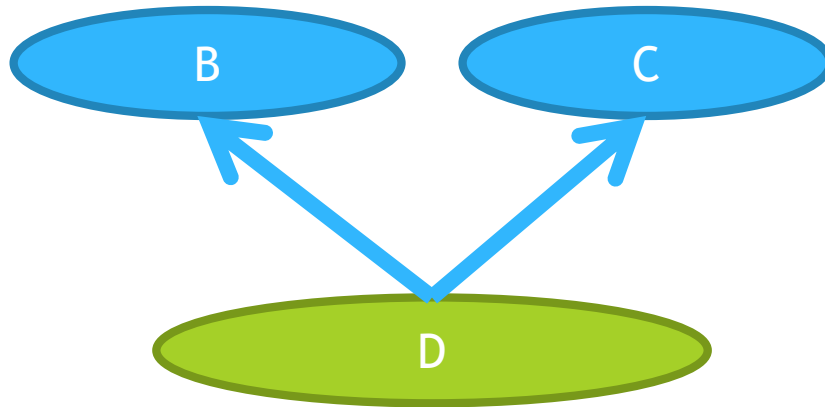


ДИАМАНТЕН ПРОБЛЕМ И ВИРТУАЛНИ КЛАСОВЕ

доц. д-р Нора Ангелова

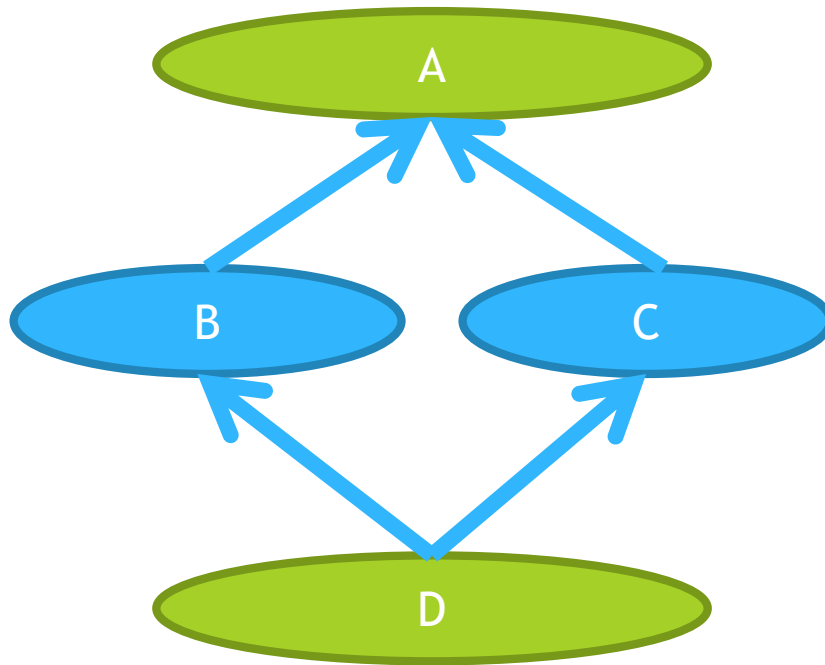
ДИАМАНТЕН ПРОБЛЕМ

- Множествено наследяване



ДИАМАНТЕН ПРОБЛЕМ

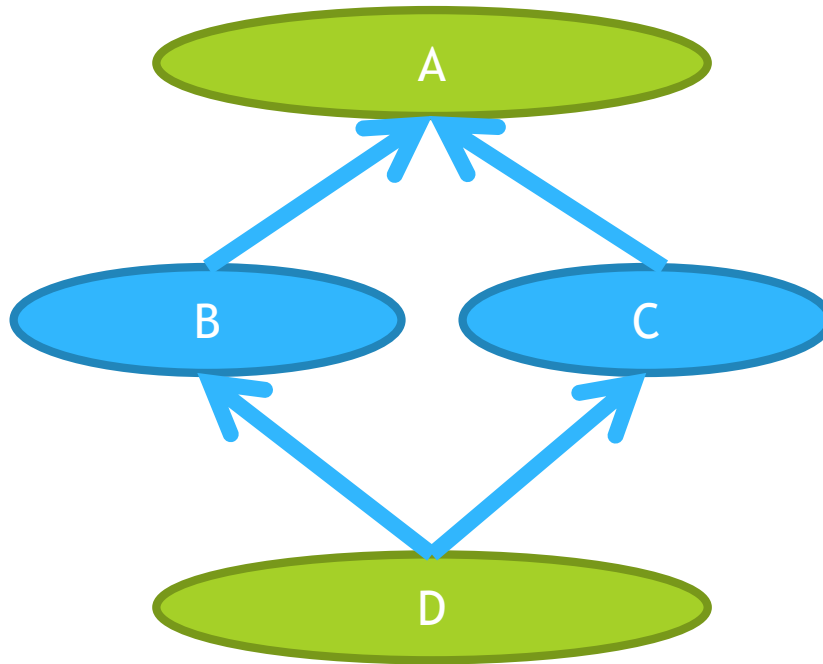
- Ако класовете имат общ основен клас?



ДИАМАНТЕН ПРОБЛЕМ

Проблем:

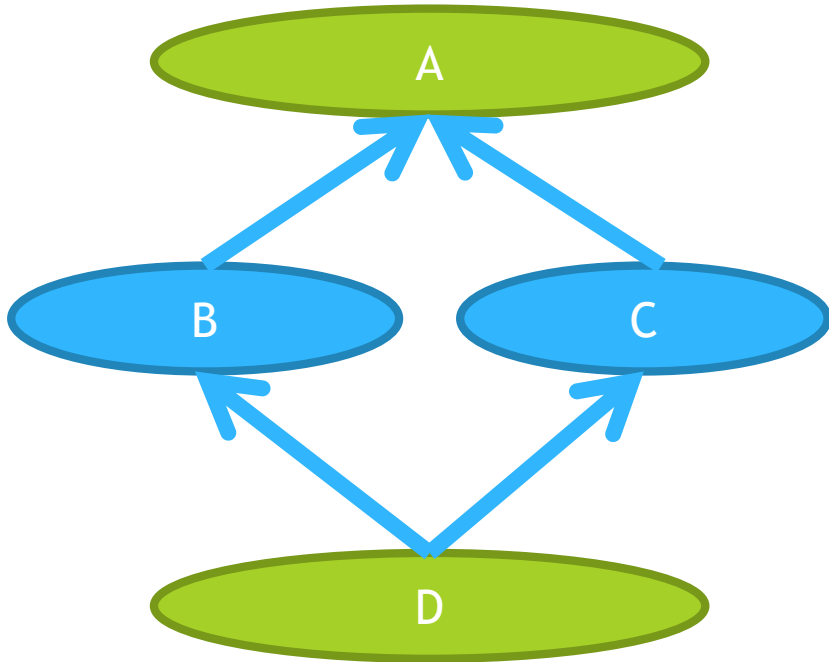
Многократно наследяване на основния клас



ДИАМАНТЕН ПРОБЛЕМ

Проблем:

Многократно наследяване на основен клас - достъп до дублираните компоненти на основния клас



Клас D - собствени
член-данни

клас B - собствени
член-данни

клас B - наследени
член-данни от клас A

клас C - собствени
член-данни

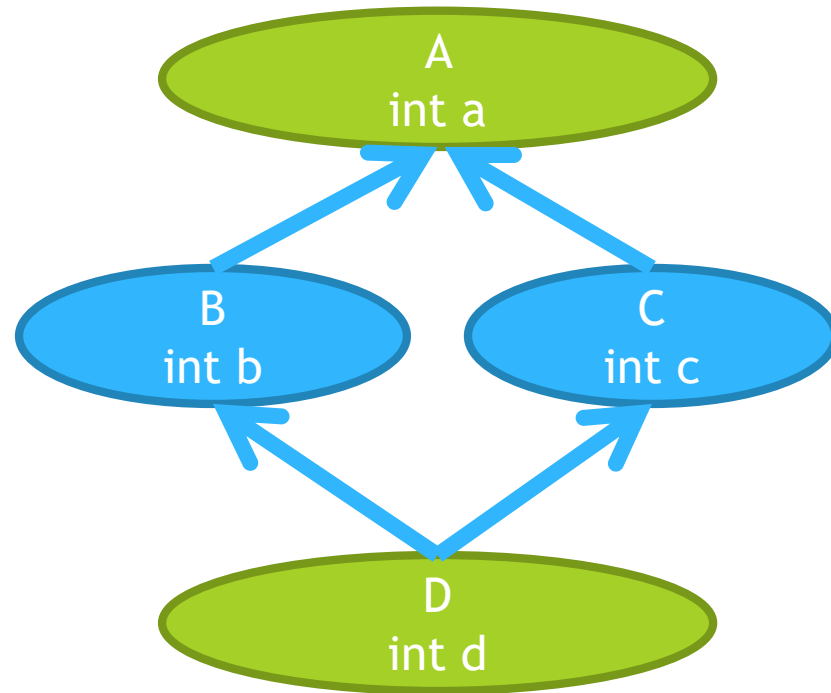
клас C - наследени
член-данни от клас A

ДИАМАНТЕН ПРОБЛЕМ

Проблем:

Многократно наследяване на базов клас - нееднозначност при използване.

```
class A {  
protected:  
    int a;  
public:  
    A(int aVal) {  
        a = aVal;  
    }  
    void print() const;  
};
```



// D има две инстанции на int a, как можем да ги достъпим?!?

// B::a, C::a – валидни извиквания

// A::a – не е валидно извикване

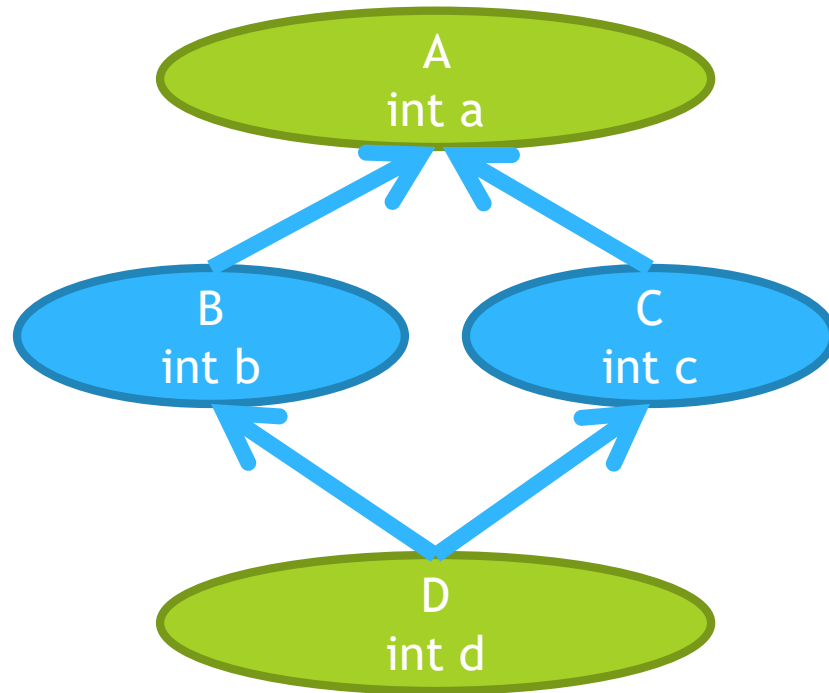
ДИАМАНТЕН ПРОБЛЕМ

Проблем:

Многократно наследяване на базов клас - нееднозначност при използване.

```
class C : public A {  
protected:  
    int c;  
public:  
    C(int aVal, int cVal) : A(aVal) {  
        c = cVal;  
    }  
    // ...  
};
```

```
class B : public A {  
private:  
    int b;  
public:  
    B(int aVal, int bVal) : A(aVal) {  
        b = bVal;  
    }  
    // ...  
};
```



ДИАМАНТЕН ПРОБЛЕМ

Проблем:

Многократно наследяване на базов клас - нееднозначност при използване.

```
//...
```

```
class D : public B, public C {
```

```
protected:
```

```
    int d;
```

```
public:
```

```
    D(int a1Val, int bVal, int a2Val, int cVal, int dVal)
```

```
        : B(a1Val, bVal), C(a2Val, cVal) {
```

```
            d = dVal;
```

```
        }
```

```
//...
```

```
};
```

```
int main() {
```

```
    D obj(1, 2, 3, 4, 5); // а = ? Кое "а" е равно на 1 и кое на 3
```

```
    return 0;
```

```
}
```

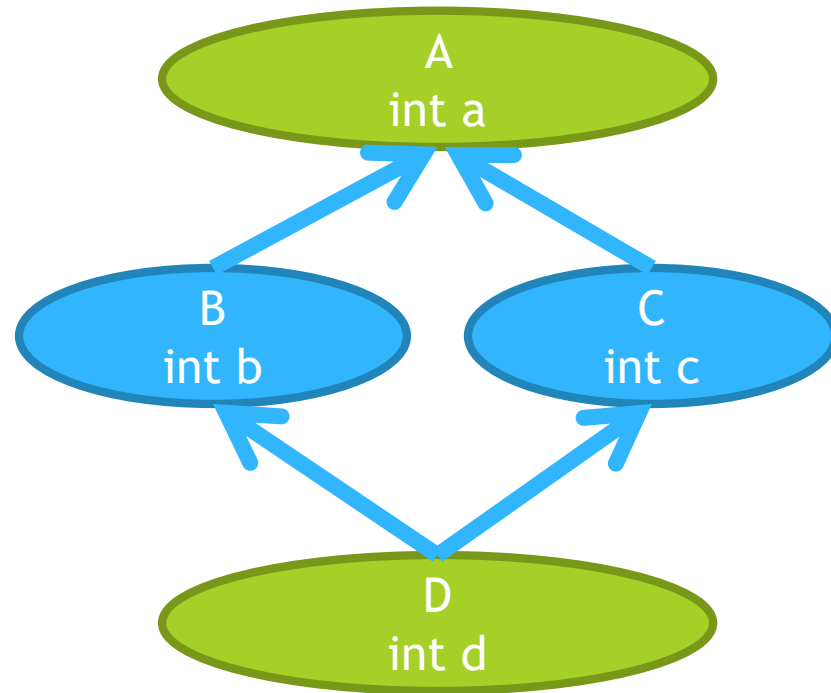

ДИАМАНТЕН ПРОБЛЕМ

Проблем:

Многократно наследяване на базов клас - нееднозначност при използване.

```
class A {
protected:
    int a;
public:
    A(int aVal) {
        a = aVal;
    }
    void print() const;
};
//...

class D : public B, public C {
private:
    int d;
public:
    D(int a1Val, int bVal, int a2Val, int cVal, int dVal)
        : B(a1Val, bVal), C(a2Val, cVal) {
        d = dVal;
    }
    void printD() const {
        A::print(); // Има две функции за извеждане от класа A (this е от тип D *)?
    }
};
```



ДИАМАНТЕН ПРОБЛЕМ

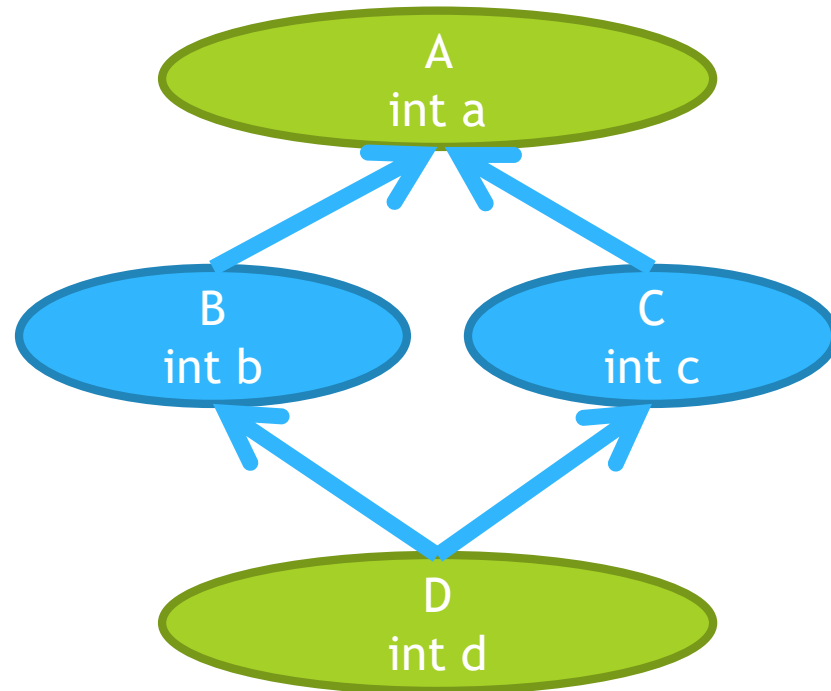
Решение:

Достъпът до „конфликтните“ компоненти на A става чрез последователно прилагане на операцията за явно преобразуване на типове. Атрибутът трябва да е public.

```
class A {  
protected:  
    int a;  
public:  
    A(int aVal) {  
        a = aVal;  
    }  
    void print() const;  
};  
  
class D : public B, public C {  
    //...  
    void printD() const {  
        ((A)(C)*this).print();  
        ((A)(B)*this).print();  
    }  
};  
//...
```

Обект:

```
D dObj(1, 2, 3, 4, 5);  
((A)(C)dObj);  
((A)(B)dObj);
```



ДИАМАНТЕН ПРОБЛЕМ

Решение:

```
D dObj(1, 2, 3, 4, 5);
```

Нека разгледаме:
dObj

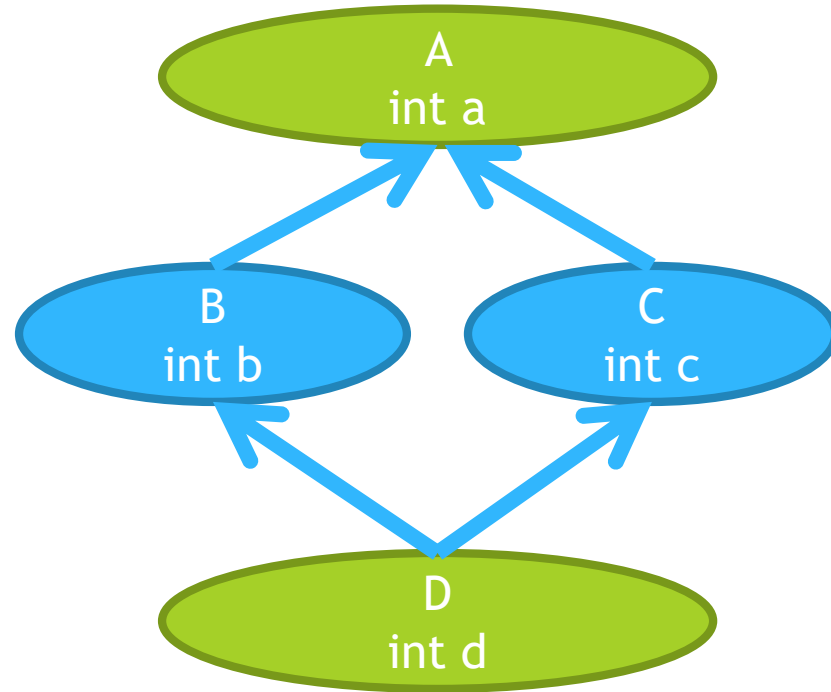
Клас D - собствени
член-данни

клас B - собствени
член-данни

клас B - наследени
член-данни от клас A

клас C - собствени
член-данни

клас C - наследени
член-данни от клас A



ДИАМАНТЕН ПРОБЛЕМ

Решение:

```
D dObj(1, 2, 3, 4, 5);
```

Може да се преобразува до обект на базов клас – B || C:
(B)dObj;

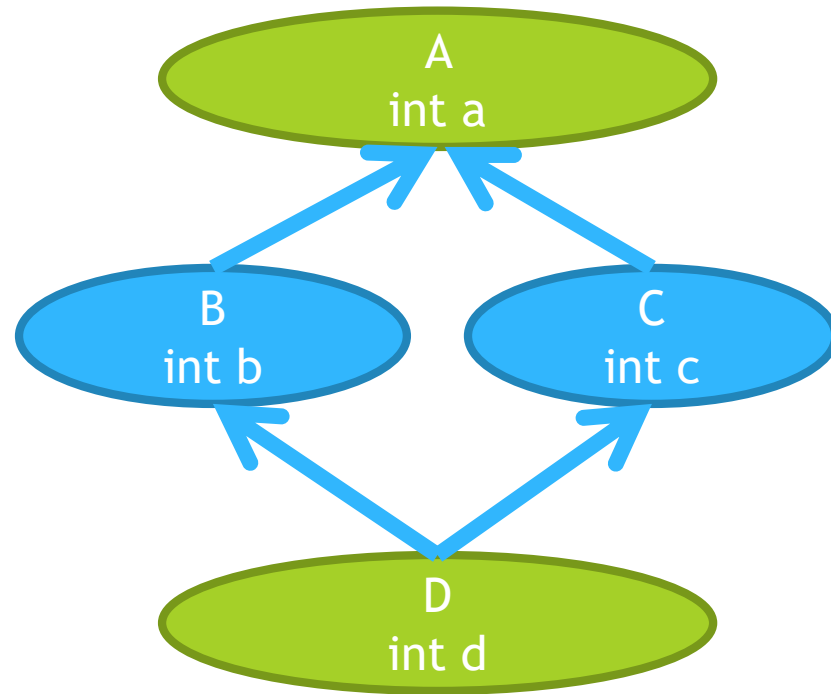
Клас D – собствени
член-данни

клас B - собствени
член-данни

клас B - наследени
член-данни от клас A

клас C – собствени
член-данни

клас C – наследени
член-данни от клас A



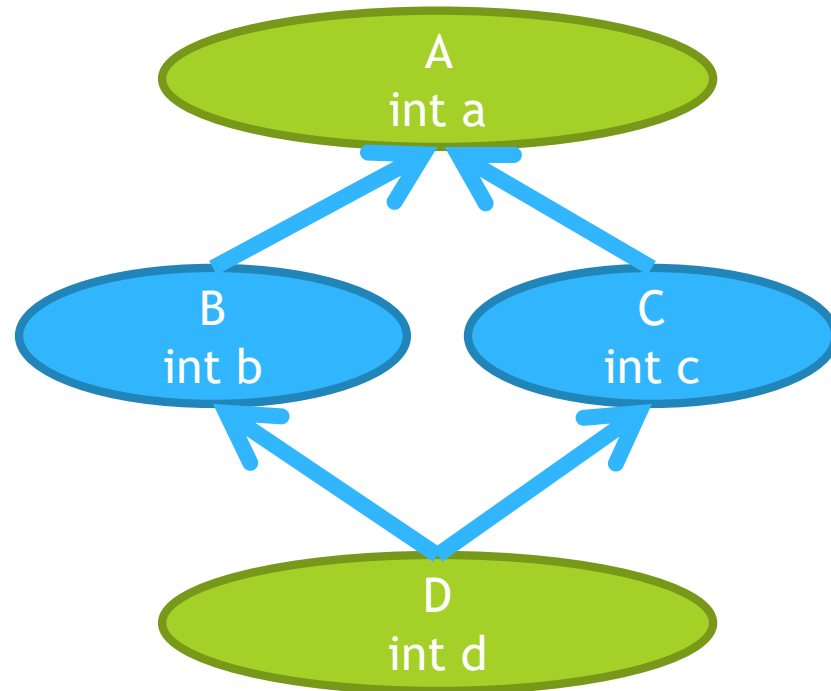
ДИАМАНТЕН ПРОБЛЕМ

Решение:

```
D dObj(1, 2, 3, 4, 5);
```

Може да се преобразува до обект на базов клас – A:

```
(A)(B)dObj; // Съдържа собствените за A компоненти
```



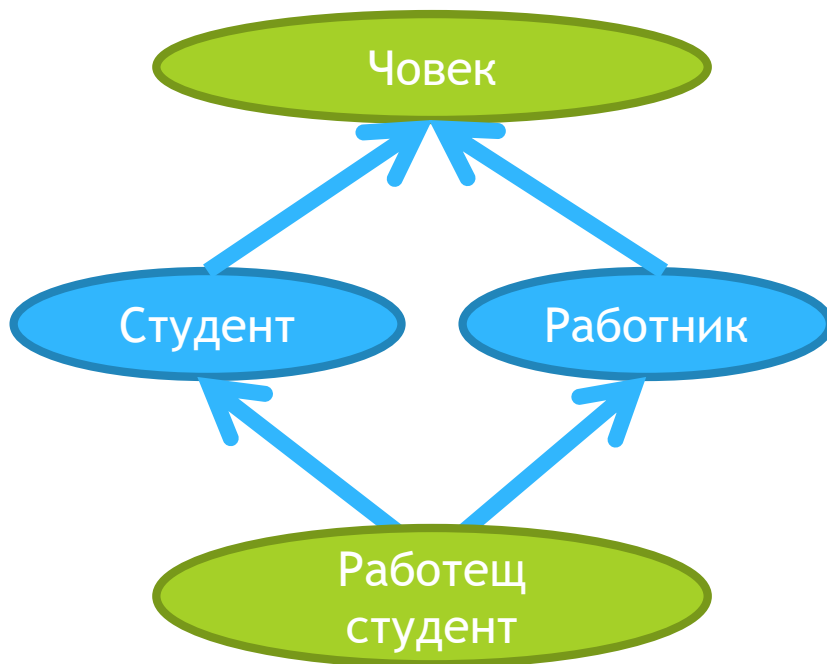
ДИАМАНТЕН ПРОБЛЕМ

Проблем

- Многократно наследяване на базов клас

В класът **Работещ студент** съдържа два пъти компоненти на класа **Човек**. Всеки **Работещ студент** е **Човек**, но има само веднъж тези характеристики.

Искаме те да бъдат наследени само веднъж.



ВИРТУАЛНИ КЛАСОВЕ

Преодоляването на голяма част от недостатъците на многократното наследяване на клас се осъществява чрез използване на т.н. **виртуални основни класове**.

ВИРТУАЛНИ КЛАСОВЕ

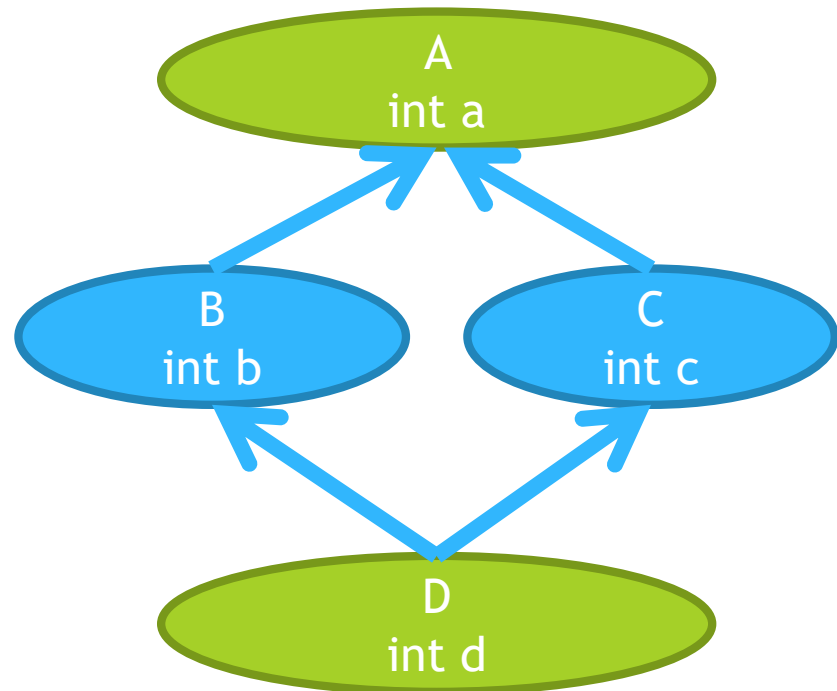
- ⦿ Дават възможност за „поделяне“ на компонентите на основните класове.
- ⦿ Създава се само едно тяхно копие.

ВИРТУАЛНИ КЛАСОВЕ

- Дават възможност за „**поделяне**“ на компонентите на основните класове.
- Създава се **само едно** тяхно копие.

Пример:

В класа D има само едно копие на член-данните на A.

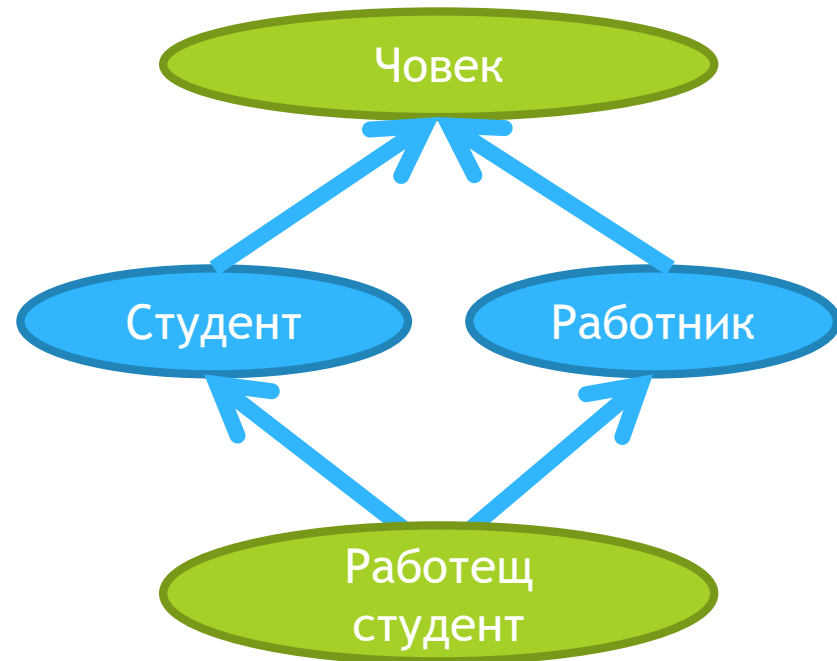


ВИРТУАЛНИ КЛАСОВЕ

- Дават възможност за „поделяне“ на компонентите на основните класове.
- Създава се само едно тяхно копие.

Пример:

В класа „Работещ студент“ има само едно копие на член-данните на „Човек“.



ВИРТУАЛНИ КЛАСОВЕ

- Декларира се като в декларацията на производния клас заедно с името и атрибута за област на основния клас се укаже и ключовата дума `virtual`.


```
class B : virtual public A {  
    // ...  
};
```

```
class C : virtual public A {  
    // ...  
};
```

ВИРТУАЛНИ КЛАСОВЕ

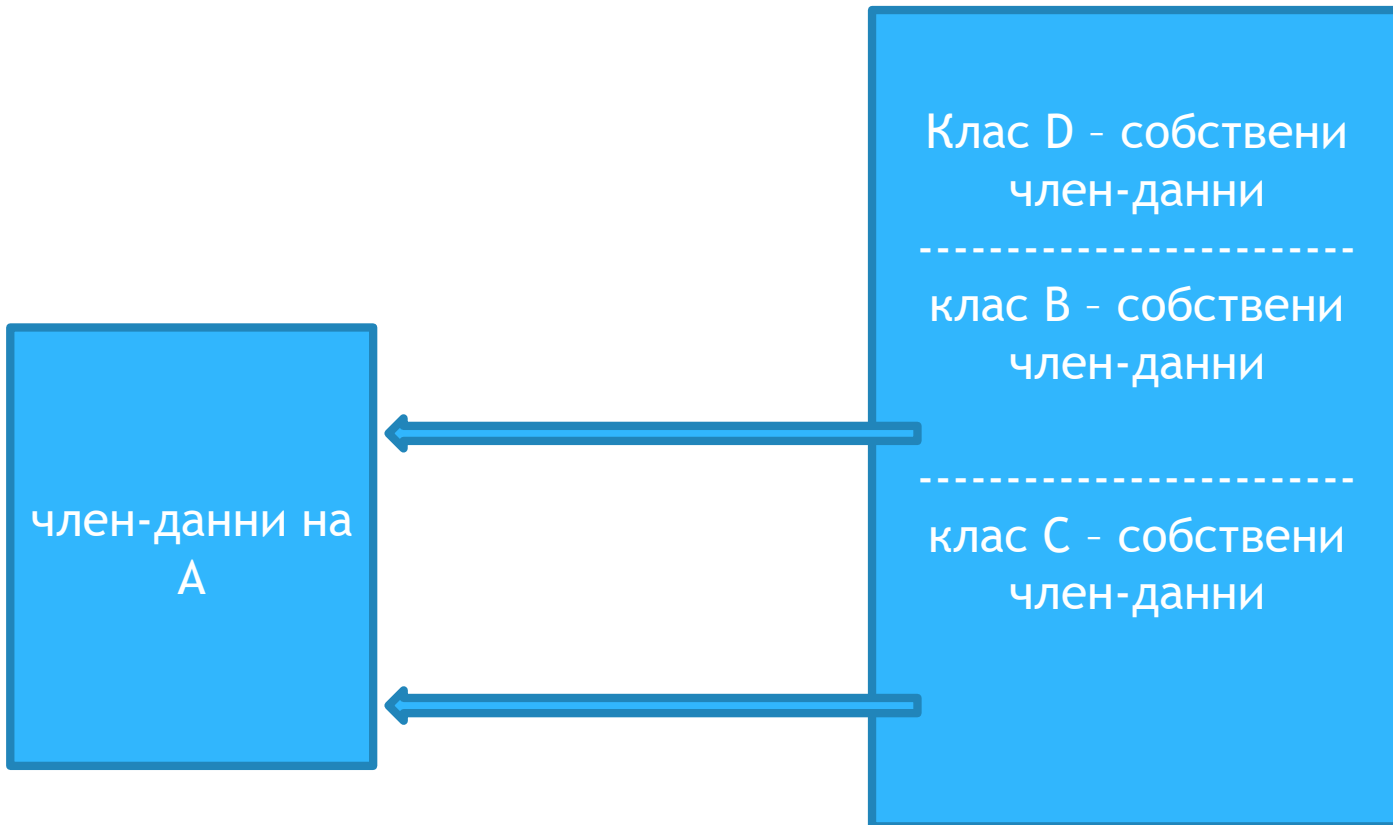
- Конструкторите с параметри на виртуални класове трябва да се извикват от конструкторите на **ВСИЧКИ** класове, които са техни наследници, а не само от конструкторите на преките им наследници.

```
class D : public B, public C {  
    public:  
        D(int aVal, int bVal, int cVal, int dVal)  
            : A(aVal), B(aVal, bVal), C(aVal, cVal) {  
            // ...  
        }  
};
```



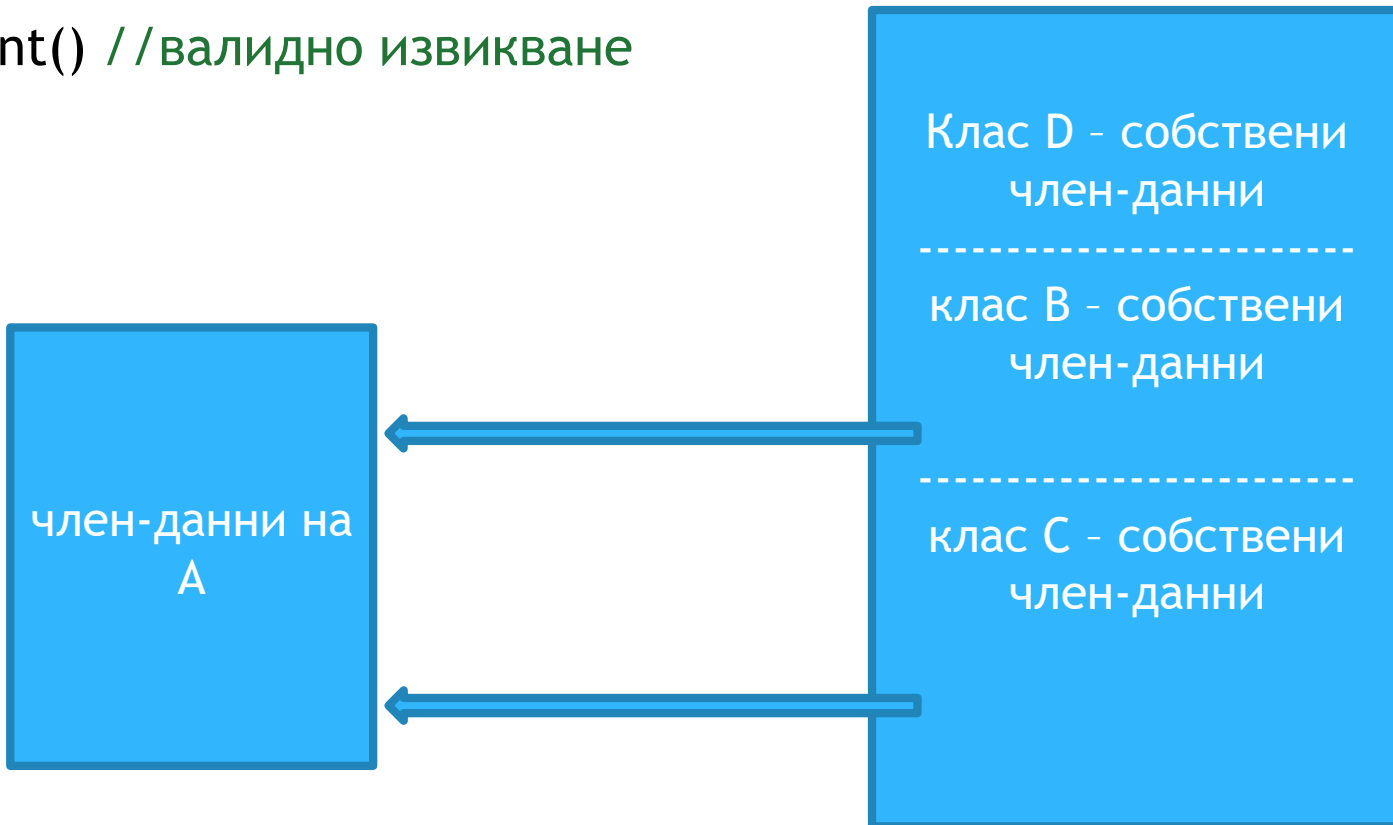
ВИРТУАЛНИ КЛАСОВЕ

- Инициализирането на виртуалните основни класове предхожда инициализирането на другите основни класове. (Първо се извикват техните конструктори)
- Редът на извикване на конструкторите става съгласно реда им в декларацията на производния клас.
- Конструкторът на виртуалния клас се извиква веднъж.



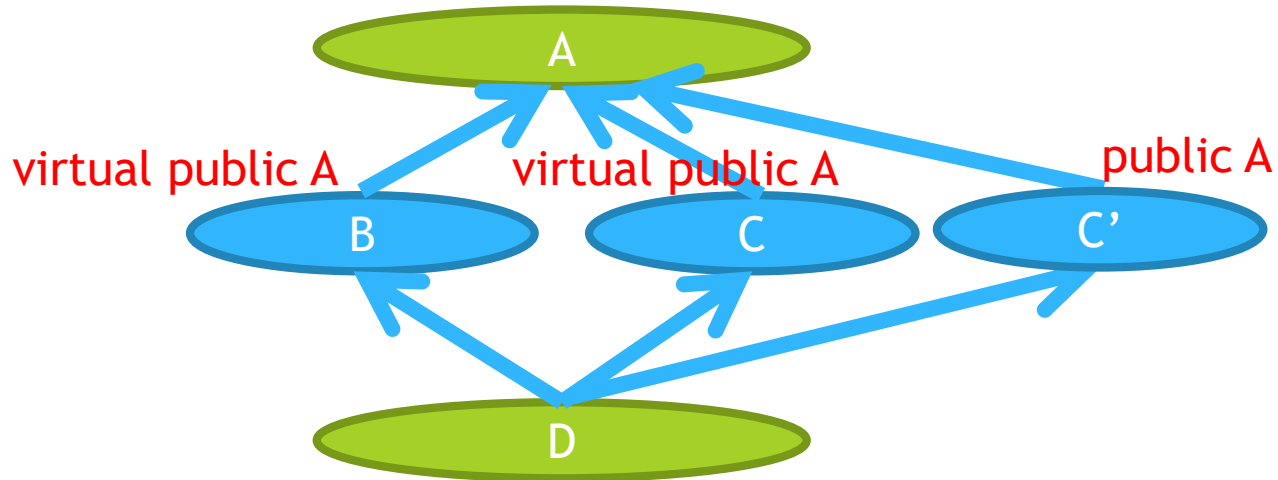
ВИРТУАЛНИ КЛАСОВЕ

- Инициализирането на виртуалните основни класове предхожда инициализирането на другите основни класове. (Първо се извикват техните конструктори)
- Редът на извикване на конструкторите става съгласно реда им в декларацията на производния клас.
- Конструкторът на виртуалния клас се извиква веднъж.
- `A::print()` // валидно извикване



ВИРТУАЛНИ КЛАСОВЕ

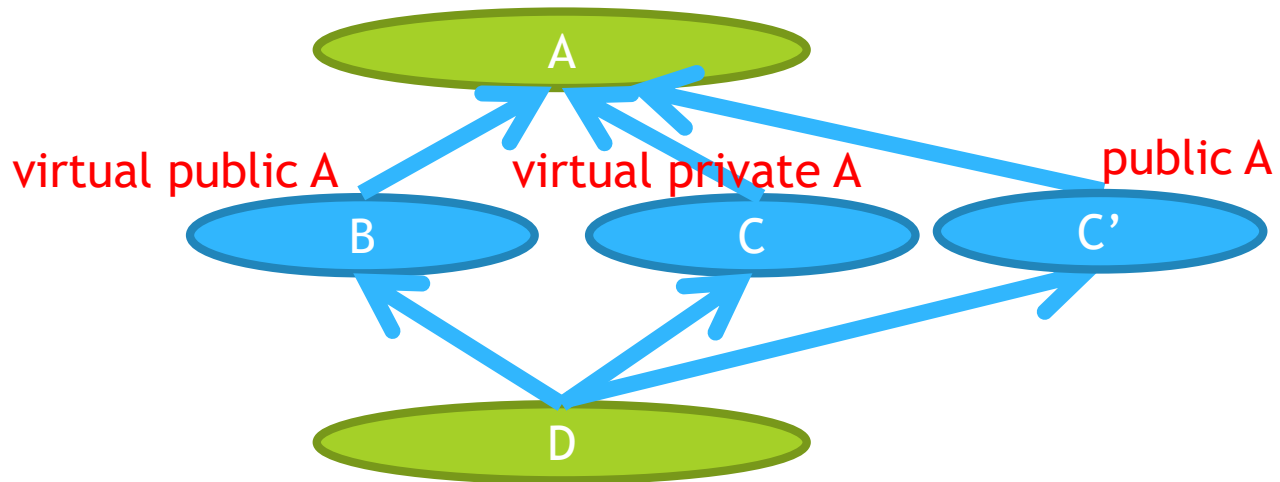
- Инициализирането на виртуалните основни класове предхожда инициализирането на другите основни класове. (Първо се извикват техните конструктори).
- Редът на извикване на конструкторите става съгласно реда им в декларацията на производния клас.
- Конструкторът на виртуалния клас се извиква веднъж.
- `A::print()` // валидно извикване
- Възможно е смесено използване - трябва да се използва преобразуване.



ВИРТУАЛНИ КЛАСОВЕ

- Атрибути за област

Ако в някоя декларация виртуалният клас е обявен като `public` се счита, че той е с атрибут `public` във всички други негови декларации като виртуален основен клас.



ВРЕМЕ
ЗА ВАШИТЕ ВЪПРОСИ