4. Kis zárthelyi, megoldások és pontozás

a kijavított kis zárthelyik megtekinthetők az április 24. gyakorlat után

---

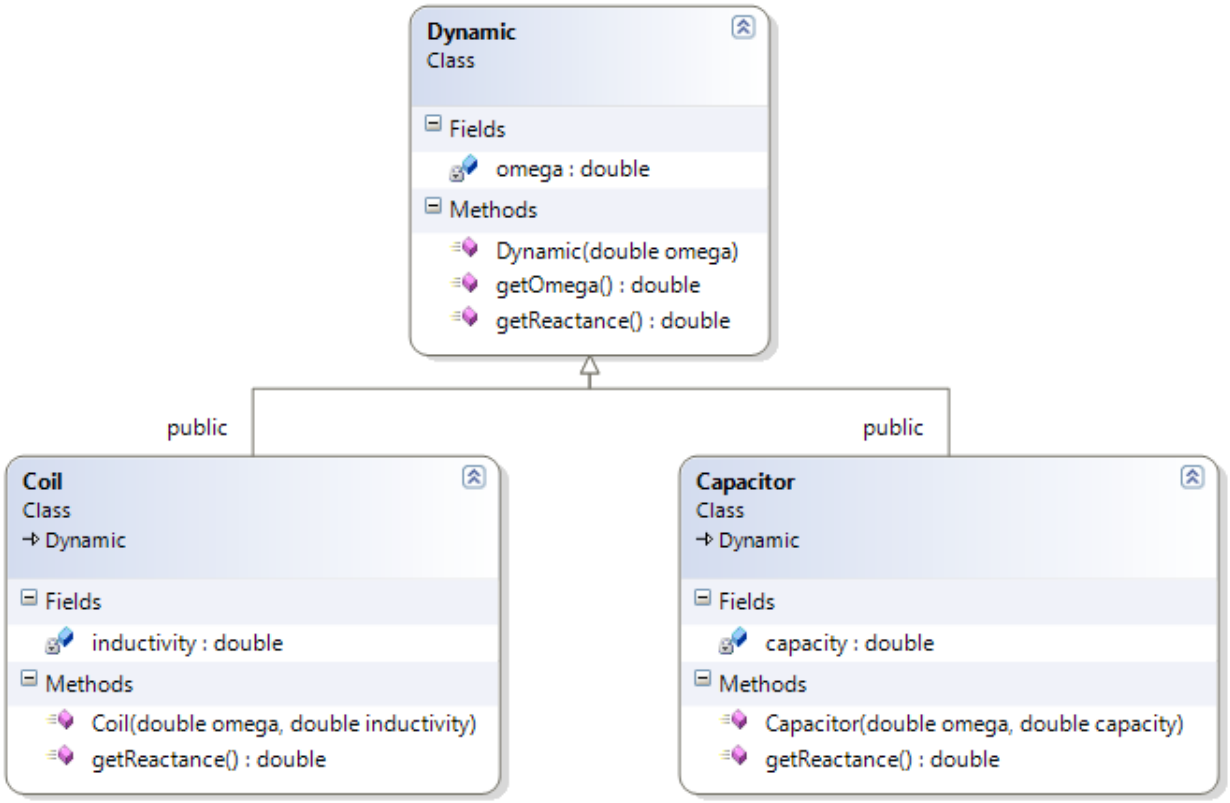
“Egy elektronikai tervező program fejlesztésében önre hárul az a nemes feladat, hogy elkészítse a dinamikus kétpólusokkal kapcsolatos programrészt! Két darab dinamikus alkatrészt kell megvalósítani a kondenzátort (*Capacitor*) és a tekercset (*Coil*). Minden alkatrészre jellemző egy körfrekvencia (omega), és egy olyan függvény mellyel lekérdezhető az ő reaktanciája (*getReactance()*).
A kondenzátornak kapacitása (capacity) van, a tekercsnek pedig induktivitása *(inductivity*). A közös részeket fogja össze egy ősosztályba (*Dynamic*), ami önmagában ne legyen példányosítható. A fő programban hozzon létre néhány dinamikus elemet (használjon heterogén gyűjtést), majd számítsa ki az eredő reaktanciát, feltételezve hogy az összes alkatrész sorosan van egymással bekötve! Mielőtt elkezdené a kód írást, egy rajzon mutassa be az osztályok közötti kapcsolatot!
Segítség:

- tekercs reaktanciája = omega*inductivity
- kondenzátor reaktanciája = -1/(omega*capacity)”

Pontozás:

1 pont: helyes hierarchia (rajzon)
1 pont: helyes öröklés (kódban)
1-1 pont: helyes konstruktorok a leszármazottakban (meghívja az ős konstruktorát)
1 pont: getter az omegához
1 pont: tisztán virtuális tagfüggvény az ősben
1-1 pont: helyes virtuális függvény felüldefiniálás
1 pont: heterogén gyűjtés
1 pont: polimorf hívás
Ha protected-el oldja meg az omegát és mindent jól csinál, akkor getter és őskonstruktor hívás nélkül is jár a megfelelő pont. Ha nem teszi az ősbe az omega-t akkor -2 pont

---




```
#include <iostream>

using namespace std;
class Dynamic{
	double omega;
public:
	Dynamic(double omega):omega(omega){}
	virtual double getReactance()const=0;
	double getOmega()const{
		return omega;
	}
};

class Capacitor: public Dynamic{
	double capacity;
public:
	Capacitor(double omega,double capacity)
		:Dynamic(omega),capacity(capacity){}
	double getReactance()const{
		return -1.0/(getOmega()*capacity);
	}
};
class Coil: public Dynamic{
	double inductivity;
public:
```

```
	Coil(double omega, double inductivity)
		:Dynamic(omega),inductivity(inductivity){}
	double getReactance()const{
		return getOmega()*inductivity;
	}
};

int main()
{
	Dynamic* twoPoles[4];
	twoPoles[0]=new Capacitor(1,2);
	twoPoles[1]=new Capacitor(2,3);
	twoPoles[2]=new Coil(4,5);
	twoPoles[3]=new Coil(6,7);
	double sum=0.0;
	for(int i=0;i<4;++i){
		sum+=twoPoles[i]->getReactance();
		delete twoPoles[i];
	}
	cout<<"Eredo reaktancia: "<<sum<<endl; // 61.33333
	return 0;
}
```

- nem kellett ilyen részletes osztálydiagramm, elegendő az öröklési kapcsolat ábrázolása
- ha az omega protected, akkor nem kell getter függvény