```
// W10_3.cpp : Defines the entry point for the console application.
//             Wielodziedziczenie
//             Klasa wirtualna
//           B                 B
//           !                 !
//           D1                D2
//           !                 !
//                    D3
// Problem polega na tym, ze obiekt klasy D3 bedzie mial dwie kopji klasy B, ktore moga
// zawierac rozne dane. Powstaje niejednoznacznosc.
// Rozwiazanie problemu: zezwolic tyljko jedna kopie klasy B - B - virtualna klasa
// Zwroci uwage na konstruktor klasy D3

#include "stdafx.h"
#include <iostream>
using namespace std;

class B
{
      int i;
public:
      B(int ia) { i = ia; cout << "Konstruktor B\n"; }
      ~B() { cout << "Destruktor B\n"; }
      int get_i() { return i; }
};


//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  virtual
class D1 : virtual public B
//class D1 : public B
{
      int j1;
public:
      D1(int j1a, int ia1): B(ia1) {j1 = j1a;  cout << "Konstruktor D1\n"; }
      ~D1() {cout << "Destruktor D1\n";}
      int get_j1() { return j1; }
};


//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  virtual
class D2 : virtual public B
//class D2 : public B
```

```cpp
{
	int j2;
public:
	D2(int j2a, int ia2) : B(ia2) {j2 = j2a; cout << "Konstruktor D2\n";}
	~D2() { cout << "Destruktor D2\n"; }
	int get_j2() { return j2; }
};

class D3 : public D1, public D2
{
	int j3;
public:
	D3(int j3a, int j1a, int j2a, int ia);
	~D3() { cout << "Destruktor D3\n"; }
	int get_j3() { return j3; }
};

//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
D3::D3(int j3a, int j1a, int j2a, int ia) : D1(j1a, ia), D2(j2a, ia), B(ia)
{
	j3 = j3a;
	cout << "Konstruktor D3\n";
}


int _tmain(int argc, _TCHAR* argv[])
{
	//Teraz obiekt d3 bedzie mial tylko jedna kopie klasy B
	D3 d3(4, 3, 2, 1);

	int B_i = d3.get_i();   //OK - sprawdz pod debugerem stan zmiennych obiektu d3

	system("pause");
	return 0;
}
```