Технологии конструирования программного обеспечения

Отчет по экзаменационной работе № 01

Группа: 221-329 **Студент:** Минчаков Аркадий Сергеевич

Задание на экзаменационную работу:

Паттерн «Прототип». Приложение с визуальным интерфейсом «Кнопки». Реализовать хранилище прототипов, содержащее элементы управления Button с различными вариантами размеров и цветов. Количество прототипов не менее 3.

UML-диаграмма классов:

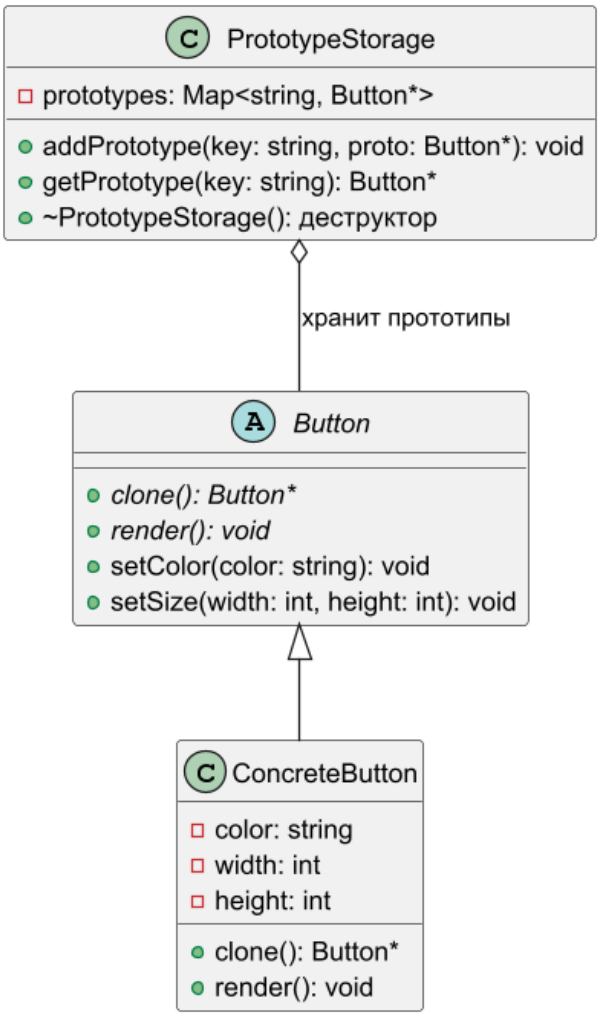

Figure 1: UML-диаграмма классов

Исходный код программы:

main.cpp

```
/*
Программа реализует паттерн "Прототип" для создания кнопок с различными стилями.
Автор: Минчаков Аркадий Сергеевич, группа 221-329

Идея реализации:
- Класс Button поддерживает клонирование через метод clone()
- Хранилище (PrototypeStorage) содержит прототипы кнопок
- Создание новых кнопок происходит через копирование прототипов
*/

#include <iostream>
#include <unordered_map>
#include <string>

using namespace std;

// Базовый класс кнопки с поддержкой клонирования
class Button {
public:
    virtual ~Button() {}

    // Метод для создания копии объекта
    virtual Button* clone() const = 0;

    // Метод отрисовки кнопки (условная реализация)
    virtual void render() const = 0;

    virtual void setColor(const string& color) { this->color = color; }
    virtual void setSize(int width, int height) {
        this->width = width;
        this->height = height;
    }

protected:
    string color = "gray";
    int width = 100;
    int height = 40;
};

// Конкретная реализация кнопки
class ConcreteButton : public Button {
public:
    // Создание копии объекта
    Button* clone() const override {
        ConcreteButton* copy = new ConcreteButton();
        copy->color = this->color;
        copy->width = this->width;
        copy->height = this->height;
        return copy;
    }

    void render() const override {
        cout << "Button: [Color: " << color
             << ", Size: " << width << "x" << height
             << "]" << endl;
    }
};

// Хранилище прототипов кнопок
class PrototypeStorage {
private:
```

```
    unordered_map<string, Button*> prototypes;

public:
    ~PrototypeStorage() {
        // Очистка памяти при удалении хранилища
        for (auto& pair : prototypes) {
            delete pair.second;
        }
    }

    // Добавление прототипа в хранилище
    void addPrototype(const string& key, Button* proto) {
        prototypes[key] = proto;
    }

    // Получение копии прототипа по ключу
    Button* getPrototype(const string& key) {
        if (prototypes.find(key) != prototypes.end()) {
            return prototypes[key]->clone();
        }
        return nullptr;
    }
};

int main() {
    // Создание хранилища
    PrototypeStorage storage;

    // Создание и настройка прототипов
    ConcreteButton* redProto = new ConcreteButton();
    redProto->setColor("red");
    redProto->setSize(120, 50);
    storage.addPrototype("red", redProto);

    ConcreteButton* blueProto = new ConcreteButton();
    blueProto->setColor("blue");
    storage.addPrototype("blue", blueProto);

    ConcreteButton* largeProto = new ConcreteButton();
    largeProto->setSize(200, 80);
    storage.addPrototype("large", largeProto);

    // Создание кнопок из прототипов
    Button* btn1 = storage.getPrototype("red");
    Button* btn2 = storage.getPrototype("blue");
    Button* btn3 = storage.getPrototype("large");

    // Демонстрация работы
    btn1->render(); // [Color: red, Size: 120x50]
    btn2->render(); // [Color: blue, Size: 100x40]
    btn3->render(); // [Color: gray, Size: 200x80]

    // Очистка памяти
    delete btn1;
    delete btn2;
    delete btn3;

    return 0;
}
```