Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01 - «Информатика и вычислительная техника»

**Отчет по лабораторной работе “Классы” №8**

**по дисциплине**

**«Теория алгоритмов и структуры данных»**

Выполнил студент гр. ИВТ-21-1б

Ахунов Руслан Булатович

Проверил:

Ст. Преподаватель кафедры ИТАС

Яруллин Денис Владимирович

(оценка) (подпись)

г. Пермь - 2022

**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.

Вариант 2:

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)

**Анализ задачи:**

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 cmFind = 7;

const int cmQuit = 101;

struct TEvent

{

    int what;

    union

    {

        int command;

        struct

        {

            int message;

            int a;

        };

    };

};

class Object

{

public:

    Object() {}

    virtual void Show() = 0;

    virtual void input() = 0;

    virtual ~Object() {}

    virtual void HandleEvent(const TEvent& e) = 0;

};

В самом начале инициализируются несколько константных значений, и они являются базовыми командами в дальнейшем. Так же есть абстрактный класс Object, так как в нем все функции чисто виртуальные.

void Add()

    {

        Object\* p;

        cout << "1. Person" << endl;

        cout << "2. Employee" << endl;

        int y;

        cin >> y;

        if (y == 1)

        {

            Person\* tmp = new Person;

            tmp->input();

            p = tmp;

            if (cur < size)

            {

                beg[cur] = p;

                cur++;

            }

        }

        else

        {

            if (y == 2)

            {

                Employee\* tmp1 = new Employee;

                tmp1->input();

                p = tmp1;

                if (cur < size)

                {

                    beg[cur] = p;

                    cur++;

                }

            }

            else return;

        }

    }

Класс vector, в нем есть метод Add, который является первым из событий, то есть определяется, какой класс будет в дальнейшем использоваться.

class Dialog : public Vector

{

protected:

    int EndState;

public:

    Dialog() : Vector()

    {

        EndState = 0;

    }

    ~Dialog() {}

    void GetEvent(TEvent& event)

    {

        string Icode = "+-szqm";

        string s;

        string param;

        char code;

        cout << '>';

        cin >> s;

        code = s[0];

        if (Icode.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 'q': event.command = cmQuit; break;

            case 'z': event.command = cmFind; break;

            }

            if (s.length() > 1)

            {

                param = s.substr(1, s.length() - 1);

                int A = atoi(param.c\_str());

                event.a = A;

            }

        }

        else event.what = evNothing;

    }

    int Execute()

    {

        TEvent event;

        do {

            EndState = 0;

            GetEvent(event);

            HandleEvent(event);

        } while (!Valid());

        return EndState;

    }

    int Valid()

    {

        if (EndState == 0) return 0;

        else return 1;

    }

    void ClearEvent(TEvent& event)

    {

        event.what = evNothing;

    }

    void EndExec()

    {

        EndState = 1;

    }

    void HandleEvent(TEvent& event)

    {

        if (event.what == evMessage)

        {

            switch (event.command)

            {

            case cmMake:

                size = event.a;

                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 cmFind:

                int tmp = event.a;

                //cout << "Введите номер элемента: "; cin >> tmp;

                Find(tmp);

                ClearEvent(event);

                break;

            //default: HandleEvent(event);

            }

        }

    }

};

Производный класс от класса vector, Dialog, в котором через проверку на нужный символ выполняется соответствующие действие, из заранее заготовленных, с помощью метода HandleEvent – обработчик событий.

int main()

{

    setlocale(LC\_ALL, "Russian");

    cout << "+ - добавить элемент" << "\n- - удалить элемент" << "\nm - создать группу (m6 - из 6 элементов)" << "\ns - просмотр группы" << "\nz - Вывести определенный элемент (Например z2 - о 2-ом элементе)" << "\nq - конец" << "" << endl;

    Dialog D;

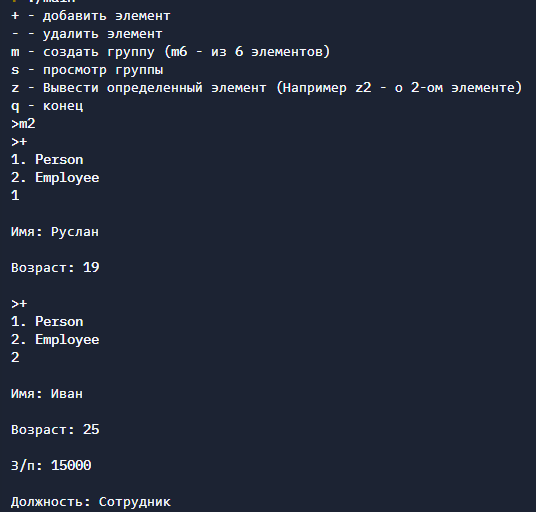
    D.Execute();

    return 0;

}

В функции main, выводиться помощь, как работать с данной программой и какой символ за что отвечает.

**Результат работы программы**

****

**Код программы**

#include <iostream>

#include <string>

using namespace std;

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 cmFind = 7;

const int cmQuit = 101;

struct TEvent

{

    int what;

    union

    {

        int command;

        struct

        {

            int message;

            int a;

        };

    };

};

class Object

{

public:

    Object() {}

    virtual void Show() = 0;

    virtual void input() = 0;

    virtual ~Object() {}

    virtual void HandleEvent(const TEvent& e) = 0;

};

class Person : public Object

{

protected:

    string name;

    int age;

public:

    Person()

    {

        name = "nothing";

        age = 0;

    }

    virtual ~Person() {}

    void Show() override

    {

        cout << "\nИмя: " << name;

        cout << "\nВозраст: " << age << endl;

    }

    void input() override

    {

        cout << "\nИмя: ";

        cin >> name;

        cout << "\nВозраст: ";

        cin >> age;

        cout << endl;

    }

    Person(string n, int a)

    {

        name = n;

        age = a;

    }

    Person(const Person& tmp)

    {

        name = tmp.name;

        age = tmp.age;

    }

    string GetName() { return name; }

    int GetAge() { return age; }

    void SetName(string n)

    {

        name = n;

    }

    void SetAge(int a)

    {

        age = a;

    }

    Person& operator=(const Person& tmp)

    {

        if (&tmp == this)

        {

            return \*this;

        }

        name = tmp.name;

        age = tmp.age;

        return \*this;

    }

    void HandleEvent(const TEvent& e)

    {

        if (e.what == evMessage)

        {

            switch (e.command)

            {

                case cmGet: cout << "Имя: " << GetName() << endl;

                break;

            }

        }

    }

};

class Employee : public Person

{

protected:

    float zp;

    string job;

public:

    Employee() : Person()

    {

        zp = 0;

        job = "nothing";

    }

    ~Employee() {}

    void Show() override

    {

        cout << "\nИмя: " << name;

        cout << "\nВозраст: " << age;

        cout << "\nЗ/п: " << zp;

        cout << "\nДолжность: " << job << endl;

    }

    void input() override

    {

        cout << "\nИмя: ";

        cin >> name;

        cout << "\nВозраст: ";

        cin >> age;

        cout << "\nЗ/п: ";

        cin >> zp;

        cout << "\nДолжность: ";

        cin >> job;

        cout << endl;

    }

    Employee(string n, int a, float z, string j) : Person(n, a)

    {

        zp = z;

        job = j;

    }

    Employee(const Employee& tmp)

    {

        name = tmp.name;

        age = tmp.age;

        zp = tmp.zp;

        job = tmp.job;

    }

    float GetZp() { return zp; }

    string GetJob() { return job; }

    void SetZp(float z)

    {

        zp = z;

    }

    void SetJob(string j)

    {

        job = j;

    }

    Employee& operator=(const Employee& tmp)

    {

        if (&tmp == this) return \*this;

        name = tmp.name;

        age = tmp.age;

        zp = tmp.zp;

        job = tmp.job;

    }

};

class Vector

{

protected:

    Object\*\* beg;

    int size;

    int cur;

public:

    Vector()

    {

        beg = nullptr;

        size = 0;

        cur = 0;

    }

    Vector(int n)

    {

        beg = new Object\* [n];

        cur = 0;

        size = n;

    }

    ~Vector()

    {

        if (beg != 0) delete[] beg;

        beg = 0;

    }

    void Add()

    {

        Object\* p;

        cout << "1. Person" << endl;

        cout << "2. Employee" << endl;

        int y;

        cin >> y;

        if (y == 1)

        {

            Person\* tmp = new Person;

            tmp->input();

            p = tmp;

            if (cur < size)

            {

                beg[cur] = p;

                cur++;

            }

        }

        else

        {

            if (y == 2)

            {

                Employee\* tmp1 = new Employee;

                tmp1->input();

                p = tmp1;

                if (cur < size)

                {

                    beg[cur] = p;

                    cur++;

                }

            }

            else return;

        }

    }

    void Show()

    {

        if (cur == 0) cout << "Пусто" << endl;

        Object\*\* p = beg;

        for (int i = 0; i < cur; i++)

        {

            (\*p)->Show();

            p++;

        }

    }

    int operator()()

    {

        return cur;

    }

    void Del()

    {

        if (cur == 0) return;

        cur--;

    }

    void Find(int tmp)

    {

        beg[tmp];

        Object\*\* p = beg;

        for (int i = 0; i < cur; i++)

        {

            if (i == tmp - 1)

            {

                (\*p)->Show();

            }

            p++;

        }

    }

    void HandleEvent(const TEvent& e)

    {

        if (e.what == evMessage)

        {

            Object\*\* p = beg;

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

            {

                (\*p)->HandleEvent(e);

                p++;

            }

        }

    }

};

class Dialog : public Vector

{

protected:

    int EndState;

public:

    Dialog() : Vector()

    {

        EndState = 0;

    }

    ~Dialog() {}

    void GetEvent(TEvent& event)

    {

        string Icode = "+-szqm";

        string s;

        string param;

        char code;

        cout << '>';

        cin >> s;

        code = s[0];

        if (Icode.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 'q': event.command = cmQuit; break;

            case 'z': event.command = cmFind; break;

            }

            if (s.length() > 1)

            {

                param = s.substr(1, s.length() - 1);

                int A = atoi(param.c\_str());

                event.a = A;

            }

        }

        else event.what = evNothing;

    }

    int Execute()

    {

        TEvent event;

        do {

            EndState = 0;

            GetEvent(event);

            HandleEvent(event);

        } while (!Valid());

        return EndState;

    }

    int Valid()

    {

        if (EndState == 0) return 0;

        else return 1;

    }

    void ClearEvent(TEvent& event)

    {

        event.what = evNothing;

    }

    void EndExec()

    {

        EndState = 1;

    }

    void HandleEvent(TEvent& event)

    {

        if (event.what == evMessage)

        {

            switch (event.command)

            {

            case cmMake:

                size = event.a;

                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 cmFind:

                int tmp = event.a;

                //cout << "Введите номер элемента: "; cin >> tmp;

                Find(tmp);

                ClearEvent(event);

                break;

            //default: HandleEvent(event);

            }

        }

    }

};

int main()

{

    setlocale(LC\_ALL, "Russian");

    cout << "+ - добавить элемент" << "\n- - удалить элемент" << "\nm - создать группу (m6 - из 6 элементов)" << "\ns - просмотр группы" << "\nz - Вывести определенный элемент (Например z2 - о 2-ом элементе)" << "\nq - конец" << "" << endl;

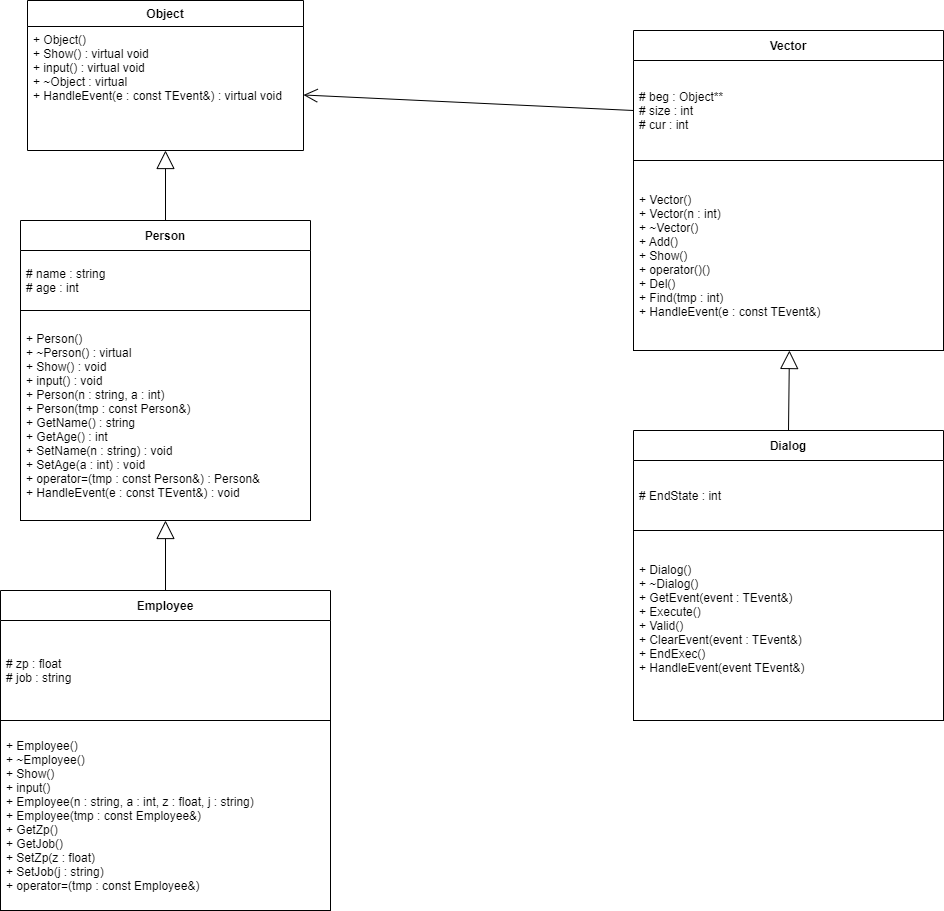
    Dialog D;

    D.Execute();

    return 0;

}

**Диаграмма класса**



**Вопросы**

* + 1. **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 группой.

Примеры групп:

* + 1. 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, порожденных от абстрактного класса TGroup( TDeskTop, TWindow, TDialog) в иерархии классов библиотеки Turbo Vision, и объекты классов, порожденных от TWindowObject в иерархии классов библиотеки OWL.
    2. Агрегат, состоящий из более мелких узлов.
    3. Огород, состоящий из растений, системы полива и плана выращивания.
    4. Некая организационная структура (например, ФАКУЛЬТЕТ, КАФЕДРА, СТУДЕНЧЕСКАЯ ГРУППА).
    5. **Привести пример описания класса-группы Список (List).**

1. template <class T>
2. class List
3. **Привести пример конструктора (с параметром, без параметров, копирования) для класса-группы Список.**

List()

{

beg = 0;

size = 0;

cur = 0;

}

List(int n)

{

beg = new object \*[n];

size = n;

cur = 0;

}

List(const List& l)

{

beg = l.beg;

size = l.size;

cur = l.cur;

}

1. **Привести пример деструктора для класса-группы Список.**

~List(void){}

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

void Info\_All()

{

if (cur == 0) cout << "Empty\n";

object\*\* p = beg;//указатель на указатель типа Object

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

{

(\*p)->show();//вызов метода Show() (позднее срабатывание)

++p;//передвигаем указатель на след объект

}

}

1. **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.

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

Для удобства

1. **Что такое событие? Для чего используются события?**

Событие в объектно-ориентированном программировании — это сообщение, которое возникает в различных точках исполняемого кода при выполнении определённых условий. События предназначены для того, чтобы иметь возможность предусмотреть реакцию программного обеспечения.

1. **Какие характеристики должно иметь событие-сообщение?**

Объект инициирует событие и вызываются все объекты-обработчики. Т.е. от одного объекта к нескольким. Причем объект инициатор события может ничего не «знать» об его обработчиках, поэтому событие называют исходящим вызовом. Раз уж в С++ события на уровне языка не поддерживаются, значит стоит организовать их на уровне библиотеки.

1. **Привести пример структуры, описывающей событие.**

struct TEvent

{

int what;

union

{

int command;

struct

{

int message;

int a;

};

};

};

1. **Задана структура события**

**Struct TEvent**

**{**

**int what;**

**union**

**{**

**MouseEventType mouse;**

**KeyDownEvent keyDown;**

**MessageEvent message;**

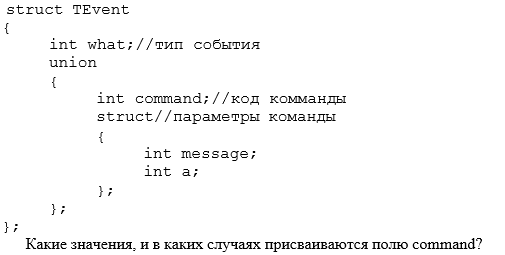
**}**

**};**

**Какие значения, и в каких случаях присваиваются полю what?**

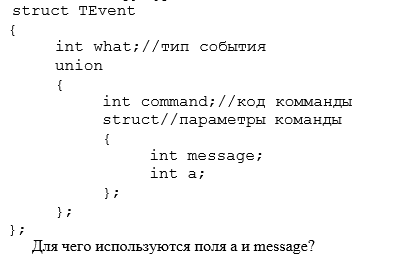
Int значение

1. **Задана структура события**

****

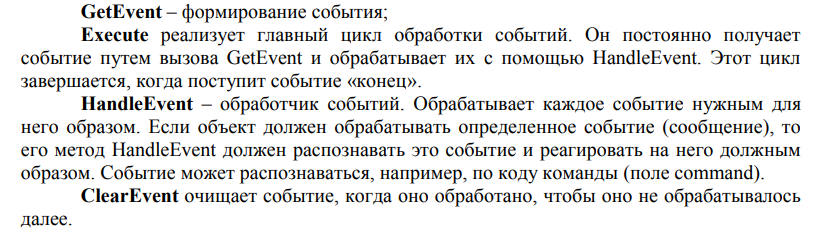
Execute реализует главный цикл обработки событий. Он постоянно получает событие путем вызова GeEvent и обрабатывает их с помощью HandleEvent. Этот цикл завершается, когда поступит событие «конец». HandleEvent – обработчик событий. Обрабатывает каждое событие нужным для него образом.

1. **Задана структура события**

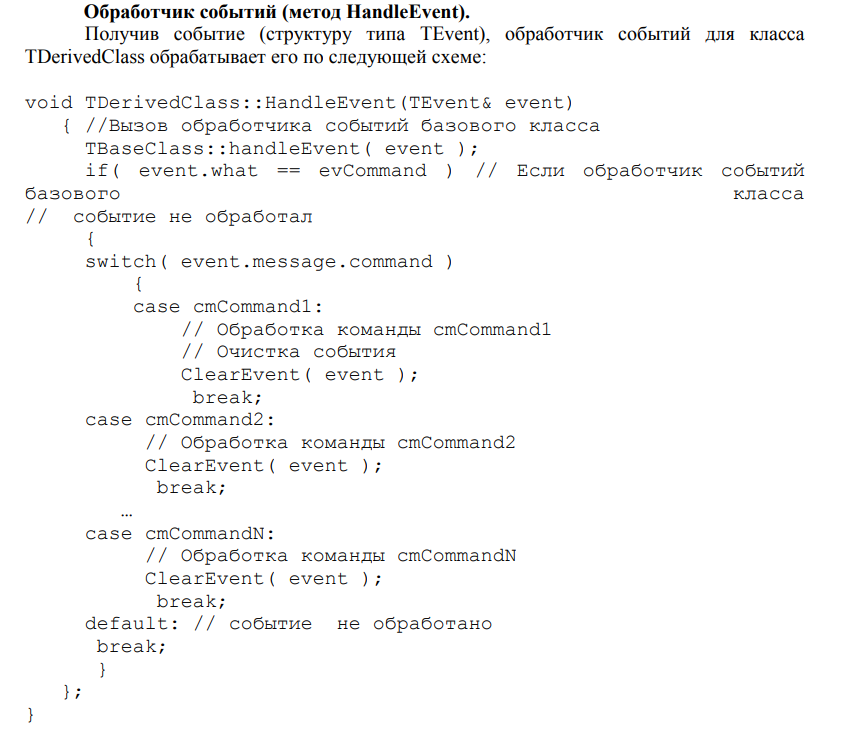
****

**Код собщения, а – преобразованный параметр(команда) в число.**

1. **Какие методы необходимы для организации обработки сообщений?**

****

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

****

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

Метод ClearEvent − очистка события. ClearEvent очищает событие, присваивая полю Event.What значение evNothing. + + Главный цикл обработки событий (метод Execute).

1. **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).

1. **Какую функцию выполняет метод GetEvent ()?**

Метод GetEvent наследуется всеми видимыми элементами от TView и является основным источником событий. Этот метод вначале проверяет, не подготовил ли событие метод PutEvent и, если это так, возвращает его.

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

Конечное состояние программы. Содержится в классе Dialog.

1. **Для чего используется функция Valid()?**

Подходит ли данное значение для дальнейших действие и есть ли вообще какое-то значение.