```
using System;

namespace LatticeModel
{
    internal enum SDEType
    {
        JR, CRR, RiskNeutral
    }

    internal interface ISDEMethod
    {
        double GetUpFactor();
        double GetDownFactor();
        double GetUpProbability();
        double GetDownProbability();
        double GetInstDRate();
    }

    internal class SDE:ISDEMethod
    {
       internal SDE(double rate, double sigma, double timeSteps, double strike, double
timeToMaturity, SDEType type)
        {
           this.sigma = sigma;
           this.rate = rate;
           this.timeToMaturity = timeToMaturity;  // time to maturity in months
           if (timeSteps > 0)
           {
               this.deltaT = (timeToMaturity / (12.0*timeSteps));
           }
           else
           {
               throw new System.InvalidOperationException("Invalid time-steps
provided.");
           }
           this.sqrtDeltaT = System.Math.Sqrt(deltaT);
           this.K = strike;
           this.type = type;
           this.timeToMaturity = timeToMaturity;
        }

       internal double Strike
       {
           get
           {
               return this.K;
           }
       }

        public double GetUpFactor()
        {
            double upFactor = 0;
            switch (type)
            {
                case SDEType.JR:

                    upFactor = System.Math.Exp((rate - 0.5 * sigma * sigma) * deltaT +
sigma * sqrtDeltaT);
                    break;

                case SDEType.CRR:
```

```
                    upFactor = System.Math.Exp(sigma * sqrtDeltaT);
                    break;

                case SDEType.RiskNeutral:

                    upFactor = System.Math.Exp(sigma * sqrtDeltaT);
                    break;

                default:
                    Console.WriteLine("Unknown SDE type");
                    upFactor = 0;
                    break;

            }
            return upFactor;
        }

        public double GetDownFactor()
        {
            double downFactor = 0;
            switch (type)
            {
                case SDEType.JR:

                    downFactor = System.Math.Exp((rate - 0.5 * sigma * sigma) * deltaT
- sigma * sqrtDeltaT);
                    break;

                case SDEType.CRR:

                    downFactor = System.Math.Exp(-1 * sigma * sqrtDeltaT);
                    break;

                case SDEType.RiskNeutral:

                    downFactor = System.Math.Exp(-1 * sigma * sqrtDeltaT);
                    break;

                default:
                    Console.WriteLine("Unknown SDE type");
                    downFactor = 0;
                    break;

            }
            return downFactor;
        }

        public double GetUpProbability()
        {
            double upProb = 0;
            switch (type)
            {
                case SDEType.JR:

                    upProb = 0.5;
                    break;

                case SDEType.CRR:

                    upProb = 0.5 + ((rate - 0.5 * sigma * sigma) / (2 * sigma)) *
sqrtDeltaT;
                    break;
```

```
                case SDEType.RiskNeutral:

                    upProb = (System.Math.Exp(rate * deltaT) - System.Math.Exp(-1 *
sigma * sqrtDeltaT)) /
                             (System.Math.Exp(sigma * sqrtDeltaT) - System.Math.Exp(-1
* sigma * sqrtDeltaT));
                    break;

                default:
                    Console.WriteLine("Unknown SDE type");
                    upProb = 0;
                    break;

            }
            return upProb;
        }

        public double GetDownProbability()
        {
            double downProb = 0;
            switch (type)
            {
                case SDEType.JR:

                    downProb = 0.5;
                    break;

                case SDEType.CRR:

                    downProb = 1.0 - (0.5 + ((rate - 0.5 * sigma * sigma) / (2 *
sigma)) * sqrtDeltaT);
                    break;

                case SDEType.RiskNeutral:

                    downProb = 1.0 - ((System.Math.Exp(rate * deltaT) -
System.Math.Exp(-1 * sigma * sqrtDeltaT)) /
                                      (System.Math.Exp(sigma * sqrtDeltaT) -
System.Math.Exp(-1 * sigma * sqrtDeltaT)));
                    break;

                default:
                    Console.WriteLine("Unknown SDE type");
                    downProb = 0;
                    break;

            }
            return downProb;
        }

        // returns single period ("instanteneous") discount factor
        public double GetInstDRate()
        {
            return System.Math.Exp(-1.0 * rate * deltaT);
        }

       private double sigma;
       private double rate;
       private double deltaT;
       private double sqrtDeltaT;
       private double K;
```

```
        private double timeToMaturity;
        private SDEType type;
    }
}
```