SPECIMEN MATERIAL

A-level
COMPUTER SCIENCE
(7517/1B)

Paper 1 Java

Skeleton Program

```
package javaapreskeleton;

import java.util.Random;


public class Main {

    final int NS = 4;
    final int WE = 6;
    AQAConsole console = new AQAConsole();

    class 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)
        {
            this.trainingGame = isATrainingGame;
            setUpGame();
            play();
        }

        public void play()
        {
            int count;
            boolean eaten;
            boolean flaskFound;
            char moveDirection;
            boolean validMove;
            CellReference position = new CellReference();
            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.println();
                            console.println("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.println();
            console.println("Enter N to move NORTH");
```

```
            console.println("Enter S to move SOUTH");
            console.println("Enter E to move EAST");
            console.println("Enter W to move WEST");
            console.println("Enter M to return to the Main Menu");
            console.println();
        }

        public char getMove()
        {
            char move;
            move = console.readChar();
            console.println();
            return move;
        }

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

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

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

        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 = new CellReference();
            do
            {
                position = getNewRandomPosition();
```

```
        } while (!cavern.isCellEmpty(position));
        cavern.placeItem(position, item);
        return position;
    }

    public void setUpGame()
    {
        CellReference position = new CellReference();
        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 = new CellReference();
        Random rnd = new Random();
        position.noOfCellsSouth = rnd.nextInt(NS + 1);
        position.noOfCellsEast = rnd.nextInt(WE + 1);
        return position;
    }
}

class Grid
```

```
    {
        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.println(" ------------- ");
                for (count2 = 0; count2 <= WE; count2++)
                {
                    if (cavernState[count1][count2] == ' ' ||
cavernState[count1][count2] == '*' || cavernState[count1][count2] == 'M' &&
monsterAwake)
                    {
                        console.print("|" + cavernState[count1][count2]);
                    }
                    else
                    {
                        console.print("| ");
                    }
                }
                console.println("|");
            }
            console.println(" ------------- ");
            console.println();
        }

        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 extends Item
{
    private boolean awake;

    public 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 void changeSleepStatus()
    {
        if (!awake)
            awake = true;
        else
            awake = false;
    }

    public Enemy()
    {
        awake = false;
    }
}

class Character extends 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 extends 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 = new CellReference();
            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;
            }
```

```
        }
    }

    public Main()
    {
        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 void displayMenu()
    {
        console.println("MAIN MENU");
        console.println();
        console.println("1. Start new game");
        console.println("2. Play training game");
        console.println("9. Quit");
        console.println();
        console.print("Please enter your choice: ");
    }


    public int getMainMenuChoice()
    {
        int choice;
        choice = console.readInteger("");
        console.println();
        return choice;
    }

    public static void main(String[] args)
    {
        new Main();
    }
}
```