
A-level
COMPUTER SCIENCE
(7517/1A)

Paper 1 C#

Skeleton Program

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

namespace CSPreALevelSkeleton
{
    class Program
    {
        public const int NS = 4;
        public const int WE = 6;

        public struct CellReference
        {
            public int NoOfCellsEast;
            public int NoOfCellsSouth;
        }

        class Game
        {
            private Character Player = new Character();
            private Grid Cavern = new Grid();
            private Enemy Monster = new Enemy();
            private Item Flask = new Item();
            private Trap Trap1 = new Trap();
            private Trap Trap2 = new Trap();
            private Boolean TrainingGame;

            public Game(Boolean IsATrainingGame)
            {
                TrainingGame = IsATrainingGame;
                SetUpGame();
                Play();
            }

            public void Play()
            {
                int Count;
                Boolean Eaten;
                Boolean FlaskFound;
                char MoveDirection;
                Boolean ValidMove;
                CellReference Position;
                Eaten = false;
                FlaskFound = false;
                Cavern.Display(Monster.GetAwake());
                do
                {
                    do
                    {
                        DisplayMoveOptions();
                        MoveDirection = GetMove();
                        ValidMove = CheckValidMove(MoveDirection);
                    } while (!ValidMove);
                    if (MoveDirection != 'M')
                    {
```

```

        Cavern.PlaceItem(Player.GetPosition(), ' ');
        Player.MakeMove(MoveDirection);
        Cavern.PlaceItem(Player.GetPosition(), '*');
        Cavern.Display(Monster.GetAwake());
        FlaskFound =
Player.CheckIfSameCell(Flask.GetPosition());
        if (FlaskFound)
        {
            DisplayWonGameMessage();
        }
        Eaten =
Monster.CheckIfSameCell(Player.GetPosition());
// This selection structure checks to see if the
player has
// triggered one of the traps in the cavern
if (!Monster.GetAwake() && !FlaskFound && !Eaten &&
((Player.CheckIfSameCell(Trap1.GetPosition())
    && !Trap1.GetTriggered()) ||
(Player.CheckIfSameCell(Trap2.GetPosition()) && !Trap2.GetTriggered())))
    {
        Monster.ChangeSleepStatus();
        DisplayTrapMessage();
        Cavern.Display(Monster.GetAwake());
    }
if (Monster.GetAwake() && !Eaten && !FlaskFound)
    {
        Count = 0;
        do
        {
            Cavern.PlaceItem(Monster.GetPosition(), ' ');
            Position = Monster.GetPosition();
            Monster.MakeMove(Player.GetPosition());
            Cavern.PlaceItem(Monster.GetPosition(), 'M');
            if
(Monster.CheckIfSameCell(Flask.GetPosition()))
                {
                    Flask.SetPosition(Position);
                    Cavern.PlaceItem(Position, 'F');
                }
            Eaten =
Monster.CheckIfSameCell(Player.GetPosition());
            Console.WriteLine();
            Console.WriteLine("Press Enter key to
continue");
            Console.ReadLine();
            Cavern.Display(Monster.GetAwake());
            Count = Count + 1;
        } while (Count != 2 && !Eaten);
    }
if (Eaten)
    {
        DisplayLostGameMessage();
    }
}
} while (!Eaten && !FlaskFound && MoveDirection != 'M');
}

```

```
public void DisplayMoveOptions()
{
    Console.WriteLine();
    Console.WriteLine("Enter N to move NORTH");
    Console.WriteLine("Enter S to move SOUTH");
    Console.WriteLine("Enter E to move EAST");
    Console.WriteLine("Enter W to move WEST");
    Console.WriteLine("Enter M to return to the Main Menu");
    Console.WriteLine();
}

public char GetMove()
{
    char Move;
    Move = char.Parse(Console.ReadLine());
    Console.WriteLine();
    return Move;
}

public void DisplayWonGameMessage()
{
    Console.WriteLine("Well done! you have found the flask
containing the Styxian potion.");
    Console.WriteLine("You have won the game of MONSTER!");
    Console.WriteLine();
}

public void DisplayTrapMessage()
{
    Console.WriteLine("Oh no! You have set off a trap. Watch out,
the monster is now awake!");
    Console.WriteLine();
}

public void DisplayLostGameMessage()
{
    Console.WriteLine("ARGHHHHHH! The monster has eaten you. GAME
OVER.");
    Console.WriteLine("Maybe you will have better luck next time
you play MONSTER!");
    Console.WriteLine();
}

public Boolean CheckValidMove(char Direction)
{
    Boolean ValidMove;
    ValidMove = true;
    if (!(Direction == 'N' || Direction == 'S' || Direction ==
'W' || Direction == 'E' || Direction == 'M'))
    {
        ValidMove = false;
    }
    return ValidMove;
}
```

```

public CellReference SetPositionOfItem(char Item)
{
    CellReference Position;
    do
    {
        Position = GetNewRandomPosition();
    } while (!Cavern.IsCellEmpty(Position));
    Cavern.PlaceItem(Position, Item);
    return Position;
}

public void SetUpGame()
{
    CellReference Position;
    Cavern.Reset();
    if (!TrainingGame)
    {
        Position.NoOfCellsEast = 0;
        Position.NoOfCellsSouth = 0;
        Player.SetPosition(Position);
        Cavern.PlaceItem(Position, '*');
        Trap1.SetPosition(SetPositionOfItem('T'));
        Trap2.SetPosition(SetPositionOfItem('T'));
        Monster.SetPosition(SetPositionOfItem('M'));
        Flask.SetPosition(SetPositionOfItem('F'));
    }
    else
    {
        Position.NoOfCellsEast = 4;
        Position.NoOfCellsSouth = 2;
        Player.SetPosition(Position);
        Cavern.PlaceItem(Position, '*');
        Position.NoOfCellsEast = 6;
        Position.NoOfCellsSouth = 2;
        Trap1.SetPosition(Position);
        Cavern.PlaceItem(Position, 'T');
        Position.NoOfCellsEast = 4;
        Position.NoOfCellsSouth = 3;
        Trap2.SetPosition(Position);
        Cavern.PlaceItem(Position, 'T');
        Position.NoOfCellsEast = 4;
        Position.NoOfCellsSouth = 0;
        Monster.SetPosition(Position);
        Cavern.PlaceItem(Position, 'M');
        Position.NoOfCellsEast = 3;
        Position.NoOfCellsSouth = 1;
        Flask.SetPosition(Position);
        Cavern.PlaceItem(Position, 'F');
    }
}

public CellReference GetNewRandomPosition()
{
    CellReference Position;
    Random rnd = new Random();
    Position.NoOfCellsSouth = rnd.Next(0, NS + 1);
}

```

```

        Position.NoOfCellsEast = rnd.Next(0, WE + 1);
        return Position;
    }
}

class Grid
{
    private char[,] CavernState = new char[NS + 1, WE + 1];

    public void Reset()
    {
        int Count1;
        int Count2;
        for (Count1 = 0; Count1 <= NS; Count1++)
        {
            for (Count2 = 0; Count2 <= WE; Count2++)
            {
                CavernState[Count1, Count2] = ' ';
            }
        }
    }

    public void Display(Boolean MonsterAwake)
    {
        int Count1;
        int Count2;
        for (Count1 = 0; Count1 <= NS; Count1++)
        {
            Console.WriteLine(" ----- ");
            for (Count2 = 0; Count2 <= WE; Count2++)
            {
                if (CavernState[Count1, Count2] == ' ' ||
CavernState[Count1, Count2] == '*' || (CavernState[Count1, Count2] == 'M' &&
MonsterAwake))
                {
                    Console.Write("|" + CavernState[Count1, Count2]);
                }
                else
                {
                    Console.Write("| ");
                }
            }
            Console.WriteLine("|");
        }
        Console.WriteLine(" ----- ");
        Console.WriteLine();
    }

    public void PlaceItem(CellReference Position, char Item)
    {
        CavernState[Position.NoOfCellsSouth, Position.NoOfCellsEast]
= Item;
    }

    public Boolean IsCellEmpty(CellReference Position)

```

```

        {
            if (CavernState[Position.NoOfCellsSouth,
Position.NoOfCellsEast] == ' ')
                return true;
            else
                return false;
        }
    }

class Enemy : Item
{
    private Boolean Awake;

    public virtual void MakeMove(CellReference PlayerPosition)
    {
        if (NoOfCellsSouth < PlayerPosition.NoOfCellsSouth)
        {
            NoOfCellsSouth = NoOfCellsSouth + 1;
        }
        else
            if (NoOfCellsSouth > PlayerPosition.NoOfCellsSouth)
            {
                NoOfCellsSouth = NoOfCellsSouth - 1;
            }
            else
                if (NoOfCellsEast < PlayerPosition.NoOfCellsEast)
                {
                    NoOfCellsEast = NoOfCellsEast + 1;
                }
                else
                {
                    NoOfCellsEast = NoOfCellsEast - 1;
                }
            }
    }

    public Boolean GetAwake()
    {
        return Awake;
    }

    public virtual void ChangeSleepStatus()
    {
        if (!Awake)
            Awake = true;
        else
            Awake = false;
    }

    public Enemy()
    {
        Awake = false;
    }
}

class Character : Item
{

```

```
public void MakeMove(char Direction)
{
    switch (Direction)
    {
        case 'N': NoOfCellsSouth = NoOfCellsSouth - 1;
                 break;
        case 'S': NoOfCellsSouth = NoOfCellsSouth + 1;
                 break;
        case 'W': NoOfCellsEast = NoOfCellsEast - 1;
                 break;
        case 'E': NoOfCellsEast = NoOfCellsEast + 1;
                 break;
    }
}

class Trap : Item
{
    private Boolean Triggered;

    public Boolean GetTriggered()
    {
        return Triggered;
    }

    public Trap()
    {
        Triggered = false;
    }

    public void ToggleTrap()
    {
        Triggered = !Triggered;
    }
}

class Item
{
    protected int NoOfCellsEast;
    protected int NoOfCellsSouth;

    public CellReference GetPosition()
    {
        CellReference Position;
        Position.NoOfCellsEast = NoOfCellsEast;
        Position.NoOfCellsSouth = NoOfCellsSouth;
        return Position;
    }

    public void SetPosition(CellReference Position)
    {
        NoOfCellsEast = Position.NoOfCellsEast;
        NoOfCellsSouth = Position.NoOfCellsSouth;
    }

    public Boolean CheckIfSameCell(CellReference Position)

```

```

        {
            if (NoOfCellsEast == Position.NoOfCellsEast && NoOfCellsSouth
== Position.NoOfCellsSouth)
                {
                    return true;
                }
            else
                {
                    return false;
                }
        }
    }

static void Main(string[] args)
{
    int Choice = 0;
    while (Choice != 9)
    {
        DisplayMenu();
        Choice = GetMainMenuChoice();
        switch (Choice)
        {
            case 1: Game NewGame = new Game(false);
                    break;
            case 2: Game TrainingGame = new Game(true);
                    break;
        }
    }
}

public static void DisplayMenu()
{
    Console.WriteLine("MAIN MENU");
    Console.WriteLine();
    Console.WriteLine("1. Start new game");
    Console.WriteLine("2. Play training game");
    Console.WriteLine("9. Quit");
    Console.WriteLine();
    Console.Write("Please enter your choice: ");
}

public static int GetMainMenuChoice()
{
    int Choice;
    Choice = int.Parse(Console.ReadLine());
    Console.WriteLine();
    return Choice;
}
}
}

```

