Tr	Try a few moves:												
8	bR		bB										
7	bP +	bP	bP	bP	bP +	bP	bP	bP	  -				
6	 +	 	 +		 +	 	 	 					
5	 +	 +	 +		 +	 	 	 					
4	 +	 +	 +		 +	 	 	 					
3	 +	 +	 +		 +	 	 	 +					
2	wP +	wP +	WP	wP	wP +	wP	WP	wP					
1	wR	wN	wB	wQ	wK +	wB	wN	wR					

abcde f g h

e2-e4

# 

+----+

a b c d e f g h

1 | wR | wN | wB | wQ | wK | wB | wN | wR |

b7	7-b6							
8	bR	bN	bB	bQ	bK	bB	bN	bR
7	bP		bP	bP	bP	bP	bP	bP
6		bP						
5	i			 				
4					wP			+ 
3								
2	wP	wP	wP	wP		wP	wP	wP
1	wR	wN	wB	wQ	wK	wB	wN	wR
	a	b	С	d	е	f	g 	h

wN	_	1-f3								
8		bR								

	+	+	+	·	+			+	+
7	bP	'	bP						  -
6	İ	bP		l					
5	!								
4		+ 		•	wP			+ 	+
3	İ		+ 	l		wN			† 
2	wP	wP		wP		wP	wP	wP	+
1	wR		+   wB						+
	+	h	+	d 	+	f		h	+

main-output.txt

bBc8-b7												
8		bN			bK		bN	bR				
7	bP			bP		bP	'	bP				
6		bP										
5												
4	!				wP							
3						wN						
2	wP	wP		wP		wP	wP	wP				
1	wR	wN			wK	wB		wR				
	a	b	С	d	е	f	g	h				

d2	-d3							
8	bR	bN		bQ	bK	bB	bN	bR
7	bP	bB	bP			bP	bP	bP
6		bP						
5								
4					wP			
3				wP		wN		
2	wP	wP	wP			wP	wP	wP
1	wR	wN	wB	wQ	wK	wB		wR
	a	b	С	d	е	f	g	h

bI	3b'	7xe4	1														
8	İ	bR	Ì	bN	İ		İ	bQ	İ	bK	İ	bB	İ	bN	İ	bR	
7	İ	bP	İ		İ	bP	İ	bP	İ	bP	İ	bP	i	bP	İ	bP	
	Τ.				- +		-Τ.				Ψ.		-Τ.		· +		_

d3xe4

+---+---+----8 | bR | bN | | bQ | bK | bB | bN | bR | 6 | | bP | | | | | | +---+ 5 | | | | | | | | +---+---+---+ 4 | | | | | | | | | | | | | | | +---+ 3 | | | | | | | | | | | | | | | | +---+ +---+ 1 | wR | wN | wB | wQ | wK | wB | | wR | +---+ a b c d e f g h

Reset board

-----

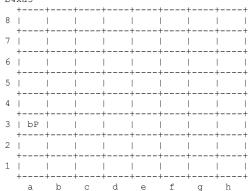
Test En Passant



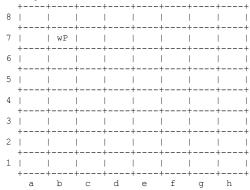
mair		

Montag, 13. November 2017 09:15

#### b4xa3



### Test promotion



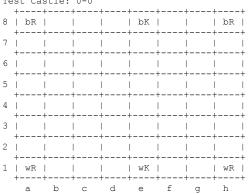
### b7-b8w0



1110	ıııı ou	quu.c							
4	-								1
_							+		
3				•		•	 +		
2									i i
							+		
1					I		I		1
	+		+	+	+	+	+	+	++
	ē	3	b	С	d	е	f	g	h

\_\_\_\_\_\_

Test Castle: 0-0



+---+ 8 | bR | | | | bK | | bR | +---+ 7 | | | | | | | | +---+ 6 | | | | | | | | +---+---+ 5 | | | | | | | | 4 | | | | | | | | 3 | | | | | | | | 2 | | | | | | | | | 1 | wR | | | | | wR | wK | | +---+ a b c d e f q h

Can white castle: false Can black castle: true

0 - 08 | bR | | | | bR | bK | | 7 | | | | | | | | | +---+ 6 | | | | | | | |

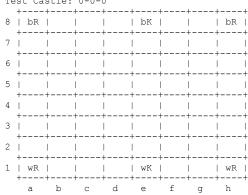
main-output.txt
-----------------

Montag, 13. November 2017 09:15

	+	+	+	+	+	+	+	+
5	1				1	l		
	+	+	+	+	+	+	+	+
4	1				1	l		
	+	+	+	+	+	+	+	+
3	1				1	l		
	+	+	+	+	+	+	+	+
2	1		[		1	l	ſ	ſ
	+	+	+	+	+	+	+	+
1	wR		[		1	wR	wK	ſ
	+	+	+	+	+	+	+	+
	a	b	С	d	е	f	g	h

Can white castle: false Can black castle: false

Test Castle: 0-0-0



0-0-0 +---+---+---+ 8 | bR | | | | bK | | bR | +---+ 7 | | | | | | | | 6 | | | | | | | | | 5 | | | | | | | | | 4 | | | | | | | | | 3 | | | | | | | | 2 | | | | | | | | | +---+ 1 | | wK | wR | | | wR | +---+---+---+ a b c d e f q h

Can white castle: false Can black castle: true

0 - 0 - 0

main-output.txt

8 | bR | | | | bK | | bR | 7 | | | | | | | | | | | 6 | | | | | | | | | | | 5 | | | | | | | | | | | 4 | | | | | | | | | | | Montag, 13. November 2017 09:15

a b c d e f q h

wRa1xa8

	+	+	+	+				
8	wR				bK			bR
7								
6			+ 					
5		+ 	+ 	++ 	+ 	++ 	++ 	
4	+	+ 	+ 	+ 	+ 	+	++ 	+ <del>+</del>
3	+	+ 	+ 	++ 	+ 	++ 	⊧+ 	+ <del>-</del>
2	+	+ I	+ I	+	+ I	+		+
1	+	+	+	+	 wK			wR
_	+	 +	 	 		 	 	
	а	b	C	d	e	f	ď	h

Can white castle: true Can black castle: true

bKe8-f8

8	wR					bK		bR
7		 		 	! !			
6					+ 			
5	+	+	+	+	+	+	++	++
4	+	+	+	+	+	+	++	++
3	+	+	+	+	+	+	+	++
2	+	+	+	+	+	+	++	++
1	+ 	+ 	+ 	+ 	+   wK	+ 	+	++   wR

+---+

8 | | | bK | bR | | | bR |

7 | | | | | | | |

6 | | | | | | | |

5 | | | | | | | |

4 | | | | | | | |

Can white castle: false Can black castle: false

Test move with king

				, ,	ı				
8	bR				bK			bR	
7									
6									
5									
4									
3									
2									
1	wR				wK			wR	
	a	b	C	d	e	f	d 	h	

wKe1-e2

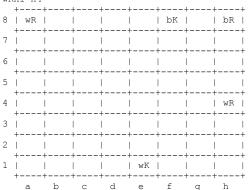
Can white castle: false

main-output.txt Montag, 13. November 2017 09:15

a b c d e f g h

Can white castle: true Can black castle: false

wRh1-h4



Can white castle: false Can black castle: false

-----

-9-

COMPUTER Exercise 1 Kraml - Reisinger, 1256528

Listing 1: source/Figure.java

```
1 package org.chpr.chess.objects;
  import org.chpr.chess.IBoard;
5 import java.util.ArrayList;
  import java.util.List;
  public class Figure {
    public static final int PAWN = 1;
    public static final int ROOK = 2;
     public static final int KNIGHT = 3;
     public static final int BISHOP = 4;
     public static final int QUEEN = 5;
     public static final int KING = 6;
     public static final String PAWN_STRING = "P";
     public static final String ROOK_STRING = "R";
     public static final String KNIGHT STRING = "N";
     public static final String BISHOP STRING = "B";
    public static final String QUEEN STRING = "Q";
     public static final String KING STRING = "K";
     public static final int WHITE OFFSET = 0;
     public static final int BLACK OFFSET = 10;
     public static final String WHITE STRING = "w";
     public static final String BLACK STRING = "b";
     public static final int WHITE = 0;
     public static final int BLACK = 1;
     public static final String [] ARR TYPE STRING = {PAWN STRING, ROOK STRING,
        KNIGHT STRING, BISHOP STRING, QUEEN STRING, KING STRING);
     public static int getType(int figureIndex) {
      return figureIndex % 10;
     public static int getColor(int figureIndex) {
      return figureIndex / 10;
    public static String toString(int figureIndex) {
      return toString(getColor(figureIndex), getType(figureIndex));
45
     public static String toString(int color, int type) {
      if (type == 0) {
        return "uu";
      String ret = "";
      if (color == WHITE)
        ret = ret.concat(WHITE STRING);
      else if (color == BLACK)
        ret = ret.concat(BLACK STRING);
       ret = ret.concat(ARR TYPE STRING[type - 1]);
      return ret;
     public static short fromString(String str) {
      short ret = 0;
      if (str.startsWith(WHITE STRING)) {
        ret += WHITE OFFSET;
```

10

```
str = str.substring(1);
65
        else if (str.startsWith(BLACK STRING)) {
          ret += BLACK OFFSET;
          str = str.substring(1);
        for (int i = 0; i < ARR TYPE STRING.length; i++)
70
          if (str.equals(ARR TYPE STRING[i]))
            return (short)(ret + i + 1);
        return -1;
75
     static public List < Move> getValidMoves (IBoard board, int col, int row) {
        //TODO: Implement in exercise 2
       return null;
      static private boolean is Valid Destination (IBoard board, int color, int col, int row)
        \mathbf{if} (col < 0 || col > 7)
         return false;
        if (row < 0 \mid | row > 7)
         return false;
85
        short[][] figures = board.getFigures();
        short fig = figures[col][row];
        if (color == WHITE && fig > 0 && fig < BLACK OFFSET)
          return false;
        if (color == BLACK && fig > BLACK OFFSET)
90
          return false;
        return true;
      static private boolean isFree(IBoard board, int col, int row) {
95
       if (\operatorname{col} < 0 \mid \mid \operatorname{col} > 7)
          return false
        if (row < 0 \mid | row > 7)
         return false;
        short[][] figures = board.getFigures();
100
        short fig = figures [col][row];
        if (fig == 0)
         return true;
        return false;
105 }
```

### Listing 2: source/Main.java

```
package org.chpr;
import org.chpr.chess.Board;
import org.chpr.chess.objects.Figure;
import org.chpr.chess.objects.Move;

public class Main {
    private static Board board;
    private static int currentColor;

    public static void main(String[] args) {
        currentColor = Figure.WHITE;
        board = new Board();

    testFewMoves();
    testEnPassant();
    testPromotion();
```

```
testCastle();
20
      private static void testFewMoves() {
          printBoard("Try_a_few_moves:");
          executeAndPrint("e2-e4");
25
          executeAndPrint("b7-b6");
          executeAndPrint("Ng1-f3");
          executeAndPrint("Bc8-b7");
          executeAndPrint("d2-d3");
          executeAndPrint("Bb7xe4");
30
          executeAndPrint("d3xe4");
          resetBoard(true);
          System.out.println("-----\n");
35
      private static void testEnPassant()
           short[][] figures = new short[8][8];
           figures [0][3] = Figure .PAWN + Figure .WHITE OFFSET;
           figures [1][3] = Figure .PAWN + Figure .BLACK OFFSET;
40
          board.setFigures(figures);
          currentColor = Figure.BLACK;
          printBoard("Test, En, Passant");
          executeAndPrint("b4xa3");
45
          resetBoard (false);
      private static void testPromotion()
          short[][] figures = new short[8][8];
50
          figures [1][6] = Figure .PAWN + Figure .WHITE_OFFSET;
          board.setFigures(figures);
          currentColor = Figure.WHITE;
          printBoard("Test_promotion");
55
           executeAndPrint("b7-b8Q");
          resetBoard (false);
          System.out.println("-----\n");
      private static void testCastle() {
          setUpCastle();
          printBoard("Test_Castle: 0-0");
          executeAndPrint("0-0");
          printCastleStatus();
          executeAndPrint("0-0");
          printCastleStatus();
          setUpCastle();
          printBoard("Test_Castle: 0-0-0");
70
          executeAndPrint("0-0-0");
          printCastleStatus();
          executeAndPrint("0-0-0");
           printCastleStatus();
          setUpCastle();
75
          printBoard("Test_move_with_king");
          executeAndPrint("Ke1-e2");
          printCastleStatus();
80
          setUpCastle();
          printBoard("Test_move_with_rooks");
          executeAndPrint("Ra1xa8");
```

```
printCastleStatus();
            executeAndPrint("Ke8-f8");
 85
            printCastleStatus();
            executeAndPrint("Rh1-h4");
            printCastleStatus();
            resetBoard (false);
            System.out.println("-----\n");
 90
        private static void setUpCastle() {
            resetBoard (false);
 95
            short [][] figures = new short [8][8];
            figures [0][0] = Figure .ROOK + Figure .WHITE OFFSET;
            figures [7][0] = Figure .ROOK + Figure .WHITE OFFSET;
            figures [4][0] = Figure.KING+ Figure.WHITE OFFSET;
100
            figures [0][7] = Figure .ROOK + Figure .BLACK OFFSET;
            figures [7] [7] = Figure .ROOK + Figure .BLACK OFFSET;
            figures [4][7] = Figure.KING + Figure.BLACK OFFSET;
            board.setFigures(figures);
105
            currentColor = Figure.WHITE;
        private static void executeAndPrint(String moveString) {
            Move m = Move.Import(moveString, board, currentColor);
110
            board.executeMove(m);
            System.out.println(m);
            System.out.println(board);
            System.out.println("\n");
115
            currentColor = (currentColor == Figure.WHITE ? Figure.BLACK : Figure.WHITE);
        private static void printBoard(String title) {
            System.out.println(title);
120
            printBoard();
        private static void printBoard() {
            System.out.println(board);
125
            System.out.println("\n");
        private static void printCastleStatus() {
            System.out.println("Can_white_castle: " + board.canWhiteCastle());
130
            System.out.println("Can, black, castle: " + board.canBlackCastle());
            System.out.println("\n");
        private static void resetBoard(boolean print) {
135
            board.reset();
            currentColor = Figure.WHITE;
            if (print) {
                System.out.println("Reset_board");
                System.out.println(board);
140
                System.out.println("\n");
```

Listing 3: source/Move.java

```
1 | // e2-e4 pawn moves from e2 to e4
   // e2xf3 pawn moves from e2 to f3 and hits
     Ra1-a8 rook moves from a1 to a8
   // a7xb8N pawn hits b8 and gets promoted to knight
5 // a7-a8Q pawn moves to a8 and gets promoted to queen
   // 0-0 king side castle
   // 0-0-0 queen side castle
10 package org.chpr.chess.objects;
  import org.chpr.chess.IBoard;
  import java.util.List:
  public class Move {
     private int color;
     private final int type;
    private final int sourceCol;
     private final int sourceRow;
     private final int destCol;
     private final int destRow;
     private int fig;
    private IBoard board;
     private boolean hit;
     private boolean prom;
     public Move(IBoard board, int color, int type, int sourceCol, int sourceRow, int
        destCol, int destRow, boolean newType) {
       this.color = color;
       this.type = type;
       this.sourceCol = sourceCol;
       this.sourceRow = sourceRow;
       this.destCol = destCol:
       this.destRow = destRow;
       this.board = board;
      this.fig = type + (color == Figure.WHITE ? Figure.WHITE OFFSET :
          Figure .BLACK OFFSET);
       this.hit = false; // will be handled by setHit()
      this.prom = newType;
40
     public IBoard getBoard() {
      return board;
     public int getFigureIndex() {
      return fig;
    public int getColor() {
      return color;
     public int getType() {
      return type;
     public int getSourceCol() {
      return sourceCol;
60
     public int getSourceRow() {
```

```
return sourceRow;
65
      public int getDestCol() {
       return destCol;
     public int getDestRow() {
       return destRow;
      public void setColor(int color) {
75
       if (this.color != color) {
          if (color == Figure.WHITE)
            fig = 10;
          else if (color == Figure.BLACK)
            fig += 10;
80
        this.color = color;
      public boolean isHit() {
85
       return hit;
      public void setHit() {
        hit = true;
90
      public boolean isProm() {
       return prom;
95
      @Override
      public String toString() {
        String ret = "";
        if (Figure.getType(fig) == Figure.KING) {
100
         if (sourceCol = 4 && destCol = 6)
           return "0-0";
          if (sourceCol = 4 && destCol = 2)
           return "0-0-0";
        if (Figure.getType(fig) != Figure.PAWN && !prom) {
105
         ret = ret.concat(Figure.toString(fig));
        char[] col = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'};
        ret = ret.concat(Character.toString(col[sourceCol]));
110
        ret = ret.concat(Integer.toString(sourceRow + 1).concat(hit ? "x" : "-"));
        ret = ret.concat(Character.toString(col[destCol]));
        ret = ret.concat(Integer.toString(destRow + 1));
        if (prom)
          ret = ret.concat(Figure.toString(fig));
115
         short[][] b = board.getFigures();
         for (int column = 0; column < b.length; column++) {
           for (int row = 0; row < b[0]. length; row++) {
              if (b[column][row] == Figure.KING + (color == Figure.WHITE)
       Figure .BLACK OFFSET : Figure .WHITE OFFSET) ) {
120
                // TODO check for check, need valid moves for this
        return ret;
125
```

```
@Override
      public boolean equals(Object obj) {
       if (obj.getClass() != Move.class)
         return false;
        Move m = (Move)obj;
        if (color != m.getColor())
         return false;
        if (!board.equals(m.getBoard()))
135
         return false;
        if (sourceCol != m.getSourceCol())
         return false;
        if (sourceRow != m.getSourceRow())
         return false:
        if (destCol != m.getDestCol())
         return false;
        if (destRow != m.getDestRow())
         return false;
        if (fig != m.getFigureIndex())
145
         return false;
        // no need to check for hit, because when all is equal, then hit is also equal
            (when setHit() is used correctly)
        // no need to check for color, its in figureIndex
        // no need to check for type, its in figureIndex (fig)
        // no need to check for promotion, because when all is equal, then promotion is
            also equal
150
       return true;
      public static Move Import(String str, IBoard board, int color) {
       if (str.equals("0-0")) {
155
         int row = (color == Figure.WHITE ? 0 : 7);
         return new Move(board, color, Figure.KING, 4, row, 6, row, false);
        if (str.equals("0-0-0")) {
         int row = (color == Figure.WHITE ? 0 : 7);
160
          return new Move(board, color, Figure.KING, 4, row, 2, row, false);
        char[] col = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'};
        int figureIndex = 0;
        if (color == Figure.BLACK)
         figureIndex += Figure.BLACK OFFSET;
165
        // if str starts with a-h, then its a pawn
        for (char c : col) {
         if (str.startsWith(Character.toString(c))) {
            figureIndex += Figure.PAWN:
170
            // concat for equal position of cols, rows, ...
            str = "P".concat(str);
            break;
175
        if (str.startsWith(Figure.ROOK STRING))
          figureIndex += Figure.ROOK;
        else if (str.startsWith(Figure.KNIGHT STRING))
          figureIndex += Figure.KNIGHT:
        else if (str.startsWith(Figure.BISHOP STRING))
          figureIndex += Figure.BISHOP;
        else if (str.startsWith(Figure.QUEEN STRING))
         figureIndex += Figure.QUEEN;
        else if (str.startsWith(Figure.KING STRING))
         figureIndex += Figure.KING;
185
        int sourceCol = str.charAt(1) - 'a';
        int sourceRow = Integer.parseInt(Character.toString(str.charAt(2))) - 1;
        boolean hit = Character.toString(str.charAt(3)).equals("x") ? true : false;
```

Kraml - Reisinger, 1256528

```
int destCol = str.charAt(4) - 'a';
        int destRow = Integer.parseInt(Character.toString(str.charAt(5))) - 1;
190
        // if str.length() is 7, then it must be a promotion with the additional
           newTypeStr at the end
        if (str.length() == 7) {
          String newTypeStr = Character.toString(str.charAt(6));
          int newType = Figure.fromString(newTypeStr);
         Move m = new Move(board, color, newType, sourceCol, sourceRow, destCol, destRow.
             true);
195
          if (hit)
           m. setHit();
         return m;
        Move m = new Move(board, color, Figure.getType(figureIndex), sourceCol, sourceRow,
           destCol, destRow, false);
200
        if (hit)
         m.setHit();
        return m;
205
      public static boolean MovesListIncludesMove(List<Move> moves, Move move) {
       for (Move m : moves)
          if (move.equals(m))
           return true;
        return false;
210
```

## Listing 4: source/IBoard.java

```
1 package org.chpr.chess;
   import org.chpr.chess.objects.Move;
5 import java.util.List;
    * Interface for public methods of our chess board
10 public interface IBoard {
      * Return all figures that are on the board
      * @return array of figures
15
     short[][] getFigures();
      * Set given figure at given position
20
      * Oparam row the row where the figure should be positioned
      * @param column the column where the figure should be positioned
     * @param figure the figure that should be positioned
25
     void setFigure(int row, int column, short figure);
     * Reset the board, i.e. set board to start position
30
     void reset();
      * Copy board except with empty history
35
     * @return incomplete copy of the board
```

```
IBoard cloneIncompletely();
     * Return list of all valid moves on the current board
     * @return list of all valid moves
     List < Move> get Valid Moves();
45
     * Return list of all valid moves for a given color on the current board
     * @param color color for which valid moves should be returned
     * @return list of all valid moves for given color
     List < Move> getValidMoves(int color);
55
     * Return List of previously executed moves
     * @return list of previously executed moves
     List < Move> get History();
60
     * Execute given move on current position
     * @param move move to execute
65
     void executeMove(Move move);
     * Check if white can still castle
70
     * @return true, if white can still castle
     boolean canWhiteCastle();
75
     * Check if black can still castle
     * @return true, if black can still castle
     boolean canBlackCastle();
     * Check if color is in mat
     * @param color color which should be checked for
     * @return true, if color is in mat
     boolean isMat(int color);
```

# Listing 5: source/Board.java

```
package org.chpr.chess;
import org.chpr.chess.objects.Figure;
import org.chpr.chess.objects.Move;

import java.util.LinkedList;
import java.util.List;
```

COMPUTER Exercise 1

```
| public class Board implements IBoard {
     static private int COLS = 8;
     static private int ROWS = 8;
     private short[][] figures;
     private boolean canWhiteCastleKingside = true;
15
     private boolean canWhiteCastleQueenside = true;
     private boolean canBlackCastleKingside = true;
     private boolean canBlackCastleQueenside = true;
     private List < Move> history;
20
     public Board() {
       this.reset();
     @Override
     public short[][] getFigures() {
       return figures;
     @Override
     public void setFigure(int row, int column, short figure) {
       figures [column] [row] = figure;
35
      * Set figures of board for better testing
      * @param figures new position of board
     public void setFigures(short[][] figures) {
       this.figures = figures;
     @Override
     public void reset() {
       figures = new short [COLS] [ROWS];
        // set white pieces
       figures [0] [0] = Figure .ROOK + Figure .WHITE OFFSET;
       figures [1][0] = Figure.KNIGHT + Figure.WHITE OFFSET;
50
       figures [2][0] = Figure.BISHOP + Figure.WHITE OFFSET;
       figures [3][0] = Figure .QUEEN + Figure .WHITE OFFSET;
       figures [4][0] = Figure.KING+ Figure.WHITE OFFSET;
       figures [5][0] = Figure .BISHOP + Figure .WHITE OFFSET;
       figures [6] [0] = Figure .KNIGHT+ Figure .WHITE OFFSET;
55
       figures [7] [0] = Figure .ROOK + Figure .WHITE OFFSET;
       for (int col = 0; col < COLS; col++) {
         figures [col][1] = Figure .PAWN + Figure .WHITE OFFSET;
       // set black pieces
60
       int lastRow = ROWS - 1;
       figures [0] [lastRow] = Figure.ROOK + Figure.BLACK OFFSET;
       figures [1] [lastRow] = Figure.KNIGHT + Figure.BLACK OFFSET;
       figures [2] [lastRow] = Figure.BISHOP + Figure.BLACK_OFFSET;
       figures [3] [lastRow] = Figure.QUEEN + Figure.BLACK OFFSET;
       figures [4] [lastRow] = Figure.KING+ Figure.BLACK OFFSET;
65
       figures [5] [lastRow] = Figure.BISHOP + Figure.BLACK OFFSET;
       figures [6] [lastRow] = Figure.KNIGHT+ Figure.BLACK OFFSET;
       figures [7] [lastRow] = Figure .ROOK + Figure .BLACK OFFSET;
       for (int col = 0; col < COLS; col++) {
70
         figures [col] [lastRow - 1] = Figure .PAWN + Figure .BLACK OFFSET;
       // reset other properties
```

```
canWhiteCastleKingside = true;
        canWhiteCastleQueenside = true:
 75
        canBlackCastleKingside = true;
        canBlackCastleQueenside = true;
        history = new LinkedList <>();
      @Override
      public IBoard cloneIncompletely() {
       Board clonedBoard = new Board();
        clonedBoard.figures = figures;
        clonedBoard.canWhiteCastleKingside = canWhiteCastleKingside;
        clonedBoard.canWhiteCastleQueenside = canWhiteCastleQueenside;
        clonedBoard.canBlackCastleKingside = canBlackCastleKingside;
        clonedBoard.canBlackCastleQueenside = canBlackCastleQueenside;
        return clonedBoard;
 90
      @Override
      public List < Move> getValidMoves() {
        //TODO: Implement in exercise 2
        return null;
      @Override
      public List < Move> getValidMoves(int color) {
        //TODO: Implement in exercise 2
100
        return null;
      @Override
      public List<Move> getHistory() {
105
       return history;
      @Override
      public void executeMove (Move move)
       int srcCol = move.getSourceCol();
        int srcRow = move.getSourceRow();
        int destCol = move.getDestCol();
        int destRow = move.getDestRow();
        short srcFigure = figures[srcCol][srcRow];
115
        short destFigure = figures[destCol][destRow];
        boolean whiteMove = move.getColor() == Figure.WHITE;
        figures [destCol][destRow] = srcFigure;
        figures[srcCol][srcRow] = 0;
        if (move.isProm()) {
           // promotion
          short newFigure = (short) (move.getType() + (whiteMove ? Figure.WHITE OFFSET :
              Figure .BLACK OFFSET));
          figures [destCol] [destRow] = newFigure;
125
        if (move.getType() == Figure.PAWN && move.isHit() && destFigure == 0) {
          figures [destCol][destRow + (whiteMove ? -1 : 1)] = 0;
130
        if (move.getType() == Figure.KING ) {
          if (Math.abs(destCol - srcCol) == 2) {
            // castle
            if (whiteMove) {
              if (destCol == 2) {
135
                figures [0][0] = 0;
```

```
figures [3][0] = Figure.ROOK + Figure.WHITE OFFSET;
             } else {
                figures [7][0] = 0;
                figures[5][0] = Figure.ROOK + Figure.WHITE OFFSET;
140
           } else {
              if (destCol == 2) {
                figures [0][7] = 0;
                figures [3][7] = Figure.ROOK + Figure.BLACK_OFFSET;
145
              } else {
                figures [7][7] = 0;
                figures [5][7] = Figure.ROOK + Figure.BLACK_OFFSET;
150
          if (whiteMove) {
           canWhiteCastleQueenside = canWhiteCastleKingside = false;
            canBlackCastleQueenside = canBlackCastleKingside = false;
155
        if (move.getType() == Figure.ROOK) {
          // if rook was moved set can castle options
160
          if (whiteMove) {
           if (srcCol == 0 && srcRow == 0) canWhiteCastleQueenside = false;
           if (srcCol = 7 && srcRow = 0) canWhiteCastleKingside = false;
           if (srcCol == 0 && srcRow == 7) canBlackCastleQueenside = false;
165
           if (srcCol = 7 && srcRow = 7) canBlackCastleKingside = false;
        history.add(move);
170
      @Override
      public boolean canWhiteCastle()
       return canWhiteCastleKingside || canWhiteCastleQueenside;
175
      @Override
      public boolean canBlackCastle()
       return canBlackCastleKingside || canBlackCastleQueenside;
180
      @Override
      public boolean isMat(int color) {
        // return true if no king of given color is on the board
        for (int row = 0; row < ROWS; row++) {
185
         for (int col = 0; col < COLS; col++) {
           short figure = figures[col][row];
           if (Figure.getColor(figure) = color && Figure.getType(figure) = Figure.KING)
             return false;
190
        return true;
195
      @Override
      public String toString() {
        String frame = "___+---+\n";
        StringBuilder sb = new StringBuilder();
```

```
200
       sb.append(frame);
       sb.append(row + 1); sb.append("_{\sqcup}");
       for (int col = 0; col < COLS; col++) {
        sb.append("|u" + Figure.toString(figures[col][row]) + "u");
       sb.append("|\n");
205
     sb.append(frame);
     return sb.toString();
210
```

COMPUTER Exercise 1