# General info

Possible moves should be stored in a SortedSet<Integer>:

SortedSet<Integer> possibleMoves = new TreeSet<>(Collections.reverseOrder());

The move integer (32 bit) representation should have on the most signifiant bits the most important aspects like isCheckMate, isCheck. This is because in sortedSet the moves will be added in descending order (because of Collections.reverseOrder() comparator provided to TreeSet constructor).

When iterating over possibleMoves set, i.e.

for (int y : possibleMoves) {

System.out.println(y);

}

This way, results will be retrieved in descending order, i. e. the best moves (checkMate or kingChess or capture) will be retrieved first.

# MoveService

There should be a MoveService class that has methods for getting values such as:

boolean isCheckMate(int move)

Position getInitialPosition(int move)

..

int createMove(Position initialPosition, Position finalPosition, boolean isCheckMate, .. etc)

# Move representation in 32 bit format

check-mate: 1 bit

capture: 4 bits

promotion: 4 bits

king in check: 1 bit

small-castling: 1 bit

big-castling: 1 bit

piece type: 4 bits

initialPosition x,y: 3+3=6 bits

finalPosition x,y: 3+3=6 bits

# Pieces

None: 0000

White king 0001

Black king 0010

White pawn 0011

Black pawn 0100

White knight 0101

Black knight 0110

White bishop 0111

Black Bishop 1000

White Rook 1001

Black Rook 1010

White Queen 1011

Black Queen 1100

# Exaplanation on using sortedSet

See the following example to understand why use SortedSet<Integer> possibleMoves = new TreeSet<>(Collections.reverseOrder()); :

public static void main(String[] args) {

//Collections.reverseOrder() is the comparator that will be used by the TreeSet in order to store moves in descending order (best moves first).

**SortedSet<Integer> possibleMoves = new TreeSet<>(Collections.reverseOrder());**

possibleMoves.add(new Integer(20));

possibleMoves.add(new Integer(4));

possibleMoves.add(new Integer(10));

possibleMoves.add(new Integer(3));

possibleMoves.add(new Integer(13));

possibleMoves.add(new Integer(14));

possibleMoves.add(new Integer(15));

possibleMoves.add(new Integer(1));

**for (int y : possibleMoves) {**

System.out.print(y);

} // prints 20 15 14 13 10 4 3 1

}