Вложенные классы. Композиция.

Различия между агрегацией и композицией

FR

Композиция

- Композиция – это отношения **HAS A** («имеет», или «целое-часть» между объектами, которые сильно связаны между собой (один объект не существует без другого объекта или же функционал целого будет серьёзно ограничен, если у него отсутствует какая-то часть)
- **Пример:** автомобиль и двигатель (конечно, чисто теоретически, они могут существовать друг без друга, но будут абсолютно бесполезны – автомобиль без двигателя не поедет).
  Также – ноутбук и тачпад (ноутбук без тачпада может и будет работать, но его функционал будет серьёзно ограничен).

Агрегация

- Агрегация - это отношения **HAS A** («имеет», или «целое-часть» между объектами, которые слабо связаны между собой (один объект может существовать без другого)
- **Пример:** самолёт и пассажиры самолёта.

FR

- Предположим, что мы захотели более детально описать нашего тигра, и начали с описания характеристик, которые присущи отдельным его частям. Например, мы ввели такие свойства как цвет и диаметр глаз, длина и острота клыков, а также размер головы (длина, ширина и высота). Тогда наш класс разрастётся, и его приватная часть будет выглядеть следующим образом:

```
 int speed; // Speed of a tiger
string breed; // Breed of a tiger
int age; // Age of a mammal
int size; // Size of a mammal
bool isHungry; // A mammal can be hungry or not
string colourOfSkin;
string colourOfEyes;
double head_width;
double head_height;
double head_depth;
double diameter_of_eyes;
```

- Для того, чтобы избежать «мешанины» в свойствах класса можно создать отдельный вложенный класса Голова (Head). При этом класс Голова (Head) будет вложен в класс «Тигр», поскольку голова тигра может существовать только как часть конкретного тигра. Аналогично создадим классы для клыков (Fang) и глаз (Eye), которые, в свою очередь, будут вложены в класс «Голова» (Head), поскольку клыки и глаза отдельно от головы не существуют.

Для классов **Head, Eye и Fang** определены: два конструктора (конструктор копирования и конструктор по-умолчанию), а также (только для класса **Head**) – деструктор. Он высвобождает динамическую память, поскольку в состав класса Head входит массив из глаз и клыков (у тигра 2 клыка и 2 глаза).

```
class Head {

    class Eye {
        string colour; // colour of an eye
        double diameter; // diameter of an eye
    public:
        Eye(string colour = "red", double diameter = 0.5) : colour(colour),
diameter(diameter) {}
        Eye(const Eye &e) : colour(e.colour), diameter(e.diameter) {}
        friend ostream &operator<<(ostream &output, const Tiger &T);
    };

    class Fang {
        int length; // length of a fang
        bool isDull; // dull or not
    public:
        friend ostream &operator<<(ostream &output, const Tiger &T);
        Fang(int length = 34, bool isDull = false) : length(length),
isDull(isDull) {}
        Fang(const Fang &f) : length(f.length), isDull(f.isDull) {}
    };
};
```

Для классов Head, Eye и Fang определены: два конструктора (конструктор копирования и конструктор по-умолчанию), а также (только для класса Head) - деструктор

Поля класса Head

```
double height; // measurements of a head: height
double width;  // measurements of a head: width
double depth; // measurements of a head: depth
int h_degree; // angle of rotation
Eye* eyes; // eyes are a part of head
Fang* fangs; // fangs are a part of head
```

Конструктор класса Head

```
// Default constructor
Head(string colourOfEyes= "red", double diameterOfEyes=5.0,
     int lengthOfFangs=10, bool areFangsDull=false,
     double height=54.3, double width=22.4, double depth=54.6) :
        height(height), width(width), depth(depth), h_degree(0) {
    eyes = new Eye[2]; // init array of eyes
    eyes[0] = Eye(colourOfEyes, diameterOfEyes); // init eye
    eyes[1] = Eye(colourOfEyes, diameterOfEyes);
    fangs= new Fang[2]; // init array of fangs
    fangs[0] = Fang(lengthOfFangs, areFangsDull); // init fang
    fangs[1] = Fang(lengthOfFangs, areFangsDull);
}
```

Здесь происходит инициализация двух массивов: eyes (массив глаз) и клыков (fangs), и затем – инициализация каждого элемента массива. Остальные поля класса инициализируются с помощью списков инициализации (вне тела конструктора)

Конструктор копирования класса Head

FR

```
// Copy constructor
Head(const Head& h):height(h.height), width(h.width),
depth(h.depth), h_degree(h.h_degree) {
    eyes = new Eye[2];
    eyes[0] = h.eyes[0];
    eyes[1] = h.eyes[1];
    fangs = new Fang[2];
    fangs[0] = Fang(h.fangs[0]);
    fangs[1] = Fang(h.fangs[1]);
}
```

Аналогично:
происходит инициализация двух массивов: eyes (массив глаз) и клыков (fangs), и затем – инициализация каждого элемента массива. Остальные поля класса инициализируются с помощью списков инициализации (вне тела конструктора). Я намеренно использовал (в лекционном материале) два способа вызова конструктора копирования: для глаз – с помощью оператора присваивания «=«, а для клыков – напрямую, через передачу ссылки на объект, который нужно скопировать (оригинал).

Класс «тигр»

- Голова, как мы уже говорили, является неотъемлемой частью тигра. В конструкторах класса «тигр», с помощью списков инициализации, мы проинициализировали голову (а вместе с ней, глаза и клыки, потому что вызов конструктора класса «Голова» вызывает конструкторы классов «Клыки» и «Глаза»), и затем происходит инициализация остальных полей класса.

```
// Default constructor
Tiger(int age=5, int size=9, bool isHungry=false, string colourOfSkin="orange", string
colourOfEyes="red", double diameterOfEyes=54.0,
      int lengthOfFangs=34, bool areFangsDull=false,
      double h_height=3.5, double h_width=6.7, double h_depth=5.4,
      int speed=340, string breed="Bengal") : head(colourOfEyes, diameterOfEyes,
lengthOfFangs, areFangsDull, h_height, h_width, h_depth),
                                              age(age), size(size), speed(speed),
breed(breed), isHungry(isHungry), colourOfSkin(colourOfSkin) {}

// copying constructor
Tiger(Tiger const& instance):age(instance.age), size(instance.size),
speed(instance.speed), breed(instance.breed), isHungry(instance.isHungry),
colourOfSkin(instance.colourOfSkin),
head(instance.head)
{}
```

Класс «Тигр»

- Компоненты данных класса тигр теперь будут выглядеть следующим образом (с учётом того, что мы вынесли характеристики глаз, клыков и головы в отдельный вложенный класс):

```
int speed; // Speed of a tiger
string breed; // Breed of a tiger
int age; // Age of a mammal
int size; // Size of a mammal
bool isHungry; // A mammal can be hungry or not
string colourOfSkin;
Head head;
```