```
namespace LatticeModel
{
    internal interface IGetPrice
    {
        double GetPrice();
    }
    internal class BinomialPricer:IGetPrice
    {
        internal BinomialPricer(IGetPayoff payoff, SDE sde, double[] spotPrices)
        {
            this.sde = sde;
            this.payoff = payoff;
            this.spotPrices = spotPrices;
        }

        public double GetPrice()
        {
            int nodes = spotPrices.Length;  // number of timesteps + 1
            int timeSteps = nodes-2;
            double upProb = sde.GetUpProbability();
            double downProb = sde.GetDownProbability();
            double instDRate = sde.GetInstDRate();
            double strike = sde.Strike;

            double[] tempOptionPrices = new double[nodes-1];
            double[] firstOptionPrices = new double[nodes];

            // calculate option's prices based on terminal simulated prices of the
underlying
            // and the specified payoff function
            for(int i = 0; i< nodes; i++)
            {
                firstOptionPrices[i] = payoff.GetPayoff(strike, spotPrices[i]);
            }

            nodes = nodes -1;
            // use binomila model to work backwards to t0 using the risk neutral
pricing formula and the first option prices
            // reuse the firstoptionprices array
            for (int i = timeSteps; i>=0; i--)
            {
                for (int j=0; j<nodes; j++)
                {
                    tempOptionPrices[j] = instDRate*(upProb*firstOptionPrices[j] +
downProb*firstOptionPrices[j + 1]);
                    firstOptionPrices[j] = tempOptionPrices[j]; // reuse
firstoptionprices array for next iteration
                }
                nodes--;
            }

                return tempOptionPrices[0];
        }

        private SDE sde;
        private IGetPayoff payoff;
        private double[] spotPrices;
    }
}
```