OOPS WITH DESIGN PATTERNS

Part 1 Creating Game Material

GROUP 1

Part 1 Creating Game Material

DUE DATE - 14TH OCTOBER ,2023

SUBMITTED DATE - 14TH OCTOBER , 2023

Table of Contents

SOURCE CODE

MAIN.CPP

```
#include <iostream>
#include "Dice.h"
#include "Square.h"

int main() {
    Dice randomGen;

    std::cout << "Random number between 1 and 6: " <<
randomGen.getRandomNumber() << std::endl;

    Square game(6,9);
    game.initializeBoard();
    //game.placeObstacles();
    game.displayBoard();

    return 0;
}
```

Square.h

```
#ifndef SQUARE_H
#define SQUARE_H

#include <string>
#include <iostream>
#include <iomanip>


class Square {
public:
   Square() = default;
   virtual ~Square() = default;

   virtual void display() const = 0;
};

#endif
```

Normal square.h

```
#ifndef NORMAL_SQUARE_H
#define NORMAL_SQUARE_H
#include "Square.h"


class NormalSquare : public Square {
private:
   std::string label;

public:
   NormalSquare(const std::string& label): label(label) {}

   void display() const override {
      std::cout << std::setw(5) << label;
   }
};

#endif
```

Obstacle.h

```
#ifndef OBSTACLE_H
#define OBSTACLE_H
#include "Square.h"


class Obstacle : public Square {
public:
   Obstacle(const std::string& label) : Square(label) {}
   virtual void display() const override {
      std::cout << std::setw(5) << label;
   }
};

#endif
```

Gameboard.h

```
#ifndef GAMEBOARD_H
```

```
#define GAMEBOARD_H
#include <vector>
#include <string>


class GameBoard {
public:
    GameBoard(int tracks, int columns);
    void initializeBoard();
    void placeObstacles();
    void displayBoard();

private:
    int tracks;
    int columns;
    std::vector<std::vector<std::string>> board;
};

#endif
```

Gameboard.cpp

```
#include "GameBoard.h"
#include <iostream>
#include <random>
#include <algorithm>
#include <iterator>
#include <numeric>
#include <iomanip>

GameBoard::GameBoard(int tracks, int columns) : tracks(tracks), columns(columns) {
    board.resize(tracks, std::vector<std::string>(columns, " "));
}

void GameBoard::initializeBoard() {
    for (int t = 0; t < tracks; ++t) {
        board[t][0] = "T" + std::to_string(t + 1) + "S"; // Start squares
        for (int c = 1; c < columns - 1; ++c) {
            board[t][c] = "T" + std::to_string(t + 1) + static_cast<char>('a' + c - 1);
        }
        board[t][columns - 1] = "T" + std::to_string(t + 1) + "Z";
    }
}
```

```
void GameBoard::placeObstacles() {
    std::vector<int> columnIndices(columns - 3);
    std::iota(columnIndices.begin(), columnIndices.end(), 2);

    std::random_device rd;
    std::mt19937 g(rd());
    std::shuffle(columnIndices.begin(), columnIndices.end(), g);

    for (int t = 0; t < tracks; ++t) {

        int obstacleColumn = columnIndices[t % columnIndices.size()];
        board[t][obstacleColumn] = "Obs" + std::to_string(obstacleColumn - 1);
    }
}

void GameBoard::displayBoard() {
    for (const auto& row : board) {
        for (const auto& cell : row) {
            std::cout << std::setw(5) << cell;
        }
        std::cout << std::endl;
    }
}
```

NormalObstacle.h

```
#ifndef NORMAL_OBSTACLE_H
#define NORMAL_OBSTACLE_H
#include "Obstacle.h"

class NormalObstacle : public Obstacle {
public:
    NormalObstacle(const std::string& label) : Obstacle(label) {}
};

#endif
```

Shallowpit.h

```
#ifndef SHALLOW_PIT_H
#define SHALLOW_PIT_H

#include "Obstacle.h"

class ShallowPit : public Obstacle {
public:
   ShallowPit(const std::string& label) : Obstacle(label) {}
};

#endif
```

deepPit.h

```
#ifndef DEEP_PIT_H
#define DEEP_PIT_H
#include "Obstacle.h"

class DeepPit : public Obstacle {
public:
   DeepPit(const std::string& label) : Obstacle(label) {}
};

#endif
```

Wormhole.h

```
#ifndef WORM_HOLE_H
#define WORM_HOLE_H

#include "Obstacle.h"

class WormHole : public Obstacle {
public:
   WormHole(const std::string& label) : Obstacle(label) {}
};

#endif
```

Blackhole.h

```
#ifndef BLACK_HOLE_H
#define BLACK_HOLE_H

#include "Obstacle.h"

class BlackHole : public Obstacle {
public:
    BlackHole(const std::string& label) : Obstacle(label) {}
};

#endif
```

OUTPUT

```
T1S  T1a  T1b  T1c  T1d  T1e Obs5  T1g  T1Z
T2S  T2a  T2b  T2c  T2d Obs4  T2f  T2g  T2Z
T3S  T3a  T3b Obs2  T3d  T3e  T3f  T3g  T3Z
T4S  T4a  T4b  T4c  T4d  T4e  T4f Obs6  T4Z
T5S  T5a  T5b  T5c Obs3  T5e  T5f  T5g  T5Z
T6S  T6a Obs1  T6c  T6d  T6e  T6f  T6g  T6Z
```

THINGS I LEARNED

In this assignment, I gained practical experience inheritance and how to orange a project for move forward and also the importance of planning all the elements and to have a clear picture before starting to code.

WORK CITED

1. http://nifty.stanford.edu/2012/kurmas-igel-argern/
2. chatgpt.