Наследование

Тема 5

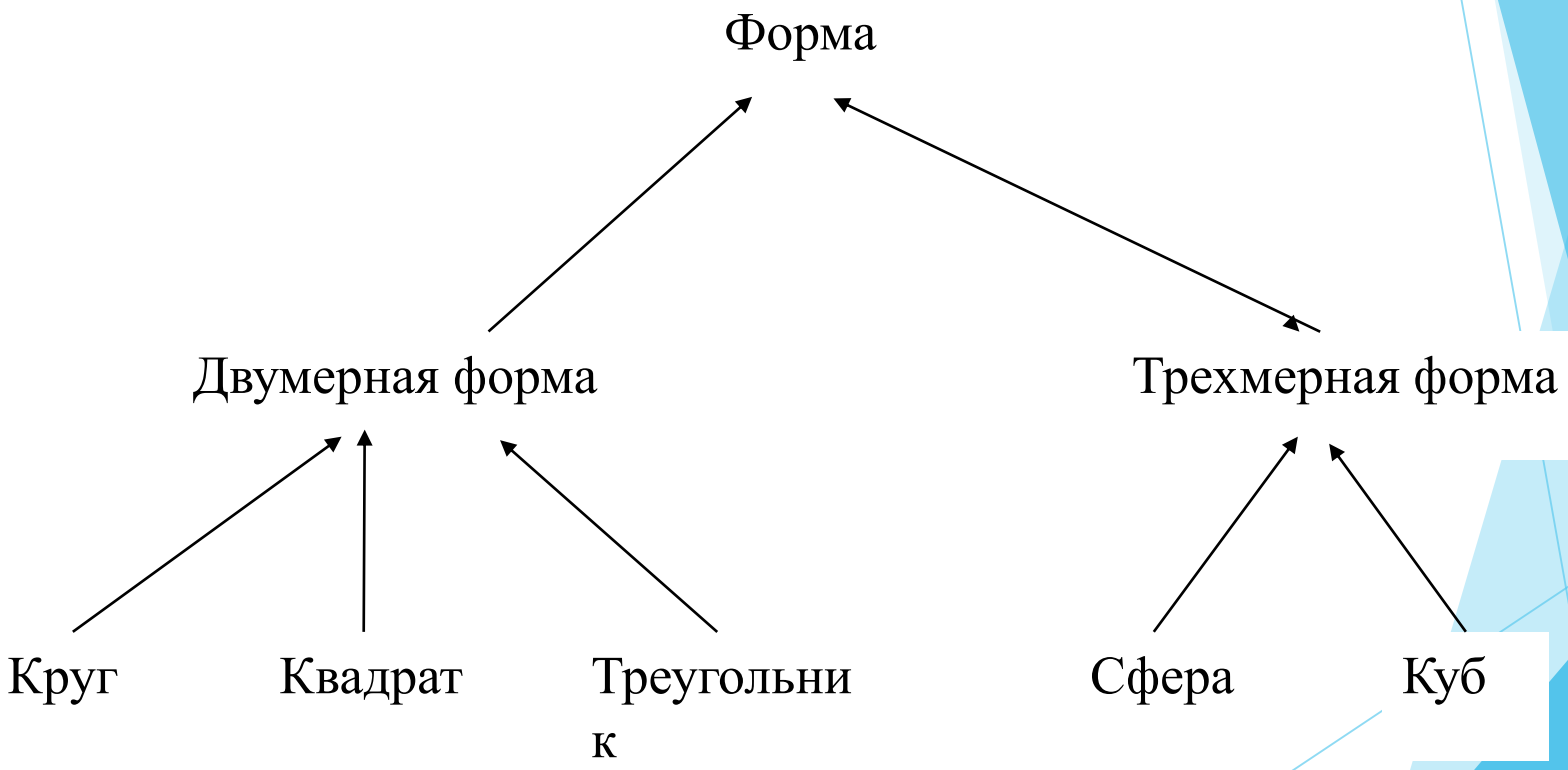
Отношения между классами

- Наследование
- Ассоциация
- Композиция
- Агрегация

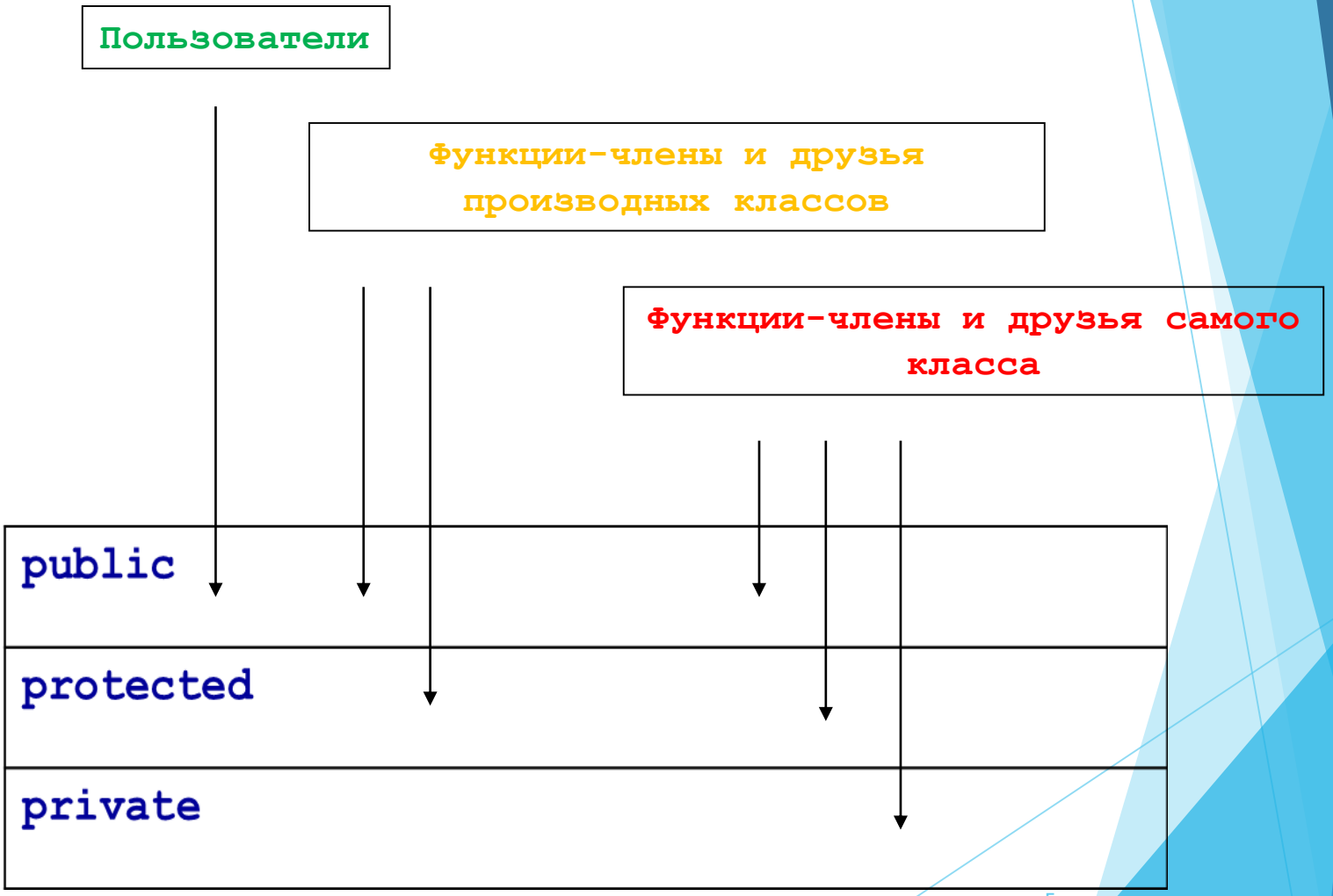
Наследование - это

способ повторного использования кода, при котором новые классы создаются из уже существующих классов путем заимствования их атрибутов и функций и обогащения этими возможностями новых классов

Иерархическая структура наследования



Доступ к полям класса



Типы наследования

- Открытое – наследование интерфейса
- Закрытое – наследование реализации
- Защищенное – наследование особенностей иерархии классов

Влияние типа наследования на доступ к полям базового класса через объект производного класса

Спецификатор доступа к элементам в базовом классе	Тип наследования		
	public открытое наследование	protected защищенное наследование	private закрытое наследование
public	public в производном классе	protected в производном классе	private в производном классе
protected	protected в производном классе	protected в производном классе	private в производном классе
private	невидим в производном классе	невидим в производном классе	невидим в производном классе

Не наследуются:

- конструкторы;
- деструктор;
- оператор присваивания;
- друзья класса.

Учебный пример

Разработать следующую иерархию классов:

Класс точка

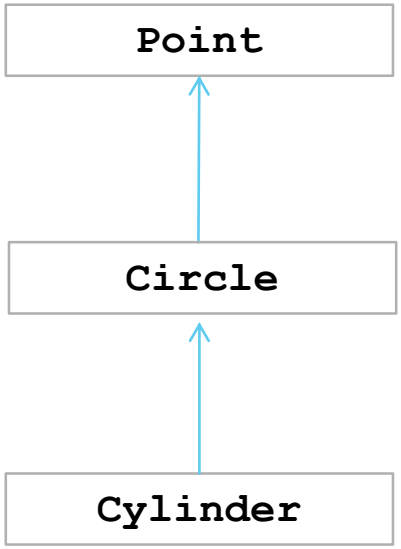
```
class Point
{
public:
    Point(float a =0, float b =0):x(a),y(b) {}
    void SetPoint (float a, float b) {x = a; y = b;}
    float GetX()const {return x;}
    float GetY()const {return y;}
    void Print()const {std::cout << "[" << x << ", " << y
                                 << "]" << std::endl;}
private:
    float x, y;
};
```

Класс круг

```
class Circle : public Point
{
public:
    Circle(float r = 0, float x = 0, float y = 0) :
            Point(x,y), radius(r){}
    void SetRadius(float r) {radius = r;}
    float GetRadius()const {return radius;}
    float Area()const {return 3.14*radius*radius;}
    void Print()const {
        std::cout << "Center = ";
        Point::Print();
        std::cout << "; Radius = " << radius << std::endl;}
private:
    float radius;
};
```

Класс цилиндр

```
class Cylinder : public Circle {
public:
    Cylinder(float h=0, float r= 0, float x= 0, float y= 0):
            Circle(r,x,y), height(h){}
    void SetHeight(float h){height = h;}
    float GetHeight()const {return height;}
    float Area()const
        {return 2 * Circle::Area() + 2 * 3.14 * GetRadius() * height;}
    float Volume()const {return Circle::Area()*height;}
    void Print()const {
        Circle::Print();
        std::cout<<"; Height = "<< height<<std::endl;}
private:
    float height;
};
```




```
Point::Point()
Point::SetPoint()
Point::GetX()
Point::GetY()
Point::Print()
```

```
Circle::Circle()
Circle::SetRadius()
Circle::GetRadius()
Circle::Area()
Circle::Print()
Circle::Point::SetPoint()
Circle::Point::GetX()
Circle::Point::GetY()
Circle::Point::Print()
```

```
Cylinder::Cylinder()
Cylinder::SetHeight()
Cylinder::GetHeight()
Cylinder::Area()
Cylinder::Volume()
Cylinder::Print()
Cylinder::Circle::SetRadius()
Cylinder::Circle::GetRadius()
Cylinder::Circle::Area()
Cylinder::Circle::Print()
Cylinder::Circle::Point::SetPoint()
Cylinder::Circle::Point::GetX()
Cylinder::Circle::Point::GetY()
Cylinder::Circle::Point::Print()
```

```
Point::
x
y
Circle::
Point::x
Point::y
radius
Cylinder::
Circle::Point::x
Circle::Point::y
Circle::radius
height
```

Работа с классами

```
int main(){
    Cylinder cyl(5.7, 2.5, 1.2, 2.3);
    cyl.SetHeight(10);
    cyl.SetRadius(5);
    cyl.SetPoint(2,2);
    std::cout<<"New:"<<std::endl;
    cyl.Print();
    Point &pRef = cyl; //восходящее преобразование
    std::cout<<"As Point:"<<std::endl;
    pRef.Print();
    //cout<<pRef.area()<<endl; недопустимо,
     // так как функция area() отсутствует в базовом классе
    Point p(1,1);
    Point *a = &cyl; //   восходящее преобразование
    //Cylinder &aRef = p; cannot convert from 'class Point'
    // to 'class  Cylinder &'
    return 0;
}
```

Center = [2.00, 2.00];
Radius = 5.00;
Height = 10.00

[2.00, 2.00]

Результат работы программы

Constructor Point

Constructor Circle

Constructor Cylinder

New:

Center = [2.00, 2.00]; Radius = 5.00; Height = 10.00

As Point:

[2.00, 2.00]

Constructor Point

Destructor Point

Destructor Cylinder

Destructor Circle

Destructor Point

Функции-члены

Ошибка!!!

```
void Circle::Print()const
{
    std::cout << "Center = ";
    std::cout << "[" << x << ", " << y << "]" << std::endl;
    std::cout << "; Radius = "<< radius  <<std::endl;
}
```

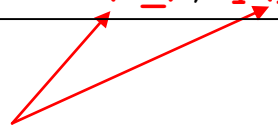
```
void Circle::Print()const
{
    std::cout << "Center = ";
    Point::Print();
    std::cout << "; Radius = " << radius << std::endl;
}
```

Хорошо

Конструкторы

```
Point(float a =0, float b =0):x(a),y(b) {}
```

```
Circle::Circle (float r = 0, float x_ = 0, float y_ = 0):
                    x(x_), y(y_), radius(r){}
```



Ошибка!!!

```
Circle (float r = 0, float x = 0, float y = 0):
                    Point(x,y), radius(r) {}
```

Хорошо

Порядок вызова конструкторов/деструкторов

```
Cylinder * pCyl = new Cylinder(5.7, 2.5, 1.2, 2.3);
```

Constructor Point

Constructor Circle

Constructor Cylinder

```
delete pCyl;
```

Destructor Cylinder

Destructor Circle

Destructor Point

Конец