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: «Основы алгоритмизации и программирования»

Тема: «Программа, управляемая событиями»

Семестр 2

Выполнил работу:

Студент группы ИВТ-22-2б

Цыбуцынин Фёдор Александрович

Проверил:

Доцент кафедры ИТАС

Полякова Ольга Андреевна

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:

ЧЕЛОВЕК (Person)

Имя – string

Возраст – int

Производный класс

СОТРУДНИК (Emloyee)

Заработная плата – float

Должность - string

Группа – Вектор (Vector).

Команды:

-Создать группу (формат команды: m количество элементов группы).

-Добавить элемент в группу (формат команды: +)

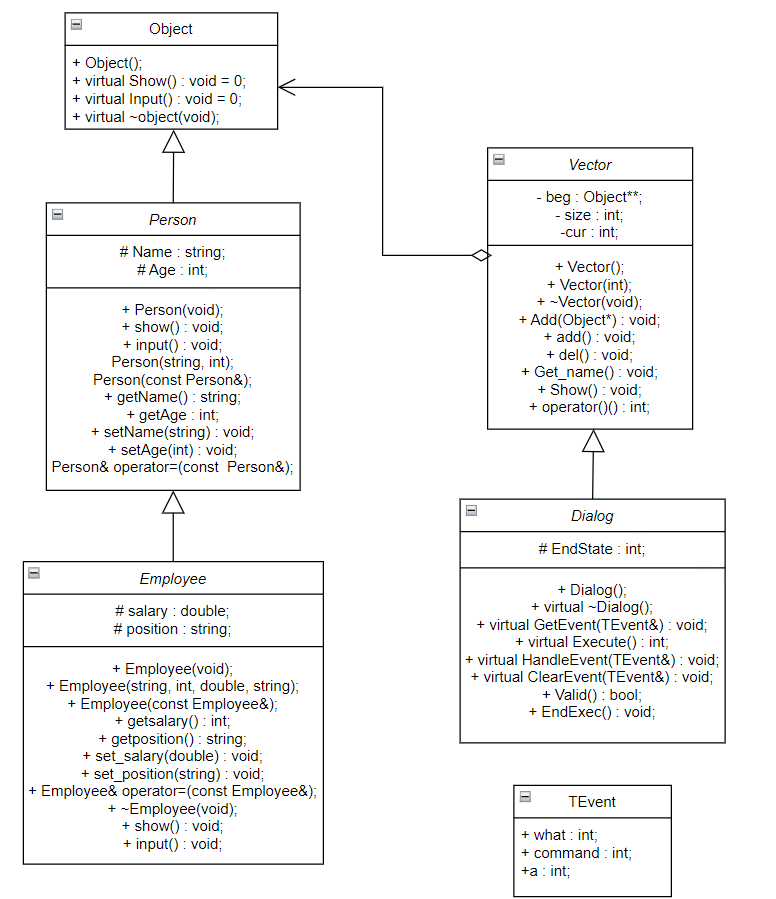
-Удалить элемент из группы (формат команды -)

-Вывести информацию об элементах группы (формат команды: s)

-Вывести информацию об имени элемента с номером k (формат команды : z k, где k – целое число)

-Конец работы (формат команды: q)

**UML - Диаграмма**



**Рис. 1 – UML-диаграмма классов Object, TEvent, Person, Employee, Vector и Dialog**

**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, тоже могут быть группой.

Примеры групп:

- Окно в интерактивной программе, которое владеет такими элементами, как поля ввода и редактирования данных, кнопки, списки выбора, диалоговые окна и т.д. Примерами таких окон являются объекты классов, порождённых от абстрактного класса TGroup (TDeskTop, TWindow, TDialog) в иерархии классов библиотеки Turbo Vision, и объекты классов, порождённых от TWindowObject в иерархии классов библиотеки OWL.

- Агрегат, состоящий из более мелких узлов.

- Огород, состоящий из растений, системы полива и плана выращивания.

- Некая организационная структура (например, ФАКУЛЬТЕТ, КАФЕДРА, СТУДЕНЧЕСКАЯ ГРУППА).

Класс-группа Vector:

class Vector

{

protected:

Object\*\* beg;

int size;

int cur;

public:

Vector(int);

~Vector();

void Add();

void Del();

void Show();

int operator()();

};

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

struct Node

{

Node\* prev = nullptr, \* next = nullptr;

Object\* data;

};

struct Object

{

Object\* beg;

Object();

Object(Object\*);

~Object() {};

virtual void popback() = 0;

virtual void pushback() = 0;

virtual void Show() = 0;

};

class List

{

public:

List(int);//конструктор с параметрами

public:

List();

List(List& t);

~List(void);//деструктор

void popback();//добавление элемента в вектор

void pushback();

void Show();

int operator()();//размер вектора

protected:

Node\* head = nullptr, \* tail = nullptr;

int size;

};

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

List::List()

{

head = 0;

tail = 0;

size = 0;

}

List::List(int a)

{

for(int i = 1; i <= size; i++) pushback();

}

List::List(List& t)

{

Node\* nodeA = t.head;

List newList(t.size);

Node\* newNode = newList.head;

while (nodeA != nullptr)

{

newNode->data = nodeA->data;

nodeA = nodeA->next;

newNode = newNode->next;

}

}

1. Привести пример деструктора для класса-группы Список.

List::~List()

{

Node\* nodeA = head;

while (nodeA != nullptr)

{

head = nodeA->next;

delete nodeA->data;

delete nodeA;

nodeA = head;

}

}

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

void List::Show()

{

if (size == 0) cout << "Empty" << endl;

Node\* p = head;

for (int i = 1; i <= size; i++)

{

p->data->Show(); //вызов метода Show() (позднее связывание)

p = p->next; //передвигаем указатель на следующий объект

}

}

1. Какой вид иерархии даёт группа?

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

Реализовать группу можно разными способами.

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

Абстрактные классы используются в качестве обобщённых концепций, на основе которых можно создавать более конкретные производные классы.

Другими словами, абстрактный класс создаётся для того, чтобы связать классы с общими характеристиками в одну группу.

1. Что такое событие? Для чего используются события?

События лучше всего представить себе как 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 передаётся по цепочке объектов, пока не найдётся объект, знающий, как обрабатывать это событие. Для того чтобы событие могло передаваться от объекта к объекту, все объекты программы должны быть объединены в группу. Отсюда следует, что прикладная программа должна быть объектом-группой, в которую должны быть включены все объекты, используемые в программе.

1. Какие характеристики должно иметь событие-сообщение?

Сообщение передаваемое от одних объектов другим имеет, как правило, следующие характеристики:

- код класса сообщения, отличающий сообщения объектов одного класса от объектов другого класса

- адрес объекта, которому предназначено сообщение (м. б. не задан, тогда сообщение могут прочитать все объекты)

- информационное поле

1. Привести пример структуры, описывающей событие.

struct TEvent

{

int what;

int command;

};

1. Задана структура события

struct TEvent

{

int what;

union

{

MouseEventType mouse;

KeyDownEvent keyDown;

MessageEvent message;

}

};

Какие значения, и в каких случаях присваиваются полю what?

- evNothing – это пустое событие, которое означает, что ничего делать не надо. Полю what присваивается значение evNothing, когда событие обработано каким-либо объектом.

- evMessage  событие-сообщение от объекта.

Для события от объекта (evMessage) задаются два параметра :

- command – код команды, которую необходимо выполнить при появлении данного события;

- передаваемая с событием информация (сообщение).

1. Задана структура события

struct TEvent

{

int what;//тип события

union

{

int command;//код комманды

struct//параметры команды

{

int message;

int a;

};

};

};

Какие значения, и в каких случаях присваиваются полю command?

Полю command присваиваются коды различных определённых команд. При получении того или иного сообщения, поле command принимает одно из ранее указанных кодов, например:

const int evNothing = 0;

const int evMessage = 100;

const int cmAdd = 1;

const int cmDel = 2;

const int cmGet = 3;

const int cmShow = 4;

const int cmMake = 6;

const int cmQuit = 101;

1. Задана структура события

struct TEvent

{

int what;//тип события

union

{

int command;//код команды

struct//параметры команды

{

int message;

int a;

};

};

};

Для чего используются поля a и message?

Эти поля необходимы тогда, когда программа получает сообщение с параметром. Поле message здесь будет отвечать за то, какая команда поступила, а поле a будет являться параметром команды.

1. Какие методы необходимы для организации обработки сообщений?

class Dialog :

{

public:

Dialog(void);

~Dialog(void); // деструктор

void GetEvent(TEvent& event); // получение события

void Execute(); // главный цикл обработки

void HandleEvent(TEvent& event); // обработка события

void ClearEvent(TEvent& event); // удаление события

bool Valid(); // провека атрибута завершения обработки

void EndExec(); // обработчик конца работы

protected:

int EndState; // конечное положение

};

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

void Dialog::Execute()

{

TEvent event;

do

{

EndState = 0;

GetEvent(event);

HandleEvent(event);

} while (Valid());

}

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

ClearEvent очищает событие, присваивая полю event.what значение evNothing, когда оно обработано, чтобы оно не обрабатывалось далее.

void Dialog::ClearEvent(TEvent& event)

{

event.what = evNothing;

}

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

HandleEvent – обработчик событий. Обрабатывает каждое событие нужным для него образом. Если объект должен обрабатывать определённое 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).

void Dialog::HandleEvent(TEvent& event)

{

if (event.what == evMessage)

{

switch (event.command)

{

case cmMake:

cout << "Enter size: ";

cin >> size;

beg = new Object \* [size];

cur = 0;

ClearEvent(event);

break;

case cmAdd:

Add();

ClearEvent(event);

break;

case cmDel:

Del();

ClearEvent(event);

break;

case cmShow:

Show();

ClearEvent(event);

break;

case cmQuit:

EndExec();

ClearEvent(event);

break;

case cmGet:

Get\_Name();

ClearEvent(event);

break;

}

}

}

1. Какую функцию выполняет метод GetEvent ()?

Метод GetEvent() формирует событие:

void Dialog::GetEvent(TEvent& event)

{

setlocale(LC\_ALL, "ru");

string OpInt = "m+-szq";

string s;

char code;

cout << "Введите команду: ";

cin >> s;

code = s[0];

if (OpInt.find(code) >= 0)

{

event.what = evMessage;

switch (code)

{

case 'm':

event.command = cmMake;

break;

case '+':

event.command = cmAdd;

break;

case '-':

event.command = cmDel;

break;

case 's':

event.command = cmShow;

break;

case 'z':

event.command = cmGet;

break;

case 'q':

event.command = cmQuit;

break;

}

}

else event.what = evNothing;

}

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

Поле EndState содержится в классе-обработчике Dialog. Оно проверяет, окончил ли пользователь свою работу:

void Dialog::EndExec()

{

EndState = 1;

}

1. Для чего используется функция Valid()?

Функция Valid проверяет значение поля EndState:

- Если программа не завершена, она возвращает значение истины

- Если программа завершена, то значение лжи

Функция используется в главном обработчике событий Execute.

class Dialog : public Tree

{

public:

...

bool Valid(); // провека атрибута завершения обработки

...

};

void Dialog::Execute()

{

TEvent event;

do

{

EndState = 0;

GetEvent(event);

HandleEvent(event);

} while (Valid());

}