1. Include Necessary Libraries

```
#include <Trade/Trade.mqh>
CTrade Trader;
```

Purpose:

- Includes the `Trade.mqh` library to access the `CTrade` class, which simplifies order placement and trade management.
- Declares an instance of `CTrade` called `Trader` for executing trades.

2. Declare Inputs and Variables

```
int handleStoch;

// The Stochastic Oscillator
input int StochKPeriod = 5;
input int StochDPeriod = 3;
input int StochSlowing = 3;
input ENUM_MA_METHOD StochMethod = MODE_SMA;
input ENUM_STO_PRICE StochPriceLevel = STO_CLOSECLOSE;

input double lot_size;
int barsTotal;
int bars;
bool trade = true;
input int Position_Limiter;
```

Purpose:

- Defines the parameters for the Stochastic Oscillator:
  - `StochKPeriod`: The %K period (fast line) length.
  - `StochDPeriod`: The %D period (slow line) length.
  - `StochSlowing`: The slowing factor for smoothing.
  - `StochMethod`: The moving average method (SMA by default).
  - `StochPriceLevel`: The price level used (close/close by default).
- Other inputs:
  - `lot_size`: Lot size for trades.
  - `Position_Limiter`: The maximum number of positions allowed.
- Variables:
  - `barsTotal`: Tracks the last bar index processed.
  - `bars`: Stores the current bar index.
  - `trade`: Boolean to enable or disable trading based on position limits.

3. Trade Opener Function

```
//  A FUNCTION THAT OPENS THE TRADE
void Trade_Opener(){

   double main[];
   double signal[];

   CopyBuffer(handleStoch,MAIN_LINE,0,1,main);
   CopyBuffer(handleStoch,SIGNAL_LINE,0,1,signal);

   if(barsTotal == bars && trade){
      if(main[0] < 20 && signal[0] < 20 && main[0] > signal[0]){
         Trader.Buy(lot_size);
      }
      else if(main[0] > 80 && signal[0] > 80 && main[0] < signal[0]){
         Trader.Sell(lot_size);
      }
   }
}
```

Purpose:

- Fetches the Stochastic Oscillator's %K (`MAIN_LINE`) and %D (`SIGNAL_LINE`) values using `CopyBuffer`.
- Checks conditions for overbought and oversold levels:
  - Buy: If both lines are below 20 (oversold) and %K crosses above %D.
  - Sell: If both lines are above 80 (overbought) and %K crosses below %D.
- Places a trade only if the current bar hasn't already been processed.

4. Trade Closer Function

```
// A FUNCTION THAT CLOSES THE TRADE
void Trade_Closer(){

   double main[];
   double signal[];

   CopyBuffer(handleStoch,MAIN_LINE,0,1,main);
   CopyBuffer(handleStoch,SIGNAL_LINE,0,1,signal);


   if(main[0] <= 50 && signal[0] <= 50 && main[0] > signal[0]){
      for(int i = PositionsTotal() - 1; i >= 0; i--){
         ulong positionTicket = PositionGetTicket(i);
         if(PositionSelectByTicket(positionTicket)){
            if(PositionGetInteger(POSITION_TYPE) == POSITION_TYPE_SELL){
               Trader.PositionClose(positionTicket);
            }
         }
      }
   }
```

```
   else if(main[0] >= 50 && signal[0] >= 50 && main[0] < signal[0]){
      for(int i = PositionsTotal() - 1; i >= 0; i--){
         ulong positionTicket = PositionGetTicket(i);
         if(PositionSelectByTicket(positionTicket)){
            if(PositionGetInteger(POSITION_TYPE) == POSITION_TYPE_BUY){
               Trader.PositionClose(positionTicket);
            }
         }
      }
   }
}
```

Purpose:

- Closes trades when the Stochastic Oscillator reaches neutral levels (around 50):
  - Closes SELL positions when %K crosses above %D while both are ≤ 50.
  - Closes BUY positions when %K crosses below %D while both are ≥ 50.

5. Trade Limiter Function

```
// A LIMIT TO THE NUMBER OF TRADES OPENED/OPENING
void Trade_Limiter(){

   int TradeLimit = PositionsTotal();

   if(TradeLimit == Position_Limiter){
      trade = false;
   }
   else{
      trade = true;
   }
}
```

Purpose:

- Ensures the EA doesn't exceed the specified maximum number of open positions (`Position_Limiter`).
- Toggles `trade` to enable or disable new trade openings.

6. OnInit Function

```
// THE ONINIT FUNCTION
int OnInit(){

   handleStoch = iStochastic(Symbol(),PERIOD_CURRENT, StochKPeriod, StochDPeriod, StochSlowing, StochMethod, StochPriceLevel);

   return(INIT_SUCCEEDED);
}
```

Purpose:

- Initializes the Stochastic Oscillator indicator handle (`handleStoch`) with the specified parameters.
- Validates the handle and reports an error if initialization fails.

7. OnTick Function

```
// THE ONTICK FUNCTION
void OnTick(){

   bars = iBars(Symbol(), PERIOD_CURRENT);
   if(barsTotal == bars) return;
   barsTotal = bars;

// THE TRADE FUNCTIONS ARE CALLED HERE
   Trade_Opener();
   Trade_Limiter();
   Trade_Closer();
}
```

Purpose:

- Executes the trading logic on every new tick:
  - Updates the bar index to avoid processing the same bar multiple times.
  - Calls the trade-opening, limiting, and closing functions.