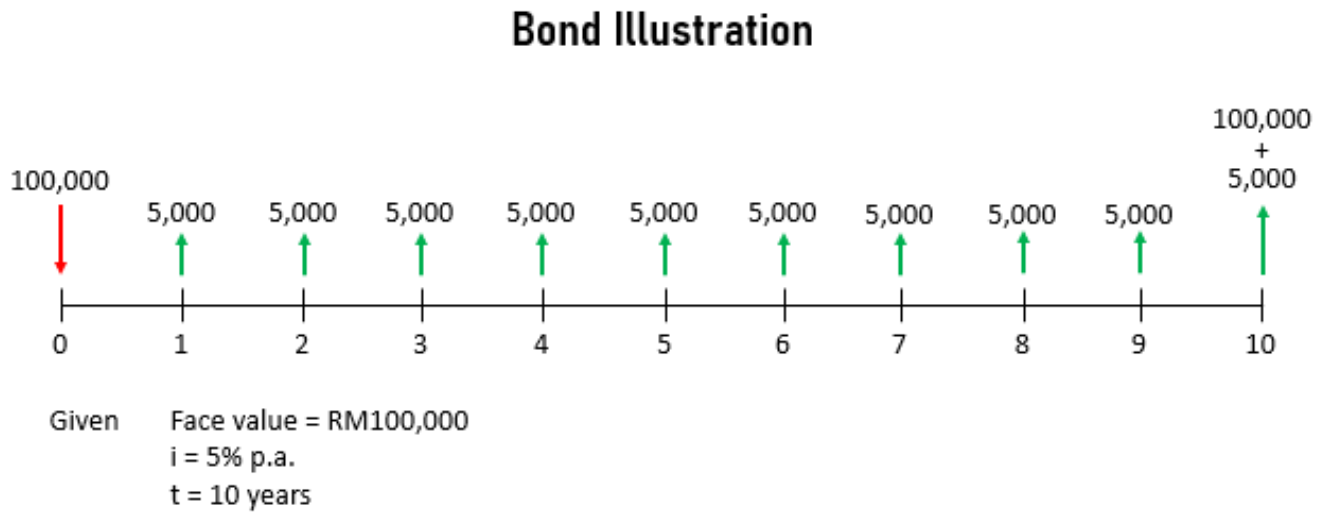
C++ fundamentals with use cases from finance

Tutorial 2: Algorithmic Interest Rate Risk Hedging

Ivan Zhdankin

Bond as Interest rate product

- What is a bond:



- Price of the bond P_m is taken from the market
- We can define yield y (yield-to-maturity) using the price P_m and the following formula:

$$P_m = \frac{C_1}{(1+y)} + \frac{C_2}{(1+y)^2} + \ldots + \frac{C_{10}}{(1+y)^{10}} + \frac{N}{(1+y)^{10}}$$

- In trading we are reasoning in terms of risk, which in case of interest rate products we can measure by $DV01_{bond}$ (dollar value of 1 basis point) with the formula:

$$DV01_{bond} = \frac{\Delta P}{10000 \Delta y}$$

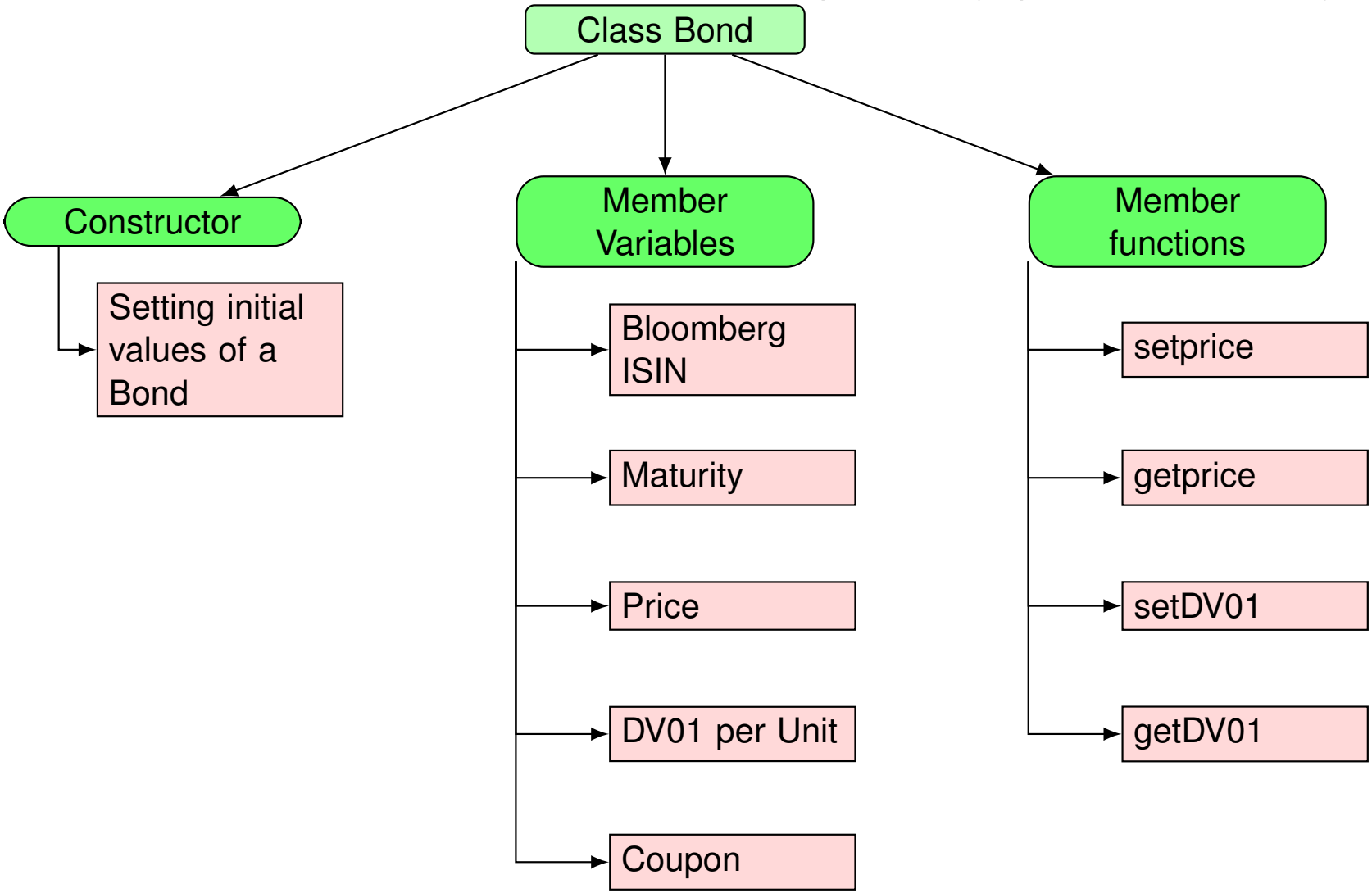
That is, if we change the yield by 1 basis point what is our impact on the price of a bond

Hedging with Bond Future

- Bond future is a future that at settlement date delivers a bond
- Correspondingly we can have the risk $DV01_{futbond}$ when trading bond futures
- We can use bond futures for hedging when trading bonds to get rid of interest rate risk
- The problem is how many bond futures we need to sell when we buy 1mm of a given bond

Part 1/4

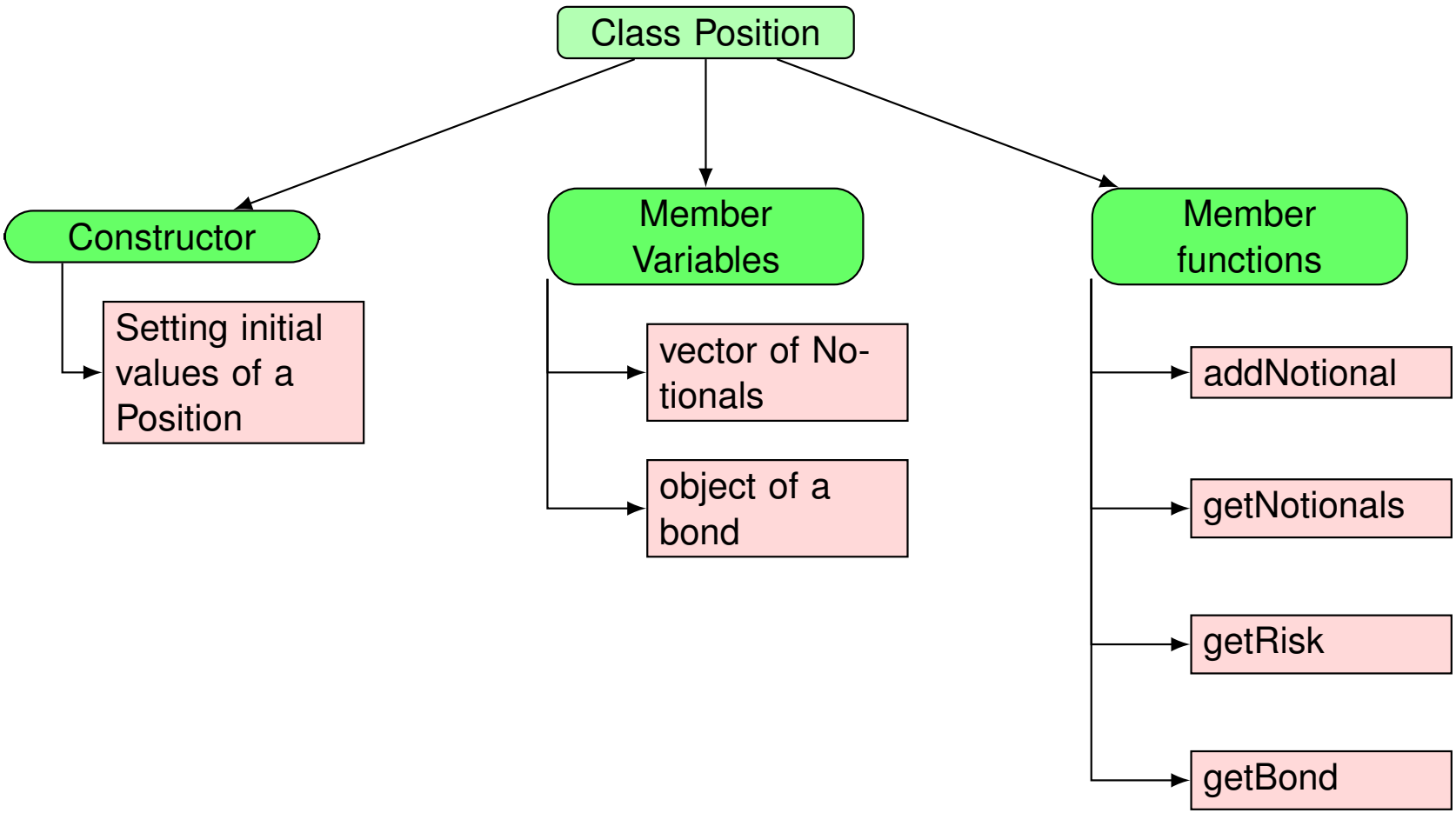
- Define a class “Bond” which contains private variables: maturity of the bond, price, DV01 per unit (mm), coupon, Bloomberg ISIN
- Define Constructor of a Class which assigns private variables
- Define some member functions of the curve class: setprice and getprice, setDV01 and getDV01



Part 2/4

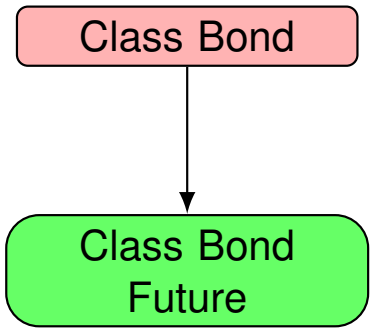
- Define a class “Position” which contains private variables: vector of Nationals, object of bond
- Define Constructor of a Class which assigns private variables
- Define some member functions of the curve class: addNotional, getNotionals, getBond
- Define public function getRisk, that computes total risk of the position as:

$$\sum DV01_{perunit} * Notional$$



Part 3/4

- Define a derived class *BondFuture*, which inherits from *Bond* class
- Define a Constructor of the Class using Contractor of the *Bond* class
- The derived class *BondFuture* should have a member variable *settlementperiod*



Part 4/4

- Define a free-function *calculateNumberOfContracts* that take Position a Bond and Bond Future as arguments and returns (double) number of future contracts we need to have to hedge the position:

$$\sum Notional \frac{DV01_{bond}}{DV01_{futbond}}$$

- In main function test the code:
  - Initialize the object of a bond
  - Initialize the object of a Position
  - Add several bond trades to Position
  - Initialize the object of a bond future
  - Output number of contracts on the screen
- Move all the classes in separate header files