Scenario: You are building a simple text-based adventure game where the player explores a virtual forest. The forest is represented by a 2D array of characters, where:

'T': Represents a tree (impassable)

'.': Represents an open space (passable)

'P': Represents the player's current location

Objective: Write a Java program that fulfills the following functionalities:

Forest Generation:

Define a static method generateForest(int rows, int cols) that takes the desired number of rows and columns and returns a 2D character array representing the forest.

The method should randomly populate the array with '.' (open space) and 'T' (trees) with a probability of 70% open space and 30% trees.

Place the player 'P' at a random empty location within the forest. (Make sure the chosen location is not a tree 'T').

Display Forest:

Define a static method displayForest(char[][] forest) that takes the forest array and prints it to the console, representing each character as a visual representation (e.g., 'T' as a tree symbol, '.' as a space).

Player Movement:

Define a method movePlayer(char[][] forest, char direction) that takes the forest array and a direction ('W' for Up, 'S' for Down, 'A' for Left, 'D' for Right) as input.

The method should check if the move is valid (within forest boundaries and not a tree) and update the player's location in the forest array if valid.

If the move is invalid, print an appropriate message to the user.

```
import java.util.Random;
import java.util.Scanner;


public class ForestAdventure {


    public static void main(String[] args) {
        int rows = 10;
        int cols = 10;
        char[][] forest = generateForest(rows, cols);
        displayForest(forest);


        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("Enter your move (WASD to move, Q to quit): ");
            char move = scanner.next().toUpperCase().charAt(0);
            if (move == 'Q') {
                System.out.println("Quitting the game.");
                break;
            }
            movePlayer(forest, move);
```

```
        displayForest(forest);
    }
    scanner.close();
}

public static char[][] generateForest(int rows, int cols) {
    char[][] forest = new char[rows][cols];
    Random rand = new Random();
    int playerRow = 0;
    int playerCol = 0;

    // Populate the forest with trees and open spaces
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            forest[i][j] = rand.nextInt(100) < 30 ? 'T' : '.';
        }
    }

    // Place the player in a random open space
    do {
        playerRow = rand.nextInt(rows);
        playerCol = rand.nextInt(cols);
    } while (forest[playerRow][playerCol] == 'T');

    forest[playerRow][playerCol] = 'P';
    return forest;
}
```

```
public static void displayForest(char[][] forest) {
    for (int i = 0; i < forest.length; i++) {
        for (int j = 0; j < forest[i].length; j++) {
            System.out.print(forest[i][j] + " ");
        }
        System.out.println();
    }
}

public static void movePlayer(char[][] forest, char direction) {
    int playerRow = -1, playerCol = -1;

    // Find the player's current location
    outer:
    for (int i = 0; i < forest.length; i++) {
        for (int j = 0; j < forest[i].length; j++) {
            if (forest[i][j] == 'P') {
                playerRow = i;
                playerCol = j;
                break outer;
            }
        }
    }

    // Determine new position based on direction
    int newRow = playerRow, newCol = playerCol;
```

```
        switch (direction) {
            case 'W':
                newRow--;
                break;
            case 'S':
                newRow++;
                break;
            case 'A':
                newCol--;
                break;
            case 'D':
                newCol++;
                break;
            default:
                System.out.println("Invalid move. Use WASD keys to move.");
                return;
        }

        // Check if the new position is valid
        if (newRow >= 0 && newRow < forest.length && newCol >= 0 && newCol < forest[0].length)
{
            if (forest[newRow][newCol] == '.') {
                forest[playerRow][playerCol] = '.';
                forest[newRow][newCol] = 'P';
            } else {
                System.out.println("Invalid move. You can't move into a tree.");
            }
```

```
        } else {
            System.out.println("Invalid move. You are trying to move out of the forest.");
        }
    }
}
```