ГУАП

КАФЕДРА № 42

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| канд. техн. наук, доцент |  |  |  | О. И. Красильникова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №3 |
| РАБОТЫ С ПРОТОТИПАМИ И КЛАССАМИ В JAVASCRIPT |
| по курсу: |
|  |
| ИТ-МОДУЛЬ "ОСНОВЫ FRONTEND-РАЗРАБОТКИ" |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 1245 |  | 31.05.2024 |  | Г. С. Куранов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2025

1. **Цель:**

Формирование практических навыков работы с прототипами и классами в JavaScript.

1. **Формулировка** заданий**:**

* Создайте класс "Фигура", который будет иметь методы для вычисления площади и периметра фигуры. Напишите классы "Прямоугольник" и "Круг", которые наследуются от класса "Фигура" и реализуют соответствующие методы.
* Создайте класс "Студент", который будет иметь защищенные свойства "имя", "возраст" и "средний балл". Напишите методы для изменения и получения значений этих свойств.
* Создайте класс "Калькулятор", который будет иметь методы для выполнения основных математических операций (сложение, вычитание, умножение, деление).

1. **Скриншоты, демонстрирующие работу web-приложения:**

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1 – Класс фигура с методами. Реализация

Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 – Вызов методов в классе студент

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3 – Некоторые примеры работы калькулятора

1. **Выводы**

В ходе лабораторной работы поставленная цель работы достигнута. В процессе выполнения задания были:

* Освоены основные принципы ООП в JavaScript (инкапсуляция, наследование, полиморфизм)
* Приобретены навыки работы с классами, конструкторами, методами и свойствами
* Поняты различия между абстрактными классами и конкретными реализациями
* Получен практический опыт создания иерархий классов и работы с защищенными свойствами

Приложение А. **Shape.js, Student.js, Calculator.js**

**Shape.js**

// Базовый класс Фигура

class Shape {

constructor() {

if (new.target === Shape) {

throw new Error("Нельзя создать экземпляр абстрактного класса Shape");

}

}

// Абстрактные методы

getArea() {

throw new Error("Метод getArea должен быть реализован");

}

getPerimeter() {

throw new Error("Метод getPerimeter должен быть реализован");

}

}

// Класс Прямоугольник

class Rectangle extends Shape {

constructor(width, height) {

super();

this.width = width;

this.height = height;

}

getArea() {

return this.width \* this.height;

}

getPerimeter() {

return 2 \* (this.width + this.height);

}

}

// Класс Круг

class Circle extends Shape {

constructor(radius) {

super();

this.radius = radius;

}

getArea() {

return Math.PI \* this.radius \*\* 2;

}

getPerimeter() {

return 2 \* Math.PI \* this.radius;

}

}

// Пример использования

const rect = new Rectangle(5, 10);

console.log('Прямоугольник:');

console.log('Площадь:', rect.getArea()); // 50

console.log('Периметр:', rect.getPerimeter()); // 30

const circle = new Circle(7);

console.log('\nКруг:');

console.log('Площадь:', circle.getArea()); // ≈153.938

console.log('Периметр:', circle.getPerimeter()); // ≈43.982  
  
**Student.js**  
class Student {

constructor(name, age, averageGrade) {

this.\_name = name;

this.\_age = age;

this.\_averageGrade = averageGrade;

}

// Геттеры

getName() {

return this.\_name;

}

getAge() {

return this.\_age;

}

getAverageGrade() {

return this.\_averageGrade;

}

// Сеттеры

setName(name) {

if (typeof name === 'string' && name.trim() !== '') {

this.\_name = name;

} else {

console.error('Имя должно быть непустой строкой');

}

}

setAge(age) {

if (Number.isInteger(age) && age > 0) {

this.\_age = age;

} else {

console.error('Возраст должен быть положительным целым числом');

}

}

setAverageGrade(grade) {

if (typeof grade === 'number' && grade >= 0 && grade <= 100) {

this.\_averageGrade = grade;

} else {

console.error('Средний балл должен быть числом от 0 до 100');

}

}

}

// Пример использования

const student = new Student('Иван Иванов', 20, 85);

console.log('\nСтудент:');

console.log('Имя:', student.getName());

console.log('Возраст:', student.getAge());

console.log('Средний балл:', student.getAverageGrade());

student.setName('Петр Петров');

student.setAge(21);

student.setAverageGrade(90);

console.log('\nПосле изменений:');

console.log('Имя:', student.getName());

console.log('Возраст:', student.getAge());

console.log('Средний балл:', student.getAverageGrade());

**Calculator.js**  
class Calculator {

constructor() {

this.\_lastResult = null;

}

add(a, b) {

this.\_lastResult = a + b;

return this.\_lastResult;

}

subtract(a, b) {

this.\_lastResult = a - b;

return this.\_lastResult;

}

multiply(a, b) {

this.\_lastResult = a \* b;

return this.\_lastResult;

}

divide(a, b) {

if (b === 0) {

throw new Error('Деление на ноль невозможно');

}

this.\_lastResult = a / b;

return this.\_lastResult;

}

getLastResult() {

return this.\_lastResult;

}

clear() {

this.\_lastResult = null;

}

}

// Пример использования

const calc = new Calculator();

console.log('\nКалькулятор:');

console.log('5 + 3 =', calc.add(5, 3)); // 8

console.log('Последний результат:', calc.getLastResult()); // 8

console.log('5 - 3 =', calc.subtract(5, 3)); // 2

console.log('5 \* 3 =', calc.multiply(5, 3)); // 15

console.log('6 / 3 =', calc.divide(6, 3)); // 2

try {

console.log('6 / 0 =', calc.divide(6, 0));

} catch (e) {

console.error(e.message); // Деление на ноль невозможно

}