**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Доклад на тему:**

Принципы ООП

Выполнил:

Никифоров Арсен Германович

Группа: К3343d

Проверил:

Говоров Антон Игоревич

Санкт-Петербург

2023 г.

Объектно-ориентированное программирование (ООП) — это методология программирования, базирующаяся на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

ООП основывается на четырех основных принципах:

1. **Инкапсуляция**: Сокрытие деталей реализации и представление пользователям только функционального интерфейса.
2. **Наследование**: Создание новых классов на основе уже существующих.
3. **Полиморфизм**: Различное использование одного и того же метода в разных классах.
4. **Абстракция**: Выделение значимых характеристик объекта и игнорирование незначимых.

## Инкапсуляция

Инкапсуляция — это механизм, который объединяет данные и методы, работающие с этими данными, в одну целостную структуру — объект, и скрывает детали реализации от пользователя.

class Rectangle {

private:

double width;

double height;

public:

void setWidth(double w) {

width = w;

}

void setHeight(double h) {

height = h;

}

double getArea() {

return width \* height;

}

};

В этом примере width и height скрыты от пользователя и доступны только через методы класса.

## Наследование

Наследование позволяет создавать новые классы на основе уже существующих с перениманием их свойств и методов. Это способствует повторному использованию кода.

class Shape {

protected:

double color;

public:

void setColor(double c) {

color = c;

}

double getColor() {

return color;

}

};

class Rectangle : public Shape {

// Реализация класса Rectangle

};

В этом примере класс Rectangle наследуется от класса Shape и имеет доступ ко всем его открытым и защищенным членам.

## Полиморфизм

Полиморфизм означает наличие нескольких форм одного и того же метода в разных классах.

class Shape {

public:

virtual void draw() = 0; // чистая виртуальная функция

};

class Rectangle : public Shape {

public:

void draw() override {

// Рисуем прямоугольник

}

};

class Circle : public Shape {

public:

void draw() override {

// Рисуем круг

}

};

В этом примере, и Rectangle, и Circle являются Shape, но у них различная реализация метода draw().

## Абстракция

Абстракция — это процесс выделения общих свойств объектов и игнорирования их незначимых свойств. Это помогает снизить сложность и упростить представление системы.

class Shape {

public:

virtual double getArea() = 0;

};

class Rectangle : public Shape {

// Реализация класса Rectangle

};

class Circle : public Shape {

// Реализация класса Circle

};

В этом примере Shape является абстрактным классом, который определяет общий интерфейс для всех форм.

В заключение, принципы ООП играют важную роль в веб-разработке, так как они помогают упростить код и дела# Принципы объектно-ориентированного программирования (ООП) с примерами на C++

Объектно-ориентированное программирование (ООП) — это методология программирования, базирующаяся на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

ООП основывается на четырех основных принципах:

1. **Инкапсуляция**: Сокрытие деталей реализации и представление пользователям только функционального интерфейса.
2. **Наследование**: Создание новых классов на основе уже существующих.
3. **Полиморфизм**: Различное использование одного и того же метода в разных классах.
4. **Абстракция**: Выделение значимых характеристик объекта и игнорирование незначимых.

## Инкапсуляция

Инкапсуляция — это механизм, который объединяет данные и методы, работающие с этими данными, в одну целостную структуру — объект, и скрывает детали реализации от пользователя.

class Rectangle {

private:

double width;

double height;

public:

void setWidth(double w) {

width = w;

}

void setHeight(double h) {

height = h;

}

double getArea() {

return width \* height;

}

};

В этом примере width и height скрыты от пользователя и доступны только через методы класса.

## Наследование

Наследование позволяет создавать новые классы на основе уже существующих с перениманием их свойств и методов. Это способствует повторному использованию кода.

class Shape {

protected:

double color;

public:

void setColor(double c) {

color = c;

}

double getColor() {

return color;

}

};

class Rectangle : public Shape {

// Реализация класса Rectangle

};

В этом примере класс Rectangle наследуется от класса Shape и имеет доступ ко всем его открытым и защищенным членам.

## Полиморфизм

Полиморфизм означает наличие нескольких форм одного и того же метода в разных классах.

class Shape {

public:

virtual void draw() = 0; // чистая виртуальная функция

};

class Rectangle : public Shape {

public:

void draw() override {

// Рисуем прямоугольник

}

};

class Circle : public Shape {

public:

void draw() override {

// Рисуем круг

}

};

В этом примере, и Rectangle, и Circle являются Shape, но у них различная реализация метода draw().

## Абстракция

Абстракция — это процесс выделения общих свойств объектов и игнорирования их незначимых свойств. Это помогает снизить сложность и упростить представление системы.

class Shape {

public:

virtual double getArea() = 0;

};

class Rectangle : public Shape {

// Реализация класса Rectangle

};

class Circle : public Shape {

// Реализация класса Circle

};

В этом примере Shape является абстрактным классом, который определяет общий интерфейс для всех форм.

**Заключение**