

- ①  $A = B + C$   
 ②  $A = B * C$   
 ④ trace (A);  $A.print();$   
 ⑤ determinant (A);  $- A.set();$   
 ⑥ print(A);  $A.print();$  - Bobog

Лекция № 29. 11.

## Наследование в ООП

```
class Animal {
public:
```

String name

public :

Animal (String new\_name = " ");

sound () /

.~~new~~  
name  
new\_name

}

class Dog: public Animal {  
public: string type;

Dog (string new-name = " " );

sound();  
}

Animal

int main()

Dog bobik ("Bobik"); Bobik("Bobik")

Dog noname; //Dog(),

Dog

Dog

type;

Animal

name;

! При создании объектов неизвестного класса  
возводится исключение `InstantiationException`.

Если есть наследование, то  
объекты наследников  
конструируются на  
уровне класса.

(Для аргумента)

Но если нужно создать  
конкретный конструктор, то:

```
class Dog : public Animal
public:
    string type;
public:
    Dog(string new_name = " ");
    Sound();
```

Ограничения доступа:

public - доступно <sup>чтобы</sup>  
protected - доступно <sup>чтобы</sup> ~~остальным~~  
private - доступно <sup>чтобы</sup> ~~никому~~  
<sup>чтобы</sup> ~~никому~~

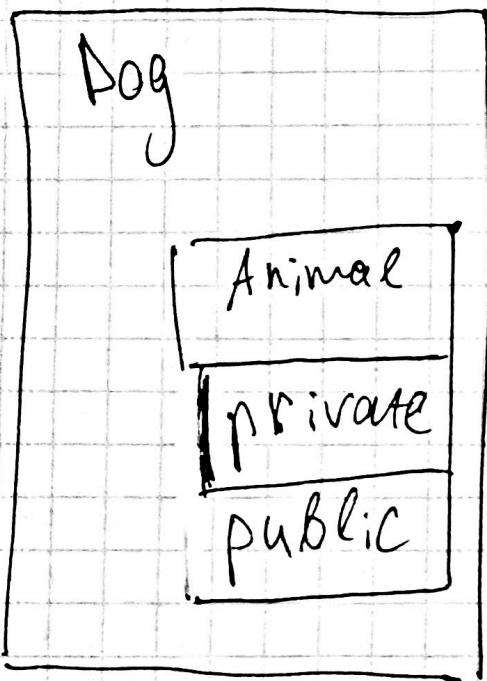
## Наследование

class Dog : public Animal {  
 public  $\Rightarrow$  public  
 protected  $\Rightarrow$  protected }  
 private  $\Rightarrow$  private }  
 <sup>изменение</sup>  
 <sup>доступа</sup>  
 <sup>унаследовано</sup>

class Dog : private Animal {  
 public  $\Rightarrow$  ~~public~~ private }  
 <sup>изм - в</sup>  
 protected  $\Rightarrow$  private }  
 private  $\Rightarrow$  ~~негативен~~  
 <sup>доступа</sup>  
 <sup>унаследовано</sup>

Animal (new\_name) {}.

class Dog : protected Animal  
 public  $\Rightarrow$  protected }  
 protected  $\Rightarrow$  protected }  
 private  $\Rightarrow$  negacywne }  
rynek - ul  
gospodarka  
wpu  
naczelny - mu



(Możemy b. na koniec implementować  
 ypsilon gospodarka. Czyli nowe net  
 TO u implementować nowy )

class Dog : public Animal  
 public:  
 string type;  
 public : (Dog(string new\_name = " " ))  
 protected:  
 using Animal :: sound // napis

некий за protected уровень  
поступа в Dog и  
алогия sound();

Следует выделить некие  
пункты:

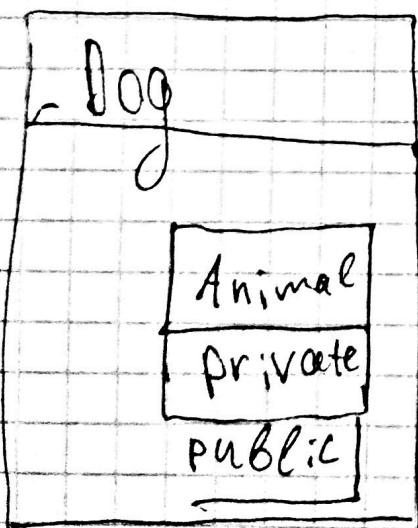
class Dog : public Animal {  
public:

string type;  
public:

Dog (string new\_name = "");

sound () = delete; // запрещен  
алогия

new\_name  
(установка)



Dog bobik ("Bobik");

Animal creature ("Хозяин");  
creature = bobik //

~~Беспечимые~~ /

6) Repellent creature won't attack  
voicē repellent Bobik, not other  
attacker u Animal

Dog bobik ("Bobik"),

Animal \* creature\_01 = ?Bobik

Animal & creature\_02 = Bobik.

Bobik • Sound(); // Sound by Dog

creature\_01 → sound(); // Sound by

creature\_02. Sound(); // Sound by Animal

void Sound - three \_ times(

Animal & creature){

creature. Sound();

creature. Sound();

creature. Sound();

?P

Фонетическое изображение языка для мааса  
Animal van ykayat calls us maize dog  
I call on you to buy me no ykayatbaes  
to obtain maize dog!



Dog bobik ("Bobik").

Animal & creature or = & bobik

dynamic - cost < Dog \* > (creature). sounds

Begleitend sound  
by maize dog.

void sound\_three\_times (Animal & creature)

creature.sound()

creature.sound()

creature.sound()

Dog → sound by

animal.

Kann eigentlich  
rallen "Dog" sound  
by Dog? //

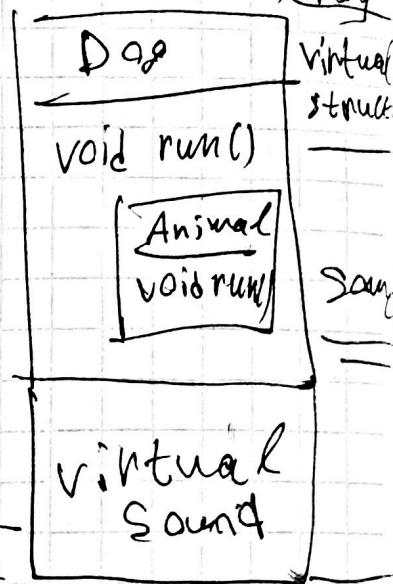
?

=

,

Нужно сделать op - wo в virtual но  
сделать override максимум

Если бы у нас было  
один класс Animal, то виртуальный  
метод run был бы общим для всех  
животных и в Dog мы бы  
нужно было перегрузить.



Виртуальное присущее — назначение  
объекта

! Деструктор дочерних классов будет  
вызываться, если есть наследование.

override — переопределение

final — запрещает изменение наследуемых максимум и переоп.  
op - wo.

virtual Sound () override final  
↳ нельзя переоп.  
↳ нельзя запись

class Dog final : public Animal  
↳ нельзя изменять наследуемых  
унаследованных максимум Dog

## Абстрактное Классы:

Наследование защищается в том, что нет методов, для класса генерируются (методы "переопредел.).  
в будущем когда-нибудь родственник (наследует), но для различных наследников могут быть разные.

Реализация через Super-ре и Animal

```
class Animal {  
public:  
private:  
    String name;  
    int age;  
public:  
    String get_name();  
    virtual void sound() = 0; //  
    // реализация виртуальных  
    // методов в наследнике
```

Абстрактной класс - класс, у которого есть ходы и от них они наследуются.

Онлайн абстрактного класса есть только классы. (такие как `Domestic`, `Wild`).

```
class Animal : public IAnimal {
```

}

---

инициализации наследование:

---

```
class Animal {
```

3

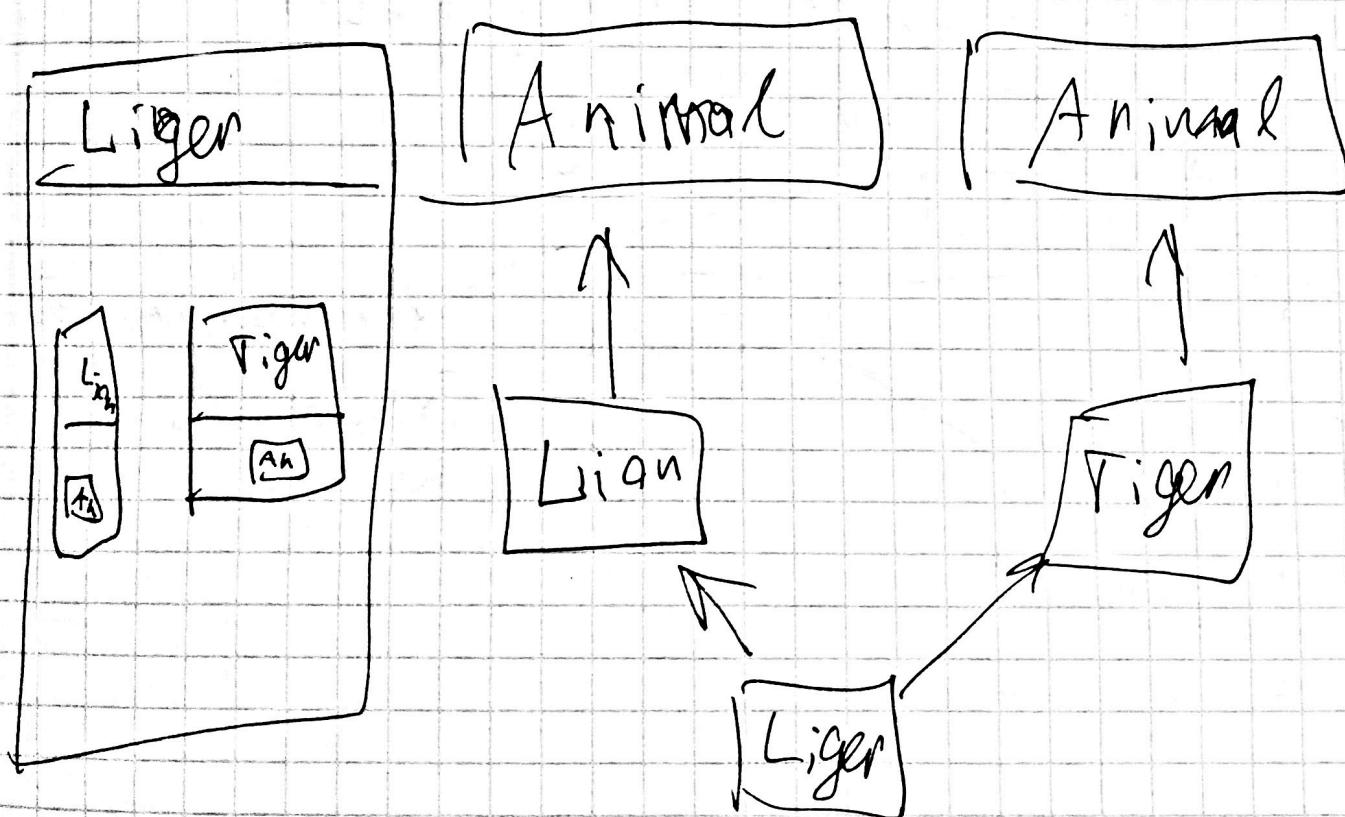
```
class Tiger : public Animal {  
    int tail — length;
```

class Lion : public Animal {  
 int tail-length;

3

class Liger : public Lion, public Tiger

5

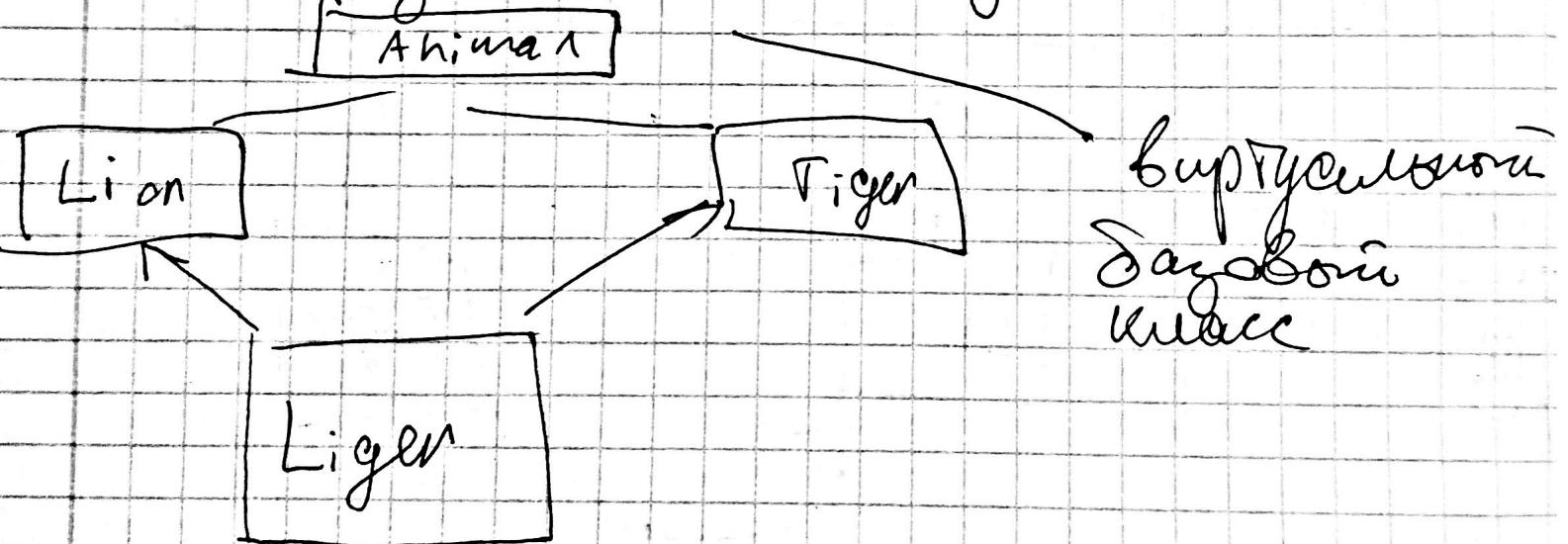


Решение:

- ① Для этого используется Animal  
базовый класс

(2). Дba огненобax наел  
tail-length

Дba наелов бин кирек  
гүчкөр толу, усун ал дбад  
наңызгылыш, т.к. ке дары  
огненобарх наелү н ишегөб  
як көпкөрт. -ын паг-да  
и пагыз алын барды.



```
[ class Animal {  
    string name;  
    int tail-length;
```

```
] class Tiger : virtual public Animal
```

...

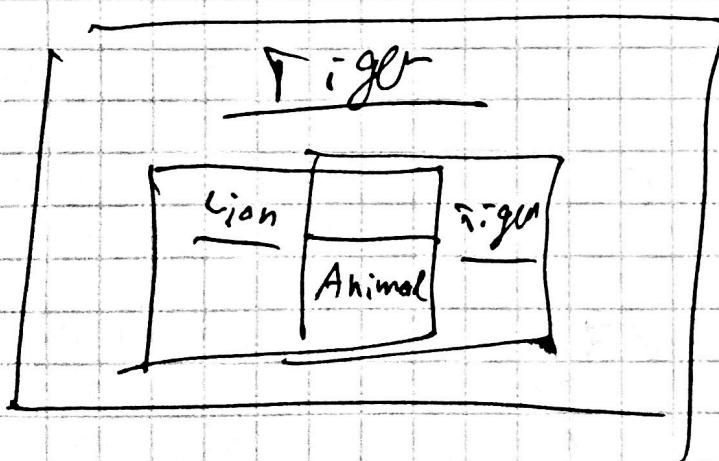
```
[ class Lion : virtual public Animal ]
```

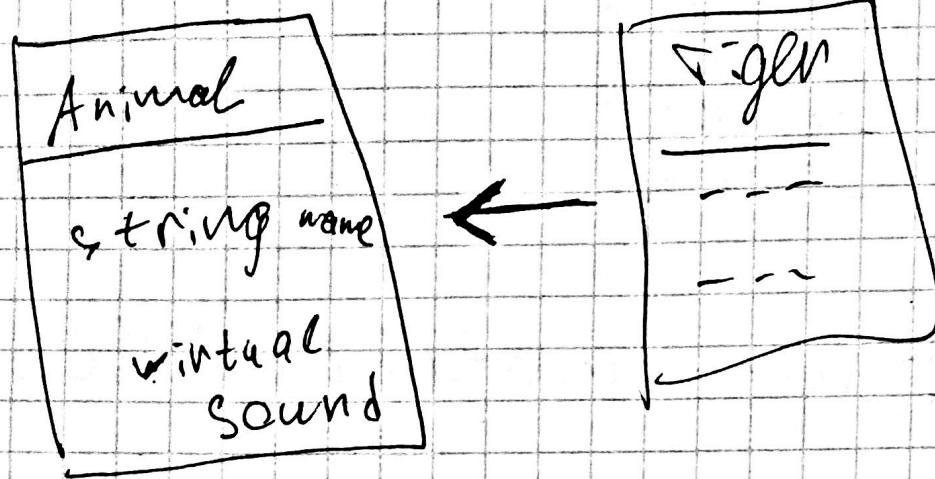
...

?

```
[ class Liger : public Lion, public  
Tiger ]
```

?





Dyncees Benne op-yeu u  
meccor

class Graph {

private:

Node \* Root;

public:

Node \* search (Node \* node);

3

class Node {

private:

Void \* data

/

~~Graph~~

str : : list<Node\* > neighbours

friend class Graph; // friend

{

my class  
Graph derives  
from node

a member class

Node.

e.g. two nodes  
can agree  
balance

friend - friend declaration.

Node is Friend of Graph,

no redeclaration

as Node private member  
Graph recognizes

A - friend of B

B - friend of C

A - is of type C

```

class Graph {
private:
    Node *root;
public:
    Node *search(Node *node);
}

class Node {
private:
    void *data;
    std::list<Node*> neighbours;
friend Node & Graph::Search(
    Node *node,
)

```

### Anonymous objects

Dog ("Bobik"). IT Cozganne  
aukmenoro  
objekta

Dog bobik = Dog ("Bobik").

Dog ("Bobik"). sound()

## Использование:

- ① Инициализация поля на при  
исчезновении
- public  
private  
protected
- ② Порядок вызова конструкторов  
при исчезновении
- ③ Порядок вызова деструкторов при  
исчезновении
- ④ Помимо этого приведение типов  
(к наследству)
- ⑤ Помимо этого приведение типов  
(к родителю)
- ⑥ Срезка объектов (animal=dog)

- ⑦ virtual метод (virtual  
член метода)
- ⑧ деструктивный класс (интегратор)
- ⑨ Многократное наследование  
(баз. класс)
- ⑩ Переопределение методов в  
классе
- ⑪ Анонимные - то - объекты