Федеральное государственное бюджетное образовательное учреждение
высшего образования «Сибирский государственный университет
телекоммуникаций и информатики»

Факультет ИВТ

Кафедра ПМ и К

Курсовая работа

По дисциплине «Объектно-ориентированное программирование»

Тема: 5. Игра «крестики-нолики».

Выполнил:студент гр. ИП-216
Русецкий А.С.

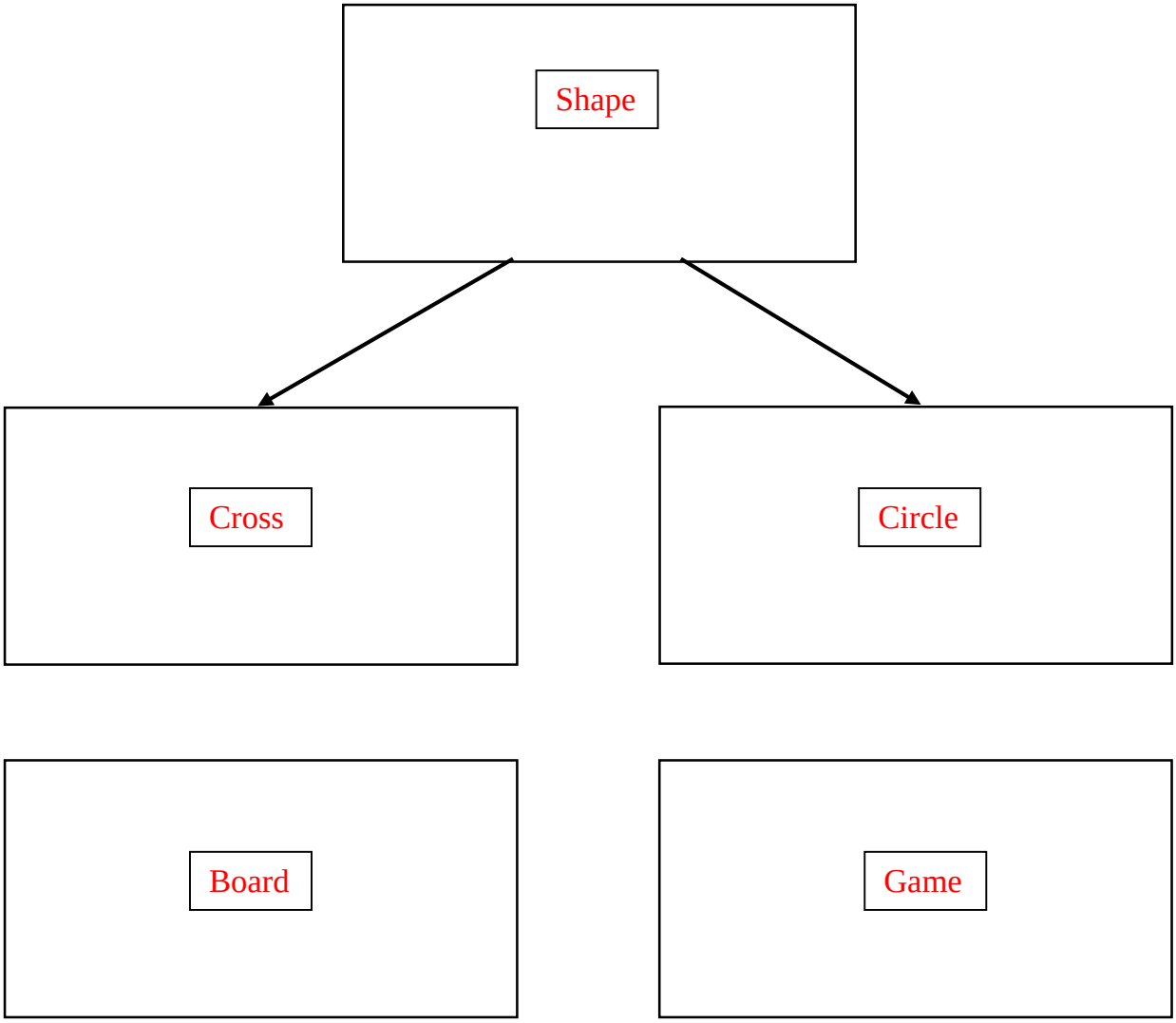
Проверил:доцент кафедры ПМ и К
Ситняковская Е. И.

Новосибирск, 2023

Содержание

Постановка задачи: Написать программу с использованием объектно-ориентированных технологий. Иерархия классов должна включать минимум четыре класса, один из которых – абстрактный. Результатом работы станет программа/игра «крестики-нолики» по всем канонам ООП.

Технологии ООП: В программе реализовано 5 классов с такой иерархией:



В этой программе используются следующие технологии ООП:

1. Наследование: классы Cross и Circle наследуются от абстрактного класса Shape.

2. Полиморфизм: метод getSymbol() в классах Cross и Circle переопределен из абстрактного класса Shape, что позволяет использовать их объекты через указатель на Shape.

3. Абстракция: абстрактный класс Shape предоставляет интерфейс для классов Cross и Circle, определяя только метод getSymbol().

4. Инкапсуляция: класс Board инкапсулирует поле cells и методы для взаимодействия с ним, скрывая детали его реализации от других частей программы.

5. Конструктор: конструктор класса Game используется для инициализации поля currentPlayer. Конструктор Board(): создает пустую игровую доску.

Структура классов:

Класс Shape является абстрактным классом, который содержит чисто виртуальную функцию getSymbol(). Этот класс определяет общий интерфейс для фигур игры.

Классы Cross и Circle наследуются от класса Shape. Они реализуют функцию getSymbol() и возвращают символы 'X' и 'O' соответственно. Эти классы представляют крестик и нолик игры.

Класс Board представляет игровую доску. Он содержит приватное поле cells, которое является двумерным массивом указателей на объекты класса Shape. Конструктор класса Board инициализирует все элементы массива значением nullptr.
Класс Board также содержит функции makeMove(), print() и checkWin(). Функция makeMove() позволяет сделать ход на игровой доске, помещая символ указанной формы на указанную позицию. Функция print() выводит текущее состояние игровой доски на экран. Функция checkWin() проверяет, выиграл ли указанный символ (форма) на игровой доске.

Класс Game содержит поле board (игровая доска), поле currentPlayer (текущий игрок), поле crossPlayer (игрок-крестики) и поле circlePlayer (игрок-нолики). Класс Game имеет конструктор, который устанавливает currentPlayer на игрока-крестики.
Класс Game содержит метод play(), который запускает игру:
- Выводит игровую доску на экран
- Выводит символ текущего игрока на экран
- Запрашивает у пользователя координаты для хода
- Проверяет, был ли ход сделан успешно
- Если ход сделан успешно:
  - Проверяет, не победил ли текущий игрок
  - Если победитель определен, выводит сообщение о победе и завершает игру
  - Меняет currentPlayer на другого игрока
- Если ход сделан неправильно, выводит сообщение об ошибке

Общая структура классов выглядит следующим образом:

- Абстрактный класс Shape
  - Функция getSymbol()

- Класс Cross наследуется от класса Shape
  - Функция getSymbol()
- Класс Circle наследуется от класса Shape
  - Функция getSymbol()

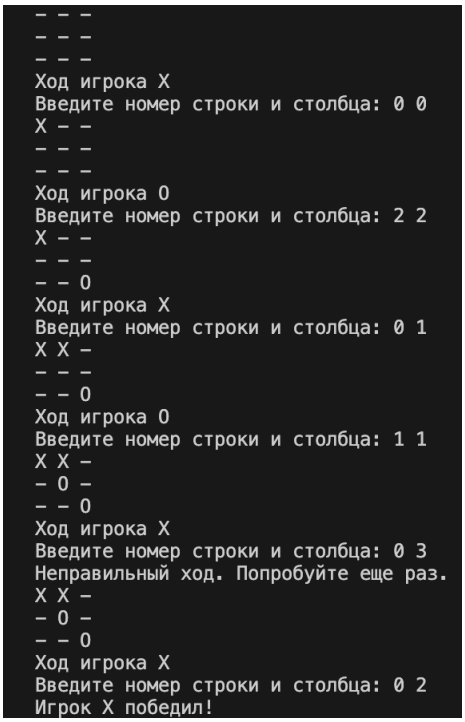
- Класс Board
  - Приватное поле cells[3][3]
  - Конструктор Board()
  - Метод makeMove()
  - Метод print()

- Метод checkWin()

- Класс Game.
  - Приватные поля:
    board
    currentPlayer
    crossPlayer
    circlePlayer
Конструктор Game()
Метод play()

Программная реализация: программа была написана на языке программирования в качестве среды разработки была выбрана программа Visual Studio Code.

Результаты работы: Чтобы проверить работоспособность программы, в функции main(): создается объект класса Game и вызывается метод play() для запуска игры:



Заключение: В процессе написания данной курсовой работы была рассмотрена постановка задачи, связанной с программированием на языке ООП. Были изучены основные технологии ООП, а также рассмотрена структура классов и их взаимосвязи в рамках разрабатываемой программы. Программная реализация представляет собой создание и описание классов, их методов и свойств, а также разработку функционала программы с использованием принципов ООП. В ходе работы были применены различные концепции, такие как наследование, полиморфизм, инкапсуляция, что позволило создать гибкую и расширяемую систему.

Использованные источники:

Приложение. Листинг

```
#include <iostream>
using namespace std;

// Абстрактный класс Фигура
class Shape {
public:
    virtual char getSymbol() const = 0;
    virtual ~Shape() {}
};

// Класс Крестик
class Cross : public Shape {
public:
    char getSymbol() const {
        return 'X';
    }
};

// Класс Нолик
class Circle : public Shape {
public:
    char getSymbol() const  {
        return 'O';
    }
};

// Класс Игровая доска
class Board {
private:
    Shape* cells[3][3];
public:
    // Конструктор Board(): Создает пустую игровую доску, заполняя все ячейки значением nullptr.
    Board() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                cells[i][j] = nullptr;
            }
        }
    }

    bool makeMove(int row, int col, Shape* shape) {
    if (row >= 0 && row < 3 && col >= 0 && col < 3 && cells[row][col] == nullptr) {
        cells[row][col] = shape;
        if (isBoardFull()) {
            cout << "Ничья!" << endl;
```

```
            exit(0); // Завершаем программу при ничье
        }
        return true;
    }
    return false;
    }

    bool isBoardFull() const {
        for (int i = 0; i < 3; ++i) {
            for (int j = 0; j < 3; ++j) {
                if (cells[i][j] == nullptr) {
                    return false; // Найдена пустая клетка, игровое поле не заполнено
                }
            }
        }
        return true; // Нет пустых клеток, игровое поле заполнено
    }


    // Выводит текущее состояние игровой доски на экран. Если ячейка пуста, то выводится символ "-". Иначе выводится символ.
    void print() const {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (cells[i][j] == nullptr) {
                    cout << '-';
                } else {
                    cout << cells[i][j]->getSymbol();
                }
                cout << ' ';
            }
            cout << endl;
        }
    }

    // Функция проверяет, выиграл ли указанный символ (shape) на игровой доске.
    bool checkWin(const Shape* shape) const {
        for (int i = 0; i < 3; i++) {
            if (cells[i][0] != nullptr && cells[i][0]->getSymbol() == shape->getSymbol() &&
                cells[i][1] != nullptr && cells[i][1]->getSymbol() == shape->getSymbol() &&
                cells[i][2] != nullptr && cells[i][2]->getSymbol() == shape->getSymbol()) {
                return true; // Победа по горизонтали
            }
            if (cells[0][i] != nullptr && cells[0][i]->getSymbol() == shape->getSymbol() &&
                cells[1][i] != nullptr && cells[1][i]->getSymbol() == shape->getSymbol() &&
                cells[2][i] != nullptr && cells[2][i]->getSymbol() == shape->getSymbol()) {
                return true; // Победа по вертикали
            }
        }
        if ((cells[0][0] != nullptr && cells[0][0]->getSymbol() == shape->getSymbol() &&
             cells[1][1] != nullptr && cells[1][1]->getSymbol() == shape->getSymbol() &&
             cells[2][2] != nullptr && cells[2][2]->getSymbol() == shape->getSymbol()) ||
```

```
            (cells[0][2] != nullptr && cells[0][2]->getSymbol() == shape->getSymbol() &&
             cells[1][1] != nullptr && cells[1][1]->getSymbol() == shape->getSymbol() &&
             cells[2][0] != nullptr && cells[2][0]->getSymbol() == shape->getSymbol())) {
            return true; // Победа по диагоналям
        }
        return false;
    }

};

// Класс Игра
class Game {
private:
    Board board;
    Shape* currentPlayer; // текущий игрок
    Cross crossPlayer; // игрок-крестики
    Circle circlePlayer; // игрок-нолики
public:
    Game() {
        currentPlayer = &crossPlayer;
    }
    void play() {
        int row, col;
        while (true) {
            board.print();
            cout << "Ход игрока " << currentPlayer->getSymbol() << endl;
            cout << "Введите номер строки и столбца: ";
            cin >> row >> col;
            if (board.makeMove(row, col, currentPlayer)) {
                if (board.checkWin(currentPlayer)) {
                    cout << "Игрок " << currentPlayer->getSymbol() << " победил!" << endl;
                    break;
                }
                if (currentPlayer == &crossPlayer) {
                    currentPlayer = &circlePlayer;
                } else {
                    currentPlayer = &crossPlayer;
                }
            } else {
                cout << "Неправильный ход. Попробуйте еще раз." << endl;
            }
        }

    }
};

int main() {
    Game game;
    game.play();
    return 0;
}
```