespace TurtleGraphics

{

class Turtle

{

private char direction;

public char Direction

{

get { return direction; }

set { direction = value; }

}

private int x;

public int X

{

get { return x; }

set

{

if (x > 19)

value = 19;

else

value = x;

}

}

private int y;

public int Y

{

get { return y; }

set

{

if (y > 19)

value = 19;

else

value = y;

}

}

private bool pen;

public bool Pen

{

get { return pen; }

set { pen = value; }

}

public Turtle(char direction, int x, int y, bool pen)

{

this.direction = direction;

this.x = x;

this.y = y;

this.pen = pen;

}

public void PenUp()

{

pen = false;

}

public void PenDown()

{

pen = true;

}

public void TurnRight(int[,] floor)

{

#region DIRECTION

switch (direction)

{

case 'N': direction = 'E'; break;

case 'S': direction = 'W'; break;

case 'E': direction = 'S'; break;

case 'W': direction = 'N'; break;

default: break;

}

#endregion

}

public void TurnLeft(int[,] floor)

{

#region DIRECTION

switch (Direction)

{

case 'N': direction = 'W'; break;

case 'S': direction = 'E'; break;

case 'E': direction = 'N'; break;

case 'W': direction = 'S'; break;

default: break;

}

#endregion

}

public void MoveForward(int[,] floor)

{

Console.Write("How many steps would you like to go? :");

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

#region DIRECTION

switch (direction)

{

case 'N':

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

{

if (pen == true)

floor[x, i] = 1;

else

floor[x, i] = 0;

}

y = y + n;

break;

case 'S':

for (int i = y - n; i <= y; i++)

{

if (pen == true)

floor[x, i] = 1;

else

floor[x, i] = 0;

}

y = y - n;

break;

case 'E':

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

{

if (pen == true)

floor[i, y] = 1;

else

floor[i, y] = 0;

}

x = x + n;

break;

case 'W':

for (int i = x - n; i <= x; i++)

{

if (pen == true)

floor[i, y] = 1;

else

floor[i, y] = 0;

}

x = x - n;

break;

}

floor[x, y] = 2;

#endregion // CONTROL HERE FINALLY!

}

public void Print(int[,] floor)

{

Console.WriteLine("THE SHAPE");

#region NESTED FORS FOR PRINTING

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

{

for (int j = 0; j < floor.GetLength(1); j++)

{

if (floor[i, j] == 0)

Console.Write(".");

else if (floor[i, j] == 1)

Console.Write("\*");

else if (floor[i, j] == 2)

Console.Write("T");

}

Console.WriteLine();

}

#endregion

}

public static void menu(Turtle player, int[,] floor)

{

#region MENU WRITINGS

Console.WriteLine("1 Pen UP");

Console.WriteLine("2 Pen DOWN");

Console.WriteLine("3 Turn RIGHT");

Console.WriteLine("4 Turn LEFT");

Console.WriteLine("5 Move FORWARD");

Console.WriteLine("6 Print");

Console.WriteLine("9 End of Data (Sentinel)");

#endregion

Console.WriteLine("Enter the command you wanna realize? :");

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

#region COMMAND SWITCH FUNCTIONEL

switch (command)

{

case 1: player.PenUp(); Program.doWork(); break;

case 2: player.PenDown(); Program.doWork(); break;

case 3: player.TurnRight(floor); Program.doWork(); break;

case 4: player.TurnLeft(floor); Program.doWork(); break;

case 5: player.MoveForward(floor); player.Print(floor); break;

case 6: player.Print(floor); Program.doWork(); break;

case 9: Console.Clear(); player.Print(floor); break;

default: Console.WriteLine("You entered undefined function, program has ended!"); break;

}

#endregion

// THIS IS FOR FUNCTIONALIZING THE MENUS

}

}

}