Лабораторная работа № 8 - "Программа, управляемая событиями"

Цель работы

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Разработка программы, управляемой событиями.

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.

> Базовый класс: ПЕЧАТНОЕ_ИЗДАНИЕ(PRINT) Название– string Автор – string Производный класс КНИГА (BOOK) Количество страниц - int Издательство - string Группа – Список (List). 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 о названии элемента группы с номером k (формат команды: z k, где k – целое число) Конец работы (формат команды: q)

Описание класса-контейнера

Класс-контейнер List представляет собой динамический массив указателей на объекты базового класса Object. Он обеспечивает следующие операции:

- Добавление объектов (Add)
- Удаление объектов (Del)
- Просмотр всех объектов (Show)
- Получение текущего количества объектов (operator())
- Обработка событий (HandleEvent)

Определение компонентных функций

Основные функции класса List:

- Add() - добавляет новый объект (либо PRINT, либо BOOK) в список после ввода его атрибутов
- Show() - выводит информацию о всех объектах в списке
- Del() - уменьшает счетчик объектов (фактическое удаление не реализовано)
- operator() - возвращает текущее количество объектов
- HandleEvent() - обрабатывает события для всех объектов в списке

Описание класса-итератора и его компонентных функций

Функция main()

```
int main() {
    Dialog D;
    D.Execute();
    return 0;
}
```

Объяснение результатов работы программы

Программа позволяет:

- Создавать группу объектов указанного размера (команда 'm')
- Добавлять объекты типа PRINT или BOOK (команда '+')
- Удалять объекты (команда '-')
- Просматривать все объекты (команда '?')
- Получать информацию об отдельных атрибутах (команда '/')
- Завершать работу (команда 'q')

Ответы на контрольные вопросы

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

Ответ: Класс-группа - это класс, который управляет коллекцией других объектов. Примеры: List, Vector, Array.

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

Ответ:

```
class List {
    Object** begin;
    int size;
    int current;
public:
    List(int n);
    ~List();
    void Add();
    void Show();
    void Del();
```

```
    int operator()() const;
};
```

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

Ответ:

```
// С параметром
List::List(int n) {
    begin = new Object*[n];
    current = 0;
    size = n;
}

// Без параметра (не реализован в данном коде)
List::List() {
    begin = nullptr;
    current = 0;
    size = 0;
}

// Копирования (не реализован в данном коде)
List::List(const List& other) {
    size = other.size;
    current = other.current;
    begin = new Object*[size];
    for(int i=0; i<current; i++)
        begin[i] = other.begin[i];
}
```

4. Привести пример деструктора для класса-группы Список.

Ответ:

```
List::~List() {
    delete[] begin;
    begin = nullptr;
}
```

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

Ответ:

```
void List::Show() {
    if(current == 0) cout << "Empty" << endl;
    Object** p = begin;
```

```
    for(int i=0; i<current; i++) {
        (*p)->Show();
        p++;
    }
}
```

6. Какой вид иерархии дает группа?

Ответ: Группа реализует иерархию "часть-целое", где группа (целое) содержит отдельные объекты (части).

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

Ответ: Абстрактный класс определяет общий интерфейс для всех объектов группы, что позволяет обрабатывать их единообразно через указатели на базовый класс.

8. Что такое событие? Для чего используются события?

Ответ: Событие - это изменение состояния системы, о котором нужно уведомить заинтересованные объекты. События используются для организации реактивного поведения программы.

9. Какие характеристики должно иметь событие-сообщение?

Ответ: Событие-сообщение должно содержать: тип события, код команды и дополнительные параметры.

10. Привести пример структуры, описывающей событие.

Ответ:

```
struct TEvent {
    int what;
    union {
        int command;
        struct {
            int message;
            int a;
        };
    };
};
```

11. Задана структура события

```
    struct TEvent
{
int what;
union {
```

```
MouseEventType mouse;
KeyDownEvent keyDown;
MessageEvent message;
}
};
```

Какие значения, и в каких случаях присваиваются полю what?

Ответ: Поле what принимает значение evNothing (0) для пустого события или evMessage (100) для события-сообщения.

12. Задана структура события

```
struct TEvent {
   int what;
   union {
      int command;
      struct {
         int message;
         int a;
      };
   };
};
```

Какие значения, и в каких случаях присваиваются полю command?

Ответ: Поле command принимает значения:

- cmAdd (1) - добавить объект
- cmDel (2) - удалить объект
- cmGet (3) - получить атрибут
- cmShow (4) - показать группу
- cmMake (6) - создать группу
- cmQuit (101) - завершить работу

13. Задана структура события

```
struct TEvent {
   int what;
   union {
       int command;
       struct {
         int message;
         int a;
      };
   };
};
```

Для чего используются поля a и message?

Ответ: Поле a используется для передачи дополнительного параметра (например, размера группы), message - для типа сообщения.

14. Какие методы необходимы для организации обработки сообщений?

Ответ:

- GetEvent() - получение события
- HandleEvent() - обработка события
- ClearEvent() - очистка события
- Valid() - проверка состояния

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

Ответ:

```
do {
    GetEvent(event);
    HandleEvent(event);
} while (!Valid());
```

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

Ответ: ClearEvent() сбрасывает событие в состояние "пустое" путем установки event.what = evNothing.

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

Ответ: HandleEvent() обрабатывает события, вызывая соответствующие методы в зависимости от типа события и команды.

18. Какую функцию выполняет метод GetEvent ()?

Ответ: GetEvent() получает команду от пользователя, преобразует ее в структуру события и заполняет соответствующие поля.

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

Ответ: EndState используется для определения завершения работы программы. Содержится в классе Dialog.

20. Для чего используется функция Valid()?

Ответ: Valid() проверяет, должно ли приложение продолжать работу (возвращает 0) или завершиться (возвращает 1).

Лабораторная работа доступна в GitHub репозитории hanriel/PSTU-CPP