Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Информатика»

Тема: Программа, управляемая событиями

Семестр 2

Выполнил работу

Студент группы ИВТ-22-2б

Ищенко Дарья Олеговна

Проверил

Доцент кафедры ИТАС

Полякова О. A.

Г. Пермь-2023

**Постановка задачи**

1. Определить иерархию пользовательских классов (см. лабораторную работу №5). Во главе иерархии должен стоять абстрактный класс с чисто виртуальными методами для ввода и вывода информации об атрибутах объектов.

2. Реализовать конструкторы, деструктор, операцию присваивания, селекторы и модификаторы.

3. Определить класс-группу на основе структуры, указанной в варианте.

4. Для группы реализовать конструкторы, деструктор, методы для добавления и удаления элементов в группу, метод для просмотра группы, перегрузить операцию для получения информации о размере группы.

5. Определить класс Диалог – наследника группы, в котором реализовать методы для обработки событий.

6. Добавить методы для обработки событий группой и объектами пользовательских классов.

7. Написать тестирующую программу.

8. Нарисовать диаграмму классов и диаграмму объектов.

**Описание класса**

Базовый класс:

ПЕЧАТНОЕ\_ИЗДАНИЕ(PRINT)

Название– string

Автор – string

Производный класс

ЖУРНАЛ (MAGAZIN)

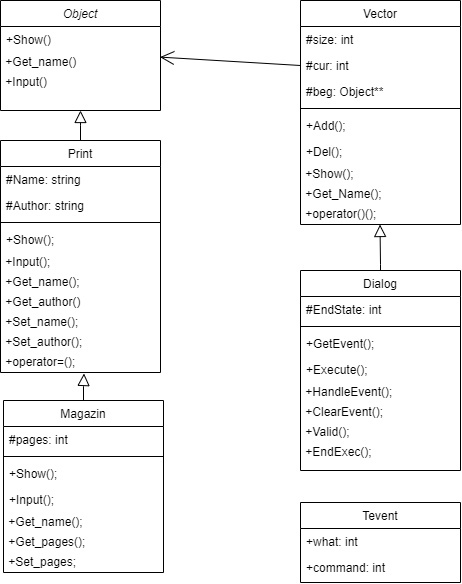
Количество страниц - int

Группа – Дерево (Tree).

Команды:

1. Создать группу (формат команды: m количество элементов группы).
2. Добавить элемент в группу (формат команды: +)
3. Удалить элемент из группы (формат команды -)
4. Вывести информацию об элементах группы (формат команды: s)
5. Вывести информацию о названиях всех элементов группы (формат команды : z)
6. Конец работы (формат команды: q)

**UML-диаграмма**

****

**Контрольные вопросы**

1. Что такое класс-группа? Привести примеры таких классов.

Группа - это объект, в который включены другие объекты. Объекты, входящие в группу, называются элементами группы. Элементы группы, в свою очередь, могут быть другой группой. В отличие от контейнера мы понимаем группу как класс, который не только хранит объекты других классов, но и обладает собственными свойствами, не вытекающими из

свойств его элементов.

*Пример*

class Faculty {

string name;

list<Department\*> \_deps; //факультет содержит список кафедр

}

class Department {

string name;

}

2. Привести пример описания класса-группы Список (List).

class List {

int size; //размер списка

elem\* head; //указатель на голову

};

3. Привести пример конструктора (с параметром, без параметров, копирования) для класса-группы Список.

List::List() { size = 0; head = nullptr; } //без параметров

List::List(int size, elem\* head) {this->size = size; this->head = head;} // с параметрами

List::List(const List& l) {head = l.head; size = l.size;} //копирование

4. Привести пример деструктора для класса-группы Список.

List::~List() {

while (size!=0) { //так как список - динамическая структура

elem del = this->head; //необходимо освобождать память

this->head = head->next;

delete del;

--size; }

5. Привести пример метода для просмотра элементов для класса-группы Список.

void List::print() {

for (int i=0; i < this->size; ++i) {

elem\* cur = this->head;

cout << cur << “ “;

cur = cur->next; }

}

6. Какой вид иерархии дает группа?

Группа дает второй вид иерархии (первый вид - иерархия классов, построенная на основе наследования) - иерархию объектов (иерархию типа целое/часть), построенную на основе агрегации.

7. Почему во главе иерархии классов, содержащихся в группе объектов должен находиться абстрактный класс?

Абстрактный класс не предполагает создание его экземпляров. Такой класс лишь задает интерфейс, который должен быть реализован в производных классах. Так как классы объединяются в группу по общим характеристикам (методам), стоит создать абстрактный класс с общими характеристиками, чтобы связать объекты в группе.

8. Что такое событие? Для чего используются события?

События - это пакеты информации, которыми обмениваются объекты и которые создаются объектно-ориентированной средой в ответ на те или иные действия пользователя. Нажатие на клавишу или манипуляция мышью порождают событие, которое передается по цепочке объектов, пока не найдется объект, знающий, как обрабатывать это событие. Для того чтобы событие могло передаваться от объекта к объекту, все объекты программы должны быть объединены в группу. События сами по себе не производят никаких действий в программе, но в ответ на событие могут создаваться новые объекты, модифицироваться или уничтожаться существующие, что и приводит к изменению состояния программы.

9. Какие характеристики должно иметь событие-сообщение?

Для события от объекта (evMessage) задаются два параметра:

* command – код команды, которую необходимо выполнить при появлении данного события;
* infoPtr – передоваемая с событием (сообщением) информация.

10. Привести пример структуры, описывающей событие.

struct KeyDownEvent() {

int keycode;

char charcode;

};

11. Задана структура события

struct TEvent{

int what;

union {

MouseEventType mouse;

KeyDownEvent keyDown;

MessageEvent message;

}

};

Какие значения, и в каких случаях присваиваются полю what?

12. Задана структура события

struct TEvent{

int what;//тип события

union {

int command;//код команды

struct {//параметры команды

int message;

int a; };

};

};

Какие значения, и в каких случаях присваиваются полю command?

13. Задана структура события

struct TEvent {

int what;//тип события

union {

int command;//код комманды

struct {//параметры команды

int message;

int a; };

};

};

Для чего используются поля a и message?

14. Какие методы необходимы для организации обработки сообщений?

Метод получения сообщений (Dialog::GetEvent()), главный цикл отработки(Dialog::Execute), обработчик (Dialog::HandleEvent), метод очистки события (Dialog::ClearEvent), проверка атрибута завершения обработки (Dialog::Valid), обработчик конца работы (Dialog::EndExec) и деструктор обработчика (Dialog::~Dialog)

15. Какой вид имеет главный цикл обработки событий-сообщений?

void Dialog::Execute(){

TEvent event;

do {

EndState = 0;

GetEvent(event); //получить событие

HandleEvent(event); //обработать событие

} while (Valid());

}

16. Какую функцию выполняет метод ClearEvent()? Каким образом?

Если события не происходит, то переменной присваивается значение пустого события.

void Dialog::ClearEvent(TEvent& event){

event.what = evNothing;//пустое событие

}

17. Какую функцию выполняет метод HandleEvent ()?Каким образом?

HandleEvent – обработчик событий. Обрабатывает каждое событие нужным для него образом. Если объект должен обрабатывать определенное событие (сообщение), то его метод HandleEvent должен распознавать это событие и реагировать на него должным образом. Событие может распознаваться, например, по коду команды (поле command).

*Пример*

void TDerivedClass::HandleEvent(TEvent& event) { //Вызов обработчика событий базового класса

TBaseClass::handleEvent( event );

if( event.what == evCommand ) // Если обработчик событий

базового класса событие не обработал

{

switch( event.message.command ) {

case cmCommand1: СlearEvent( event ); break;

case cmCommand2: СlearEvent( event ); break;

…

case cmCommandN: СlearEvent( event ); break;

default: break;

}

};

}

18. Какую функцию выполняет метод GetEvent ()?

GetEvent – формирование события;

19. Для чего используется поле EndState? Какой класс (объект) содержит это поле?

EndState содержится в классе-обработчике Dialog и проверяет, была ли завершена работа пользователя с программой, то есть, будут ли ещё поступать события

20. Для чего используется функция Valid()?

Значение переменной EndState проверяет метод–функция Valid, возвращающая значение true, если “конец работы”. Такой несколько сложный способ завершения работы программы связан с тем, что в активном состоянии могут находиться несколько элементов группы. Тогда метод Valid группы, вызывая методы Valid своих подэлементов, возвратит true, если все они возвращают true. Это гарантирует, что программа завершит свою работу, когда завершат работу все ее элементы.