public int evaluate()
{
    double material = evaluateMaterial();
    int result = -1;
    if((material - 5 >= 0) || (material + 5 <= 0))
    {
        // just counter material
        if(material > 0)
            result = whiteWin;
        else if(material < 0)
            result = blackWin;
        else
            result = equal;
    }
    else // the result isn't great enough to tell alone
    {
        int whiteScore = evaluateWhite();
        int blackScore = evaluateBlack();
        int diff = whiteScore - blackScore;
        if(diff > 0)// so white score's is bigger
        {
            if(material > 0)
                result = whiteWin;
            else
                result = whiteAdvantage;
        }
        else if(diff < 0)
        {
            if(material < 0)
                result = blackWin;
            else
                result = whiteAdvantage;
        }
        else // equal - go by material
        {
            // just counter material
            if(material > 0)
                result = whiteWin;
            else if(material < 0)
                result = blackWin;
            else
                result = equal;
        }
    }
    return result;
}
public double evaluateMaterial()
{
    double boardValue = 0;
    if(ChessBoard.Wqueen != 0)
        boardValue += Wqueen;
    if(ChessBoard.Bqueen != 0)
        boardValue += Bqueen;
    for(int i = 63; i >= 0; i--)
    {
        if((ChessBoard.Bbishop & Move.SquareBits[i]) != 0)
            boardValue += Bbishop;
        else if((ChessBoard.Bknight & Move.SquareBits[i]) != 0)
            boardValue += Bknight;
        else if((ChessBoard.Brook & Move.SquareBits[i]) != 0)
            boardValue += Brook;
        else if((ChessBoard.Bpawn & Move.SquareBits[i]) != 0)
            boardValue += Bpawn;
        else if((ChessBoard.Wbishop & Move.SquareBits[i]) != 0)
            boardValue += Wbishop;
        else if((ChessBoard.Wknight & Move.SquareBits[i]) != 0)
            boardValue += Wknight;
        else if((ChessBoard.Wrook & Move.SquareBits[i]) != 0)
            boardValue += Wrook;
        else if((ChessBoard.Wpawn & Move.SquareBits[i]) != 0)
            boardValue += Wpawn;
    }
    return boardValue;
}
```java
int score = 0; //starts out "equal"
//check for doubled pawns
for(int loc = 0; loc < 8; loc++)
{
    if(MaxFileP[loc] > loc)
        score -= 8;
} //end doubled pawns for loop

//check for isolated pawns
if ( ( MaxFileP[ 0 ] > 0 ) && ( MaxFileP[ 1 ] == 0 ) )
    score -= 15;
if ( ( MaxFileP[ 7 ] > 0 ) && ( MaxFileP[ 6 ] == 0 ) )
    score -= 15;
for(int loc = 1; loc < 7; loc++)
{
    if ( (MaxFileP[ loc ] > 0 ) && ( MaxFileP[ loc - 1 ] == 0 )
        && ( MaxFileP[ loc + 1 ] == 0 ) )
        score -= 15;
}
if(TotalPawns == 8)
    score -= 10;
score -= 8 * PawnBacked;//because restrict movement

//passed pawn?
if ( MaxMostAdvanced[ 0 ] < Math.min( MinMostBackward[ 0 ], MinMostBackward
```
1  result is : 0  White win
2  result is : 0  White win
3  result is : 1  Black win
4  result is : 1  Black win
5  result is : 0  White win
6  result is : 1  Black win
7  result is : 0  White win
8  result is : 1  Black win
9  result is : 1  Black win
10 result is : 1  Black win
11 result is : 2  White adv/slightly better
12 result is : 1  Black win
13 result is : 3  Black adv/slightly better
14 result is : 6  Equal
15 result is : 6  Equal
16 result is : 2  White adv/slightly better