МИНИСТЕРСТВО НАУКИ и высшего образования

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**Отчёт по лабораторной работе № 1**

**по учебной дисциплине «Прикладное программирование»**

**Тема: «Структура консольного приложения в C#»**

Выполнил(а) студент(ка)

специальности 09.02.03

Программирование в компьютерных

системах

IV курса группы 419/7

Косовский Семен

Андреевич

Преподаватель

Молькова Лолита Юрьевна

Санкт-Петербург,

2024

Цель работы:

1. Ознакомление с процессом классификации на примере моделирования классов предметной области. 2. Ознакомление с диаграммами классов и взаимодействия UML.
2. Задание:

Реализовать массив данных в соответствии с вариантом задания. • Определить метод установки свойств (при недопустимых аргументах функции возвращать «false» и выдавать текст ошибки на экран). • Написать демонстрационную программу, в которой показать работу с переменными. **Индивидуальное задание:**

Вариант 8.  
Пользователь вводит массив Прямоугольных треугольников  
Свойства: сторона и угол  
Операции:  
﻿﻿уменьшение/увеличение размера любой стороны (из свойств) на заданный процент;  
﻿﻿вычисление радиуса вписанной окружности;  
﻿﻿определение расстояния между центрами вписанной и описанной окружностей;  
О вычисление квадратного корня из площади;  
• определение значений сторон.  
Пользователь выбирает элемент массива и выполняемую операцию.

**Ход работы:**

На рисунке 1 изображена валидация ввода количество треугольников

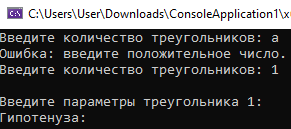


Рисунок 1 – Валидация ввода количество треугольников

На рисунке 2 изображена валидация ввода 2 свойств класса

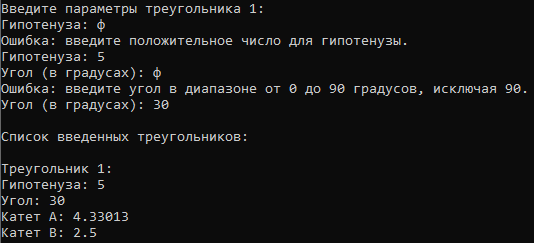


Рисунок 2 – валидация ввода 2 свойств класса

На рисунке 3 изображена валидация ввода выбора треугольника

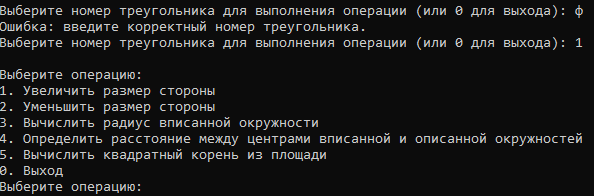


Рисунок 3 – валидация ввода выбора треугольника

На рисунке 4 изображена валидация ввода выбора метода

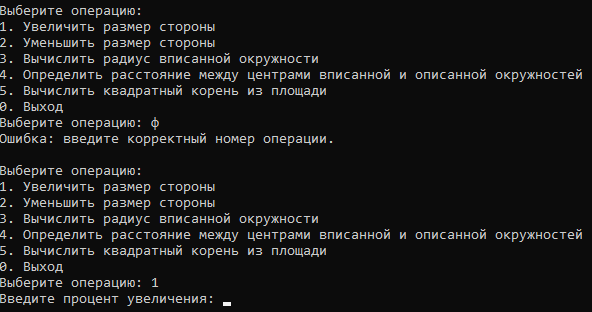


Рисунок 4 – валидация ввода выбора метода

На рисунке 5 изображена работа метода увеличения стороны

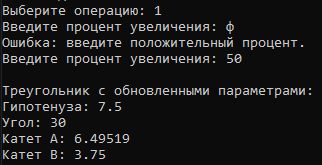


Рисунок 5 – работа метода увеличения стороны

На рисунке 6 изображена работа метода уменьшения стороны

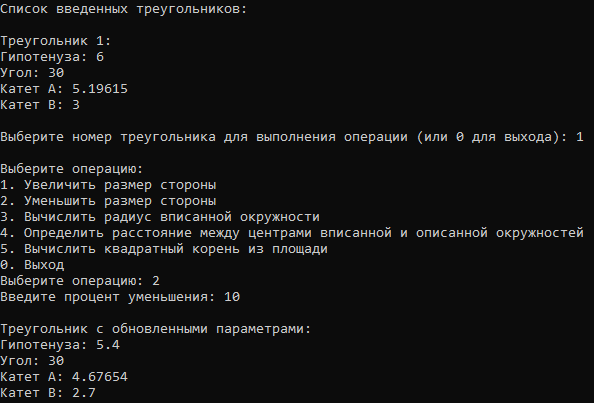


Рисунок 6 – работа метода уменьшения стороны

На рисунке 7 изображена работа метода расчета радиуса вписанной окружности

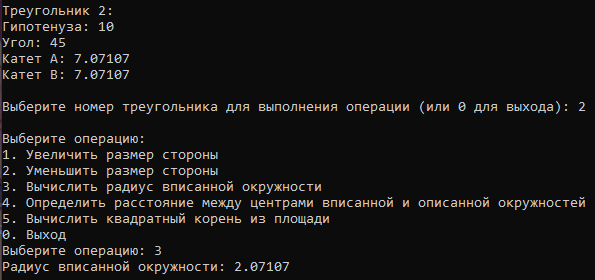


Рисунок 7 – работа метода расчета радиуса вписанной окружности

На рисунке 8 изображена работа метода расчета расстояния между центрами вписанной и описанной окружностей

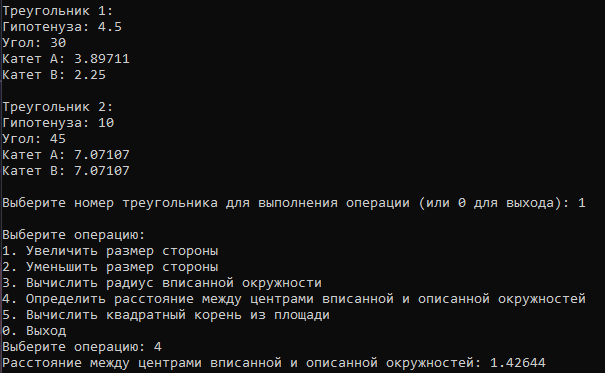


Рисунок 8 – работа метода расчета расстояния между центрами вписанной и описанной окружностей

На рисунке 9 изображена работа метода расчета квадратного корня из площади треугольника

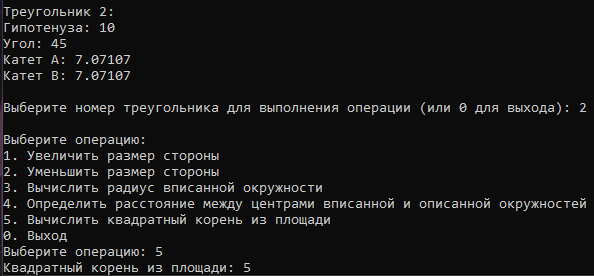


Рисунок 9 – работа метода расчета квадратного корня из площади треугольника

Код:

Заголовочный файл Triangle.h  
#ifndef TRIANGLE\_H

#define TRIANGLE\_H

#include <cmath>

#include <iostream>

using namespace std;

class Triangle {

public:

Triangle(double hypotenuse, double angle);

bool setSides(double hypotenuse, double angle);

void increaseSide(double percent);

void decreaseSide(double percent);

double calculateInradius() const;

double distanceBetweenInradiusAndCircumradius() const;

double calculateSquareRootOfArea() const;

void print() const;

private:

double hypotenuse;

double angle; // angle in degrees

double side\_a;

double side\_b;

void updateSides();

double toRadians(double degrees) const;

double calculateCircumradius() const;

double calculateArea() const;

};

#endif // TRIANGLE\_H

Файл реализации методов класса Triangle.cpp  
#include "Triangle.h"

#include <cmath> // Для функций cos, sin, sqrt и определения числа PI

#include <iostream>

using namespace std;

#ifndef M\_PI

#define M\_PI 3.14159265358979323846 // Определение PI, если оно не определено

#endif

Triangle::Triangle(double hypotenuse, double angle) : hypotenuse(hypotenuse), angle(angle) {

if (!setSides(hypotenuse, angle)) {

cerr << "Ошибка: неверные начальные параметры." << endl;

}

}

bool Triangle::setSides(double hypotenuse, double angle) {

if (hypotenuse <= 0 || angle <= 0 || angle >= 90) {

cout << "Ошибка: недопустимые значения гипотенузы или угла." << endl;

return false;

}

this->hypotenuse = hypotenuse;

this->angle = angle;

updateSides();

return true;

}

void Triangle::updateSides() {

double angleRad = toRadians(angle);

side\_a = hypotenuse \* cos(angleRad);

side\_b = hypotenuse \* sin(angleRad);

}

double Triangle::toRadians(double degrees) const {

return degrees \* M\_PI / 180.0;

}

double Triangle::calculateInradius() const {

return (side\_a + side\_b - hypotenuse) / 2.0;

}

double Triangle::calculateCircumradius() const {

return hypotenuse / 2.0;

}

double Triangle::distanceBetweenInradiusAndCircumradius() const {

return fabs(calculateCircumradius() - calculateInradius());

}

double Triangle::calculateArea() const {

return 0.5 \* side\_a \* side\_b;

}

double Triangle::calculateSquareRootOfArea() const {

return sqrt(calculateArea());

}

void Triangle::increaseSide(double percent) {

side\_a \*= (1 + percent / 100);

side\_b \*= (1 + percent / 100);

hypotenuse = sqrt(side\_a \* side\_a + side\_b \* side\_b);

}

void Triangle::decreaseSide(double percent) {

side\_a \*= (1 - percent / 100);

side\_b \*= (1 - percent / 100);

hypotenuse = sqrt(side\_a \* side\_a + side\_b \* side\_b);

}

void Triangle::print() const {

cout << "Гипотенуза: " << hypotenuse << endl;

cout << "Угол: " << angle << endl;

cout << "Катет A: " << side\_a << endl;

cout << "Катет B: " << side\_b << endl;

}  
3. Основной файл main.cpp  
#include <iostream>

#include <vector>

#include "Triangle.h"

using namespace std;

void printMenu() {

cout << "\nВыберите операцию:" << endl;

cout << "1. Увеличить размер стороны" << endl;

cout << "2. Уменьшить размер стороны" << endl;

cout << "3. Вычислить радиус вписанной окружности" << endl;

cout << "4. Определить расстояние между центрами вписанной и описанной окружностей" << endl;

cout << "5. Вычислить квадратный корень из площади" << endl;

cout << "0. Выход" << endl;

}

int main() {  
setlocale(LC\_ALL, "rus");

vector<Triangle> triangles;

int numTriangles;

cout << "Введите количество треугольников: ";

cin >> numTriangles;

while (cin.fail() || numTriangles <= 0) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите положительное число." << endl;

cout << "Введите количество треугольников: ";

cin >> numTriangles;

}

for (int i = 0; i < numTriangles; ++i) {

double hypotenuse, angle;

cout << "\nВведите параметры треугольника " << (i + 1) << ":\n";

cout << "Гипотенуза: ";

cin >> hypotenuse;

while (cin.fail() || hypotenuse <= 0) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите положительное число для гипотенузы." << endl;

cout << "Гипотенуза: ";

cin >> hypotenuse;

}

cout << "Угол (в градусах): ";

cin >> angle;

while (cin.fail() || angle <= 0 || angle >= 90) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите угол в диапазоне от 0 до 90 градусов, исключая 90." << endl;

cout << "Угол (в градусах): ";

cin >> angle;

}

Triangle triangle(hypotenuse, angle);

triangles.push\_back(triangle);

}

int choice;

while (true) {

cout << "\nСписок введенных треугольников:" << endl;

for (int i = 0; i < triangles.size(); ++i) {

cout << "\nТреугольник " << (i + 1) << ":" << endl;

triangles[i].print();

}

cout << "\nВыберите номер треугольника для выполнения операции (или 0 для выхода): ";

int index;

cin >> index;

while (cin.fail() || index < 0 || index > triangles.size()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите корректный номер треугольника." << endl;

cout << "Выберите номер треугольника для выполнения операции (или 0 для выхода): ";

cin >> index;

}

if (index == 0) break;

Triangle &selectedTriangle = triangles[index - 1];

printMenu();

cout << "Выберите операцию: ";

cin >> choice;

while (cin.fail() || choice < 0 || choice > 5) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите корректный номер операции." << endl;

printMenu();

cout << "Выберите операцию: ";

cin >> choice;

}

switch (choice) {

case 1: {

double percent;

cout << "Введите процент увеличения: ";

cin >> percent;

while (cin.fail() || percent < 0) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите положительный процент." << endl;

cout << "Введите процент увеличения: ";

cin >> percent;

}

selectedTriangle.increaseSide(percent);

break;

}

case 2: {

double percent;

cout << "Введите процент уменьшения: ";

cin >> percent;

while (cin.fail() || percent < 0) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите положительный процент." << endl;

cout << "Введите процент уменьшения: ";

cin >> percent;

}

selectedTriangle.decreaseSide(percent);

break;

}

case 3:

cout << "Радиус вписанной окружности: " << selectedTriangle.calculateInradius() << endl;

break;

case 4:

cout << "Расстояние между центрами вписанной и описанной окружностей: "

<< selectedTriangle.distanceBetweenInradiusAndCircumradius() << endl;

break;

case 5:

cout << "Квадратный корень из площади: " << selectedTriangle.calculateSquareRootOfArea() << endl;

break;

default:

cout << "Неверный выбор!" << endl;

break;

}

cout << "\nТреугольник с обновленными параметрами:" << endl;

selectedTriangle.print();

}

return 0;

}

UML диаграмма:

| **Triangle** |  |
| --- | --- |
| **Название метода** | **Описание** |
| Triangle(double hypotenuse, double angle) | Конструктор класса Triangle. |
| bool setSides(double hypotenuse, double angle) | Устанавливает стороны треугольника. |
| void increaseSide(double percent) | Увеличивает размер стороны на заданный процент. |
| void decreaseSide(double percent) | Уменьшает размер стороны на заданный процент. |
| double calculateInradius() const | Вычисляет радиус вписанной окружности. |
| double distanceBetweenInradiusAndCircumradius() const | Определяет расстояние между центрами вписанной и описанной окружностей. |
| double calculateSquareRootOfArea() const | Вычисляет квадратный корень из площади треугольника. |
| void print() const | Выводит информацию о треугольнике. |
| void updateSides() | Обновляет значения сторон треугольника на основе гипотенузы и угла. |
| double toRadians(double degrees) const | Преобразует угол из градусов в радианы. |
| double calculateCircumradius() const | Вычисляет радиус описанной окружности. |
| double calculateArea() const | Вычисляет площадь треугольника. |

2.2 Индивидуальное задание (50%)

Перечислите несколько однотипных объектов и конкретные значения их свойств.

Выделите классы из группы однотипных объектов.

Требования:  
Не менее 5 классов

Не менее 7 объектов каждого класса.

Не менее 3 свойств для каждого класса.

Не менее 3 операций для каждого класса.

Имена классов - существительные.

классов – существительные.

Предметная область

Вариант 8. Офисное оборудование.

**Ход работы:**

На рисунке 1 изображена вся работы программы

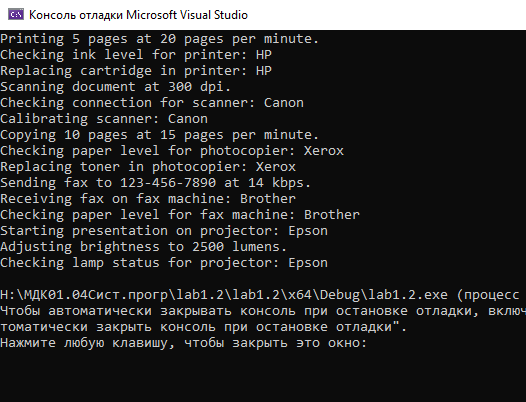


Рисунок 1 – Вся работы программы

Код:

1. Printer.h

cpp

Копировать код

#ifndef PRINTER\_H

#define PRINTER\_H

#include <iostream>

#include <string>

using namespace std;

class Printer {

public:

Printer(string brand, int printSpeed, bool colorPrinting);

void printDocument(int pages);

void checkInkLevel();

void replaceCartridge();

private:

string brand;

int printSpeed;

bool colorPrinting;

};

#endif // PRINTER\_H

2. Printer.cpp

cpp

Копировать код

#include "Printer.h"

Printer::Printer(string brand, int printSpeed, bool colorPrinting)

: brand(brand), printSpeed(printSpeed), colorPrinting(colorPrinting) {}

void Printer::printDocument(int pages) {

cout << "Printing " << pages << " pages at " << printSpeed << " pages per minute." << endl;

}

void Printer::checkInkLevel() {

cout << "Checking ink level for printer: " << brand << endl;

}

void Printer::replaceCartridge() {

cout << "Replacing cartridge in printer: " << brand << endl;

}

3. Scanner.h

cpp

Копировать код

#ifndef SCANNER\_H

#define SCANNER\_H

#include <iostream>

#include <string>

using namespace std;

class Scanner {

public:

Scanner(string brand, int resolution, bool wireless);

void scanDocument();

void checkConnection();

void calibrateScanner();

private:

string brand;

int resolution;

bool wireless;

};

#endif // SCANNER\_H

4. Scanner.cpp

cpp

Копировать код

#include "Scanner.h"

Scanner::Scanner(string brand, int resolution, bool wireless)

: brand(brand), resolution(resolution), wireless(wireless) {}

void Scanner::scanDocument() {

cout << "Scanning document at " << resolution << " dpi." << endl;

}

void Scanner::checkConnection() {

cout << "Checking connection for scanner: " << brand << endl;

}

void Scanner::calibrateScanner() {

cout << "Calibrating scanner: " << brand << endl;

}

5. Photocopier.h

cpp

Копировать код

#ifndef PHOTOCOPIER\_H

#define PHOTOCOPIER\_H

#include <iostream>

#include <string>

using namespace std;

class Photocopier {

public:

Photocopier(string brand, int copySpeed, bool duplexCopying);

void copyDocument(int pages);

void checkPaperLevel();

void replaceToner();

private:

string brand;

int copySpeed;

bool duplexCopying;

};

#endif // PHOTOCOPIER\_H

6. Photocopier.cpp

cpp

Копировать код

#include "Photocopier.h"

Photocopier::Photocopier(string brand, int copySpeed, bool duplexCopying)

: brand(brand), copySpeed(copySpeed), duplexCopying(duplexCopying) {}

void Photocopier::copyDocument(int pages) {

cout << "Copying " << pages << " pages at " << copySpeed << " pages per minute." << endl;

}

void Photocopier::checkPaperLevel() {

cout << "Checking paper level for photocopier: " << brand << endl;

}

void Photocopier::replaceToner() {

cout << "Replacing toner in photocopier: " << brand << endl;

}  
7. FaxMachine.h

cpp

Копировать код

#ifndef FAXMACHINE\_H

#define FAXMACHINE\_H

#include <iostream>

#include <string>

using namespace std;

class FaxMachine {

public:

FaxMachine(string brand, int transmissionSpeed, bool supportsVoicemail);

void sendFax(string number);

void receiveFax();

void checkPaperLevel();

private:

string brand;

int transmissionSpeed;

bool supportsVoicemail;

};

#endif // FAXMACHINE\_H  
FaxMachine.cpp

cpp

Копировать код

#include "FaxMachine.h"

FaxMachine::FaxMachine(string brand, int transmissionSpeed, bool supportsVoicemail)

: brand(brand), transmissionSpeed(transmissionSpeed), supportsVoicemail(supportsVoicemail) {}

void FaxMachine::sendFax(string number) {

cout << "Sending fax to " << number << " at " << transmissionSpeed << " kbps." << endl;

}

void FaxMachine::receiveFax() {

cout << "Receiving fax on fax machine: " << brand << endl;

}

void FaxMachine::checkPaperLevel() {

cout << "Checking paper level for fax machine: " << brand << endl;

}

9. Projector.h

cpp

Копировать код

#ifndef PROJECTOR\_H

#define PROJECTOR\_H

#include <iostream>

#include <string>

using namespace std;

class Projector {

public:

Projector(string brand, int brightness, bool supportsHD);

void startPresentation();

void adjustBrightness(int newBrightness);

void checkLampStatus();

private:

string brand;

int brightness;

bool supportsHD;

};

#endif // PROJECTOR\_H  
10. Projector.cpp

cpp

Копировать код

#include "Projector.h"

Projector::Projector(string brand, int brightness, bool supportsHD)

: brand(brand), brightness(brightness), supportsHD(supportsHD) {}

void Projector::startPresentation() {

cout << "Starting presentation on projector: " << brand << endl;

}

void Projector::adjustBrightness(int newBrightness) {

brightness = newBrightness;

cout << "Adjusting brightness to " << brightness << " lumens." << endl;

}

void Projector::checkLampStatus() {

cout << "Checking lamp status for projector: " << brand << endl;

}

main.cpp

cpp

Копировать код

#include <iostream>

#include "Printer.h"

#include "Scanner.h"

#include "Photocopier.h"

#include "FaxMachine.h"

#include "Projector.h"

using namespace std;

int main() {

// Создание 7 объектов каждого класса офисного оборудования

Printer printers[7] = {

Printer("HP", 20, true),

Printer("Canon", 15, false),

Printer("Epson", 25, true),

Printer("Brother", 18, true),

Printer("Xerox", 30, false),

Printer("Lexmark", 22, true),

Printer("Ricoh", 24, false)

};

Scanner scanners[7] = {

Scanner("Canon", 300, false),

Scanner("Epson", 600, true),

Scanner("HP", 150, false),

Scanner("Brother", 1200, true),

Scanner("Xerox", 600, false),

Scanner("Lexmark", 300, true),

Scanner("Ricoh", 1200, false)

};

Photocopier photocopiers[7] = {

Photocopier("Xerox", 15, true),

Photocopier("Canon", 10, false),

Photocopier("HP", 12, true),

Photocopier("Ricoh", 20, false),

Photocopier("Brother", 25, true),

Photocopier("Lexmark", 18, false),

Photocopier("Epson", 16, true)

};

FaxMachine faxMachines[7] = {

FaxMachine("Brother", 14, true),

FaxMachine("Canon", 12, false),

FaxMachine("HP", 10, true),

FaxMachine("Xerox", 15, true),

FaxMachine("Ricoh", 20, false),

FaxMachine("Lexmark", 18, true),

FaxMachine("Epson", 16, false)

};

Projector projectors[7] = {

Projector("Epson", 3000, true),

Projector("Benq", 3500, false),

Projector("Canon", 4000, true),

Projector("Sony", 4500, true),

Projector("Acer", 3200, false),

Projector("Dell", 2900, true),

Projector("ViewSonic", 3100, false)

};

// Вызов всех трех методов для одного объекта каждого класса

printers[0].printDocument(5);

printers[0].checkInkLevel();

printers[0].replaceCartridge();

scanners[0].scanDocument();

scanners[0].checkConnection();

scanners[0].calibrateScanner();

photocopiers[0].copyDocument(10);

photocopiers[0].checkPaperLevel();

photocopiers[0].replaceToner();

faxMachines[0].sendFax("123-456-7890");

faxMachines[0].receiveFax();

faxMachines[0].checkPaperLevel();

projectors[0].startPresentation();

projectors[0].adjustBrightness(2500);

projectors[0].checkLampStatus();

return 0;

}

UML диаграммы:

**Printer**

| **Название метода** | **Описание** |
| --- | --- |
| Printer(string brand, int printSpeed, bool colorPrinting) | Конструктор класса Printer. |
| void printDocument(int pages) | Печатает указанные страницы. |
| void checkInkLevel() | Проверяет уровень чернил в принтере. |
| void replaceCartridge() | Заменяет картридж в принтере. |

**Scanner**

| **Название метода** | **Описание** |
| --- | --- |
| Scanner(string brand, int resolution, bool wireless) | Конструктор класса Scanner. |
| void scanDocument() | Сканирует документ с заданным разрешением. |
| void checkConnection() | Проверяет подключение сканера. |
| void calibrateScanner() | Калибрует сканер. |

**Photocopier**

| **Название метода** | **Описание** |
| --- | --- |
| Photocopier(string brand, int copySpeed, bool duplexCopying) | Конструктор класса Photocopier. |
| void copyDocument(int pages) | Копирует указанные страницы. |
| void checkPaperLevel() | Проверяет уровень бумаги в копировальном аппарате. |
| void replaceToner() | Заменяет тонер в копировальном аппарате. |

**FaxMachine**

| **Название метода** | **Описание** |
| --- | --- |
| FaxMachine(string brand, int transmissionSpeed, bool supportsVoicemail) | Конструктор класса FaxMachine. |
| void sendFax(string number) | Отправляет факс на указанный номер. |
| void receiveFax() | Получает факс. |
| void checkPaperLevel() | Проверяет уровень бумаги в факсимильном аппарате. |

**Projector**

| **Название метода** | **Описание** |
| --- | --- |
| Projector(string brand, int brightness, bool supportsHD) | Конструктор класса Projector. |
| void startPresentation() | Начинает презентацию на проекторе. |
| void adjustBrightness(int newBrightness) | Изменяет яркость проектора. |
| void checkLampStatus() | Проверяет статус лампы проектора. |