Система классов «Библиотека»

Описание классов:

1. Person (абстрактный класс):
   * Общие характеристики для всех людей: имя, фамилия и адрес.
   * Абстрактный метод display() для отображения информации.
2. Employee (класс): Наследуется от Person.
   * Представляет сотрудника библиотеки. Дополнительное поле: должность.
   * Реализация метода display() для отображения информации о сотруднике.
3. Librarian (класс): Наследуется от Employee.
   * Представляет библиотекаря. Дополнительное поле: секция библиотеки.
   * Реализация метода display() для отображения информации о библиотекаре.
4. Client (класс): Наследуется от Person.
   * Представляет клиента библиотеки. Дополнительное поле: id клиента.
   * Реализация метода display() для отображения информации о клиенте.
5. Book (класс):
   * Общие характеристики для книг: название, автор.
   * Реализация метода display() для отображения информации о книге.
6. Magazine (класс): Наследуется от Book.
   * Представляет журнал. Дополнительное поле: номер выпуска.
   * Реализация метода display() для отображения информации о журнале.
7. Audiobook (класс): Наследуется от Book.
   * Представляет аудиокнигу. Дополнительное поле: длительность.
   * Реализация метода display() для отображения информации о аудиокниге.
8. Podcast (класс): Наследуется от Magazine и Audiobook.
   * Представляет подкаст.
   * Реализация метода display() для отображения информации о подкасте.
9. Address (класс): Используется в Person.
   * Представляет адрес.
10. Library (класс):
    * Представляет библиотеку. Хранит список книг, сотрудников, клиентов.
    * Добавление методов для работы со списками и поиском книги.

**Main.cpp**

#include <iostream>

#include "librarian.h"

#include "client.h"

#include "podcast.h"

#include "library.h"

using namespace std;

int main() {

Library lib(10, 5, 5);

lib.addBook(new Book("1984", "George Orwell"));

lib.addBook(new Magazine("National Geographic", "Various Authors", 202));

lib.addBook(new Audiobook("The Great Gatsby", "F. Scott Fitzgerald", 300));

lib.addBook(new Podcast("Science Talk", "Various Hosts", 15, 60));

lib.addEmployee(new Librarian("Alice", "Smith", "Khalturinsky 52", "Head Librarian", "West"));

lib.addClient(new Client("Bob", "Johnson", "Mechnikova 123", 101));

lib.displayBooks();

cout << "----------------------\n";

lib.displayClients();

cout << "----------------------\n";

lib.displayEmployees();

cout << "----------------------\n";

lib.findBookByTitle("1984");

return 0;

}

**Address.h**

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include "anyCorrect.h"

using namespace std;

using namespace anyCorrect;

class Address{

private:

string \_address;

public:

Address(string address);

~Address();

string getAddress();

void setAddress(string address);

void display();

};

**Address.cpp**

#include "address.h"

using namespace std;

Address::Address(string address)

: \_address(anyCorrect::correctPrint(address)) {}

Address::~Address() {};

string Address::getAddress() {

return \_address;

}

void Address::setAddress(string address) {

this->\_address = address;

}

void Address::display() {

cout << "Адрес: " << \_address << endl;

}

**Person.h**

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include "address.h"

#include "anyCorrect.h"

using namespace std;

using namespace anyCorrect;

class Person : public Address{

private:

string \_firstName;

string \_lastName;

public:

Person(string firstName, string lastName, string address);

~Person();

string getFirstName();

void setFirstName(string firstName);

string getLastName();

void setLastName(string lastName);

virtual void display() = 0;

};

**Person.cpp**

#include "person.h"

using namespace std;

Person::Person(string firstName, string lastName, string address)

: \_firstName(anyCorrect::correctAlpha(firstName)), \_lastName(anyCorrect::correctAlpha(lastName)), Address(address) {}

Person::~Person() {};

string Person::getFirstName() {

return \_firstName;

}

string Person::getLastName() {

return \_lastName;

}

void Person::setFirstName(string firstName) {

this->\_firstName = anyCorrect::correctAlpha(firstName);

}

void Person::setLastName(string lastName) {

this->\_lastName = anyCorrect::correctAlpha(lastName);

}

void Person::display() {

std::cout << "Имя: " << \_firstName << " " << \_lastName << "\n";

Address::display();

}

**Employee.h**

#pragma once

#include "person.h"

using namespace std;

using namespace anyCorrect;

class Employee : public Person {

private:

string \_position;

public:

Employee(string firstName, string lastName, string address, string position);

~Employee();

string getPosition();

void setPosition(string position);

void display();

};

**Employee.cpp**

#include "employee.h"

using namespace std;

Employee::Employee(string firstName, string lastName, string address, string position)

: Person(firstName, lastName, address), \_position(anyCorrect::correctAlpha(position)) {}

Employee::~Employee() {};

string Employee::getPosition() {

return \_position;

}

void Employee::setPosition(string position) {

this->\_position = \_position;

}

void Employee::display() {

Person::display();

cout << "Должность: " << \_position << "\n";

}

**Librarian.h**

#pragma once

#include "employee.h"

using namespace std;

using namespace anyCorrect;

class Librarian : public Employee {

private:

string \_librarySection;

public:

Librarian(string firstName, string lastName, string address, string position, string librarySection);

~Librarian();

string getLibrarySection();

void setLibrarySection(string librarySection);

void display();

};

**Librarian.cpp**

#include "librarian.h"

using namespace std;

Librarian::Librarian(string firstName, string lastName, string address, string position, string librarySection)

: Employee(firstName, lastName, address, position), \_librarySection(anyCorrect::correctAlpha(librarySection)) {}

Librarian::~Librarian() {};

string Librarian::getLibrarySection() {

return \_librarySection;

}

void Librarian::setLibrarySection(string librarySection) {

this->\_librarySection = librarySection;

}

void Librarian::display() {

Employee::display();

cout << "Секция библиотеки: " << \_librarySection << "\n";

}

**Client.h**

#pragma once

#include "person.h"

using namespace std;

using namespace anyCorrect;

class Client : public Person {

private:

int \_clientId;

public:

Client(string firstName, string lastName, string address, int clientId);

~Client();

int getClientId();

void setClientId(int clientId);

void display();

};

**Client.cpp**

#include "client.h"

using namespace std;

Client::Client(string firstName, string lastName, string address, int clientId)

: Person(firstName, lastName, address), \_clientId(anyCorrect::correctCount(clientId)) {}

Client::~Client() {};

int Client::getClientId() {

return \_clientId;

}

void Client::setClientId(int clientId) {

this->\_clientId = clientId;

}

void Client::display() {

Person::display();

cout << "Id клиента: " << \_clientId << "\n";

}

**Book.h**

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include "anyCorrect.h"

using namespace std;

using namespace anyCorrect;

class Book {

private:

string \_title;

string \_author;

public:

Book();

Book(string title, string author);

~Book();

string getTitle();

void setTitle(string title);

string getAuthor();

void setAuthor(string author);

virtual void display();

};

**Book.cpp**

#include "book.h"

using namespace std;

Book::Book(){

this->\_title = "undefined";

this->\_author = "undefined";

}

Book::Book(string title, string author)

: \_title(anyCorrect::correctPrint(title)), \_author(anyCorrect::correctAlpha(author)) {}

Book::~Book() {}

string Book::getTitle() {

return \_title;

}

string Book::getAuthor() {

return \_author;

}

void Book::setTitle(string title) {

this->\_title = anyCorrect::correctAlpha(title);

}

void Book::setAuthor(string author) {

this->\_author = anyCorrect::correctAlpha(author);

}

void Book::display() {

cout << "Название: " << \_title << "\n"

<< "Автор: " << \_author << "\n";

}

**Magazine.h**

#pragma once

#include "book.h"

#include <string>

using namespace std;

using namespace anyCorrect;

class Magazine : public virtual Book {

private:

int \_issueNumber;

public:

Magazine();

Magazine(string title, string author, int issueNumber);

Magazine(int issueNumber);

~Magazine();

int getIssueNumber();

void setIssueNumber(int issueNumber);

void display();

};

**Magazine.cpp**

#include "magazine.h"

using namespace std;

Magazine::Magazine(){

setTitle("undefined");

setAuthor("undefined");

this->\_issueNumber = 0;

}

Magazine::Magazine(string title, string author, int issueNumber)

: Book(title, author), \_issueNumber(anyCorrect::correctCount(issueNumber)) {}

Magazine::Magazine(int issueNumber)

: \_issueNumber(anyCorrect::correctCount(issueNumber)) {}

Magazine::~Magazine() {}

int Magazine::getIssueNumber() {

return \_issueNumber;

}

void Magazine::setIssueNumber(int issueNumber) {

this->\_issueNumber = anyCorrect::correctCount(issueNumber);

}

void Magazine::display() {

Book::display();

cout << "Номер выпуска: " << \_issueNumber << "\n";

}

**Audiobook.h**

#pragma once

#include "book.h"

#include <string>

using namespace std;

using namespace anyCorrect;

class Audiobook : public virtual Book {

private:

int \_duration;

public:

Audiobook();

Audiobook(string title, string author, int duration);

Audiobook(int duration);

~Audiobook();

int getDuration();

void setDuration(int duration);

void display();

};

**Audiobook.cpp**

#include "audiobook.h"

using namespace std;

Audiobook::Audiobook(){

setTitle("undefined");

setAuthor("undefined");

this->\_duration = 0;

}

Audiobook::Audiobook(string title, string author, int duration)

: Book(title, author), \_duration(anyCorrect::correctCount(duration)) {}

Audiobook::Audiobook(int duration)

: \_duration(anyCorrect::correctCount(duration)) {}

Audiobook::~Audiobook() {}

int Audiobook::getDuration() {

return \_duration;

}

void Audiobook::setDuration(int duration) {

this->\_duration = anyCorrect::correctCount(duration);

}

void Audiobook::display() {

Book::display();

cout << "Длительность: " << \_duration << " мин\n";

}

**Podcast.h**

#pragma once

#include "magazine.h"

#include "audiobook.h"

#include <string>

using namespace std;

using namespace anyCorrect;

class Podcast : public Magazine, public Audiobook {

public:

Podcast(string title, string author, int issueNumber, int duration);

~Podcast();

void display();

};

**Podcast.cpp**

#include "podcast.h"

using namespace std;

Podcast::Podcast(string title, string author, int issueNumber, int duration)

: Book(title, author), Magazine(issueNumber), Audiobook(duration) {}

void Podcast::display() {

Magazine::display();

cout << "Длительность: " << getDuration() << " мин\n";

}

**Library.h**

#pragma once

#include "book.h"

#include "employee.h"

#include "client.h"

#include <string>

using namespace std;

class Library {

private:

Book\*\* books;

int bookCount;

int bookCapacity;

Employee\*\* employees;

int employeeCount;

int employeeCapacity;

Client\*\* clients;

int clientCount;

int clientCapacity;

public:

Library(int bookCapacity, int employeeCapacity, int clientCapacity);

~Library();

void addBook(Book\* book);

void addEmployee(Employee\* employee);

void addClient(Client\* client);

void displayBooks();

void displayEmployees();

void displayClients();

void findBookByTitle(string title);

};

**Library.cpp**

#include "library.h"

#include <iostream>

using namespace std;

Library::Library(int bookCapacity, int employeeCapacity, int clientCapacity)

: bookCapacity(anyCorrect::correctCount(bookCapacity)), employeeCapacity(anyCorrect::correctCount(employeeCapacity)), clientCapacity(anyCorrect::correctCount(clientCapacity))

{

books = new Book\*[bookCapacity];

bookCount = 0;

employees = new Employee\*[employeeCapacity];

employeeCount = 0;

clients = new Client\*[clientCapacity];

clientCount = 0;

}

Library::~Library() {

for (int i = 0; i < bookCount; ++i)

delete books[i];

delete[] books;

for (int i = 0; i < employeeCount; ++i)

delete employees[i];

delete[] employees;

for (int i = 0; i < clientCount; ++i)

delete clients[i];

delete[] clients;

}

void Library::addBook(Book\* book) {

if (bookCount < bookCapacity) {

books[bookCount++] = book;

} else {

cout << "Достигнуто максимально возможное количество книг!" << endl;

}

}

void Library::addEmployee(Employee\* employee) {

if (employeeCount < employeeCapacity) {

employees[employeeCount++] = employee;

} else {

cout << "Достигнуто максимально возможное количество сотрудников!" << endl;

}

}

void Library::addClient(Client\* client) {

if (clientCount < clientCapacity) {

clients[clientCount++] = client;

} else {

cout << "Достигнуто максимально возможное количество клиентов!" << endl;

}

}

void Library::displayBooks() {

cout << "Список книг в библиотеке:" << endl;

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

cout << "----------------------\n";

books[i]->display();

}

}

void Library::displayEmployees() {

cout << "Список сотрудников библиотеки:" << endl;

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

cout << "----------------------\n";

employees[i]->display();

}

}

void Library::displayClients() {

cout << "Список клиентов библиотеки:" << endl;

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

cout << "----------------------\n";

clients[i]->display();

}

}

void Library::findBookByTitle(string title) {

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

if (books[i]->getTitle() == title) {

cout << "Подходящая запросу книга найдена!" << endl;

books[i]->display();

return;

}

}

cout << "Книги с таким названием нет в каталоге!" << endl;

}

**AnyCorrect.h**

#pragma once

#include <string>

namespace anyCorrect

{

std::string correctAlpha(std::string str);

int correctCount (int count);

std::string correctPrint(std::string str);

}

**AnyCorrect.cpp**

#include "anyCorrect.h"

#include <iostream>

std::string anyCorrect::correctAlpha(std::string str)

{

setlocale(LC\_ALL, "");

for (int i = 0; i < str.length(); ++i)

{

if (!isalpha(str.at(i)) and !ispunct(str.at(i)) and str.at(i) != ' ')

{

std::cout << "Некорректный ввод данных в поле\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

return "undefined";

}

}

return (str == "undefined") ? "undefined" : str;

}

int anyCorrect::correctCount(int count)

{

setlocale(LC\_ALL, "");

if (count >= 0) return count;

else

{

std::cout << "Некорректный ввод данных в поле\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

return 0;

}

}

std::string anyCorrect::correctPrint(std::string str)

{

setlocale(LC\_ALL, "");

for (int i = 0; i < str.length(); ++i)

{

if (!isprint(str.at(i)))

{

std::cout << "Некорректный ввод данных в поле\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

return "undefined";

}

}

return (str == "undefined") ? "undefined" : str;

}

Система классов «Библиотека»

Описание классов:

1. Person (абстрактный класс):
   * Общие характеристики для всех людей: имя, фамилия и адрес.
   * Абстрактный метод display() для отображения информации.
2. Employee (класс): Наследуется от Person.
   * Представляет сотрудника библиотеки. Дополнительное поле: должность.
   * Реализация метода display() для отображения информации о сотруднике.
3. Librarian (класс): Наследуется от Employee.
   * Представляет библиотекаря. Дополнительное поле: секция библиотеки.
   * Реализация метода display() для отображения информации о библиотекаре.
4. Client (класс): Наследуется от Person.
   * Представляет клиента библиотеки. Дополнительное поле: id клиента.
   * Реализация метода display() для отображения информации о клиенте.
5. Book (класс):
   * Общие характеристики для книг: название, автор.
   * Реализация метода display() для отображения информации о книге.
6. Magazine (класс): Наследуется от Book.
   * Представляет журнал. Дополнительное поле: номер выпуска.
   * Реализация метода display() для отображения информации о журнале.
7. Audiobook (класс): Наследуется от Book.
   * Представляет аудиокнигу. Дополнительное поле: длительность.
   * Реализация метода display() для отображения информации о аудиокниге.
8. Podcast (класс): Наследуется от Magazine и Audiobook.
   * Представляет подкаст.
   * Реализация метода display() для отображения информации о подкасте.
9. Address (класс): Используется в Person.
   * Представляет адрес.
10. Library (класс):
    * Представляет библиотеку. Хранит список книг, сотрудников, клиентов.
    * Добавление методов для работы со списками и поиском книги.