Inheritance, Overriding and Dynamic Binding

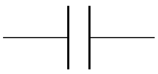
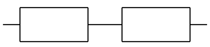
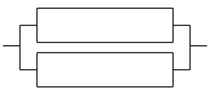
OOP in C++

2017

1 Electric Dipoles

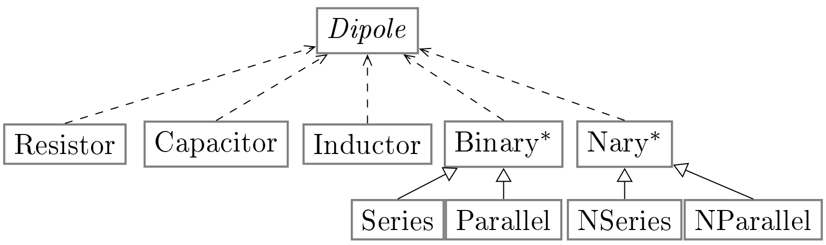
Every electric circuit is composed of differing components such as resistors, capacitors, diodes and electromagnetic coils. They can be assembled in either series or parallel circuits. Depending on their component, each circuit present a specific resistance to the current when a voltage is applied. The **impedance** extends this notion of resistance to alternating currents.

Given ω the angular frequency of the current, the impedance z of the circuit can be computed as follows (with the constant $i = 1\angle\frac{\pi}{2} = e^{j\frac{\pi}{2}}$).

Symbol	*Description*	*Impedance*
r in Ω 	A **resistor** of value r expressed in ohms (noted Ω)	$z = r$
l in H 	An **inductor** of value l expressed in henries (noted H)	$z = i(\omega * l)$
c in F 	A **capacitor** of value c expressed in farad (noted F)	$z = i(\frac{-1}{\omega * c})$
	A **series circuit** with 2 dipoles of impedance z_1 and z_2	$z = z_1 + z_2$
	A **parallel circuit** with 2 dipoles of impedance z_1 and z_2	$z = \frac{1}{\frac{1}{z_1} + \frac{1}{z_2}}$

1.1 Modeling Dipole

We will use the following class hierarchy to model the electric dipoles.

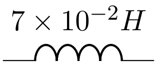


The *Dipole* class provides a method `virtual Complex impedance(double omega)` implemented below in the hierarchy. The parameter `omega` represents the angular frequency of the current. The impedance is a complex number (only resistors have a real impedance).

1.2 Implementing Dipole

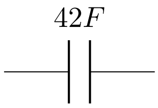
Download the provided code from https://git.io/vS2Ri and save each file your local disk and open them with your favorite editor (or `geany` if you have no favorite editor yet).

▷ **Question 1:** We will first implement the simple dipoles. Create some classes `Resistor`, `Capacitor` and `Inductor`. For each of them, you must create a cpp file and a hpp file, plus a test file. For example for Resistor, you will create the files `Resistor.cpp`, `Resistor.hpp` and `ResistorTest.cpp`. In each class, you just need to implement a `impedance()` method.



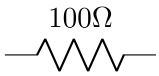
($z \approx 22j\ \Omega$)

Tested Inductor.



($z \approx -7.6 \times 10^{-5} j\ \Omega$)

Tested Capacitor.

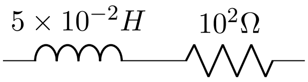


($z = 100\ \Omega$)

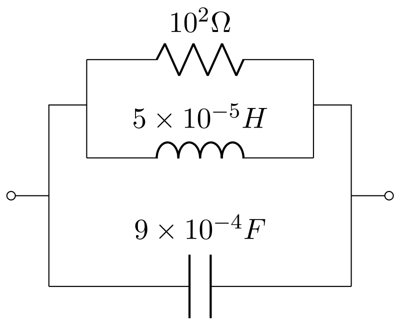
Tested Resistor.

▷ **Question 2:** We will now implement the binary circuits (according to the previous class diagram), either built in parallel or in series. What is the point of adding a `Binary` class? What do parallel and serie circuits have in common that you could factorize in this `Binary` class?

Check your implementation with the following circuits.



Serie ($z \approx 100.0 + 15.70j\ \Omega$).



Parallel ($z \approx 0.2079 + -4.55j\ \Omega$).

▷ **Question 3:** Now, we would like to implement N-ary circuits, that can connect more than 2 dipoles in either parallel or serie. For series circuit: $z = \sum_{i=1}^{n} \omega_i$. For parallel circuit: $z = \dfrac{1}{\sum_{i=1}^{n} \dfrac{1}{\omega_i}}$.

Use a `std::vector` to store the included dipoles.
See http://en.cppreference.com/w/cpp/container/vector for more info.