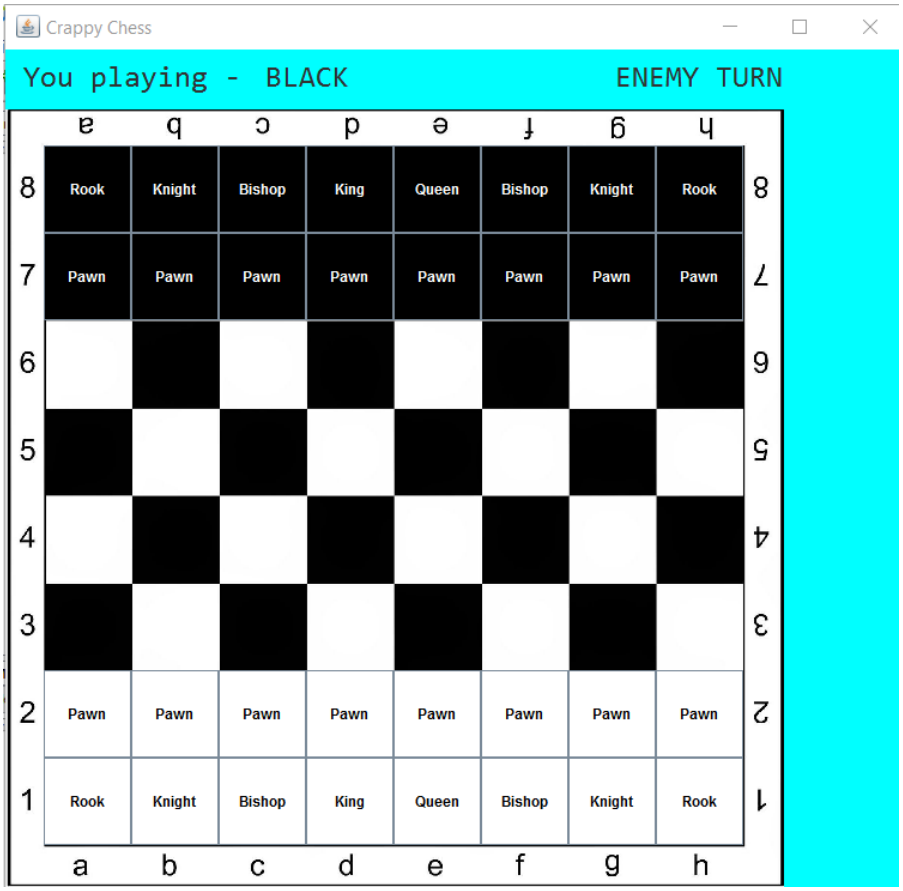
Project report

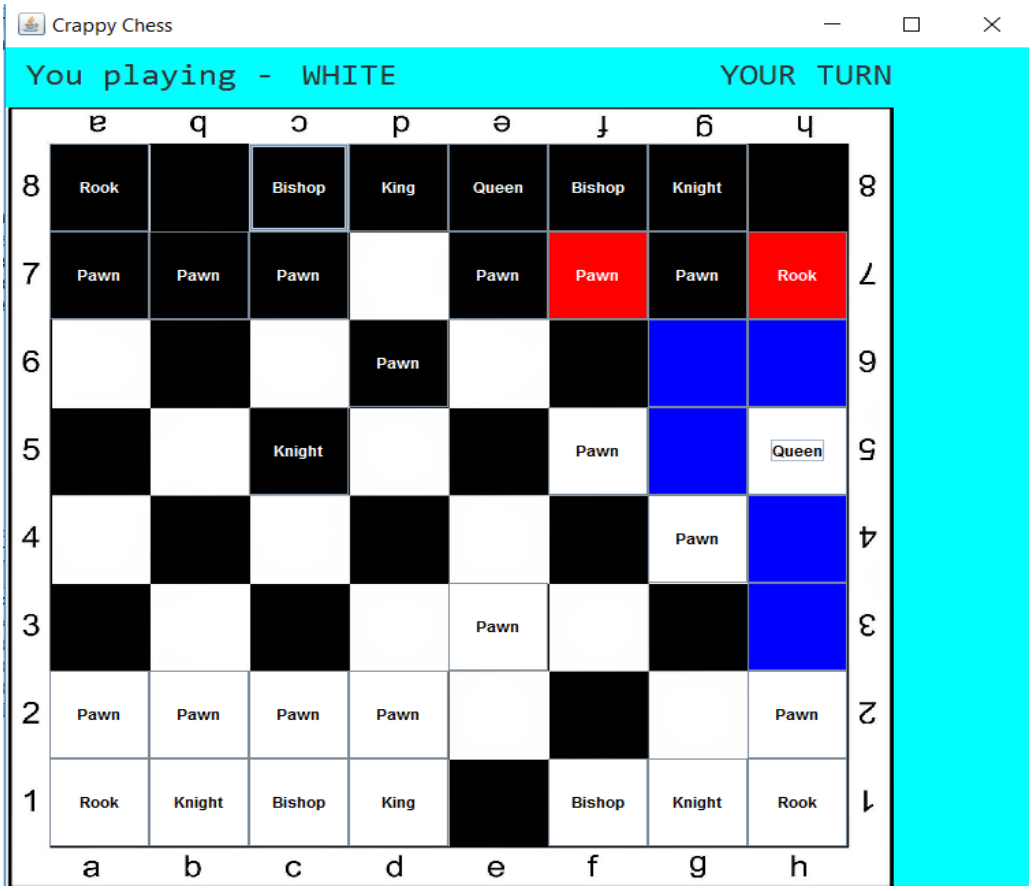
Java Chess game



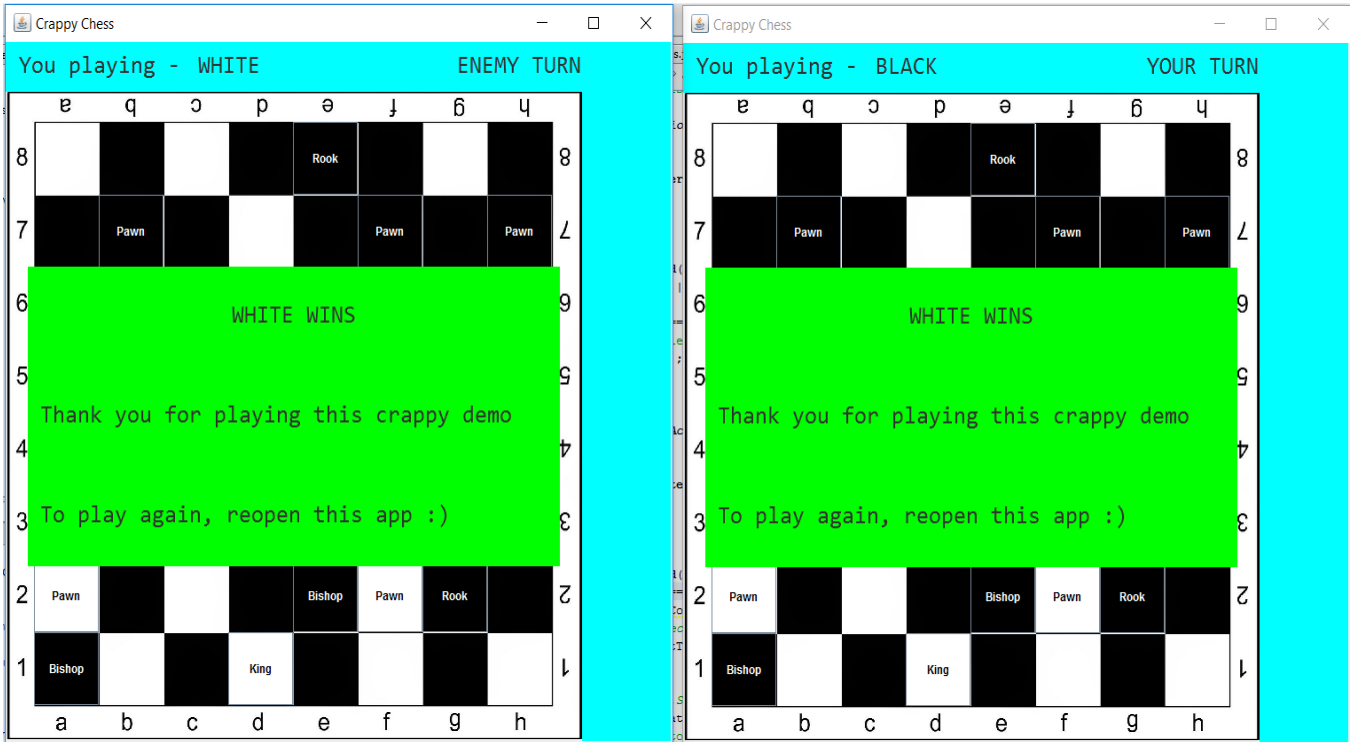
Picture 1

By Kondratenko Danylo

On Picture 1 you can see initial state of desk when two players are connected and ready to play. In-game information can be found in upper panel. All pieces are showing their possible moves after clicking on it (If its players turn to move) as its on Picture 2. Game ends when one of the kings are killed.



Picture 2



End game window

Important functions.

After game is launched we are initiating game field as a layered panel. And setting up all content in Main.InitForm(). After game field is created, we are creating thread for ServerAPI.

```
public void InitForm(){
    setTitle("Crappy Chess");
    setPreferredSize(new Dimension(770, 750));
    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setLayout(new BorderLayout());
    add(lpane, BorderLayout.CENTER);
    lpane.setBounds(0, 0, 850, 700);
    InfoPanel.setBounds(0,0,650,50);
    InfoPanel.setBackground(java.awt.Color.CYAN);
    NavMenu.setBounds(650,0,100,700);
    NavMenu.setBackground(java.awt.Color.CYAN);
    Board.setBounds(1, 49, 650, 650);
    Board.setOpaque(true);
    Field.setBounds(32, 80, 585, 585);
    Field.setOpaque(false);
    Field.setLayout(new GridLayout(8,8));
    InfoPanel.setLayout(new GridLayout(1,3));
    PlayerColorInfo.setFont(new Font("Consolas",Font.PLAIN, 25));
    Label.setFont(new Font("Consolas",Font.PLAIN, 25));
    PlayerTurnInfo.setFont(new Font("Consolas",Font.PLAIN, 25));
    PlayerTurnInfo.setHorizontalAlignment(SwingConstants.RIGHT);
    Label.setVisible(false);
    InfoPanel.add(Label);
    InfoPanel.add(PlayerColorInfo);
    InfoPanel.add(PlayerTurnInfo);
    EndGame.setBackground(Color.green);
    EndGame.setBounds(25, 225, 600, 300);
    EndGame.setLayout(new GridLayout(3,1));
    Winner.setFont(new Font("Consolas",Font.PLAIN, 25));
    Winner.setHorizontalAlignment(SwingConstants.CENTER);
    EndGame.add(Winner);
    Gratz.setFont(new Font("Consolas",Font.PLAIN, 25));;
    EndGame.add(Gratz);
    ReGame.setFont(new Font("Consolas",Font.PLAIN, 25));
    EndGame.add(ReGame);
    EndGame.setVisible(false);
    lpane.add(EndGame, new Integer(2));
    lpane.add(Board, new Integer(0));
    lpane.add(InfoPanel, new Integer(1));
    lpane.add(NavMenu, new Integer(1));
    lpane.add(Field, new Integer(1));
    pack();
    setVisible(true);
}
```

Init function to initialize all panels and game window

ServerAPI – is responsible for connection between players and receiving and sending moves. On initialization it creates a socket on port 9991, if it fails to create socket, it tries to connect to localhost on that port assuming that port is busy, and server is up. After a successful connection it thread is starting to listen commands from users.

```
public static void ReceiveMove(){
    try{
        Vector2 From = new Vector2(in.readInt(),in.readInt());
        Vector2 To = new Vector2(in.readInt(),in.readInt());
        if(Main.PiecesOnBoard[To.x][To.y].rank.getCode() == "King"){
            if(Main.PiecesOnBoard[From.x][From.y].color == Main.PlayerColor.Black.getValue())
                Main.Winner.setText("BLACK WINS");
                Main.EndGame();
        }
        Main.PiecesOnBoard[To.x][To.y].Update(Main.PiecesOnBoard[From.x][From.y]);
        Main.PiecesOnBoard[From.x][From.y].Update(new Piece(Piece.Rank.Empty, 0, 0));
        Main.PiecesOnBoard[From.x][From.y].button.setVisible(false);
        Main.PiecesOnBoard[To.x][To.y].button.setVisible(true);
        Main.IsPlayerTurn = true;
        Main.PlayerTurnInfo.setText("YOUR TURN");
    }catch (IOException ex) {
        Logger.getLogger(ServerAPI.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Function that receives piece moves from players

Every piece is separated container with a button assigned to it. Every piece has DefaultActionListener of EmptyActionListener. Player own pieces has Default listener, and rest has Empty listener. Pieces with Default listeners are can be moved, whenever pieces with Empty listeners are responding only whenever they could be possible position for piece.

```
public static class DefaultActionListener implements ActionListener {
    Piece piece = null;

    public DefaultActionListener(Piece target){
        piece = target;
    };

    @Override
    public void actionPerformed(ActionEvent e) {
        if(!Main.IsPlayerTurn || Main.PlayerColor_.getValue() != this.piece.color) return;
        Moves.Deselect();
        if(Main.SelectedPiece == piece) {Main.SelectedPiece = null; return;}
        Main.SelectedPiece = piece;
        Moves.CheckMoves(piece);
    }
}

public static class EmptyPieceActionListener implements ActionListener {
    public Piece piece = null;

    public EmptyPieceActionListener(Piece target){
        piece = target;
    };

    @Override
    public void actionPerformed(ActionEvent e) {
        if(Main.SelectedPiece == null || !piece.IsActivated) return;
        if(this.piece.rank.getCode() == "King"){
            if(Main.SelectedPiece.color == Main.PlayerColor.Black.getValue())
                Main.Winner.setText("BLACK WINS");
            Main.EndGame();
        }
        this.piece.Update(Main.SelectedPiece);
        Main.SelectedPiece.Update(new Piece(Piece.Rank.Empty, 0, 0));
        Main.SelectedPiece.button.setVisible(false);
        Main.server.SendMove(Main.FindPiece(Main.SelectedPiece),Main.FindPiece(this.piece));
        Main.SelectedPiece = null;
        Main.IsPlayerTurn = false;
        Main.PlayerTurnInfo.setText("ENEMY TURN");
        Moves.Deselect();
    }
}
```

Two types of listeners for pieces

Moves.CheckMoves(piece) – This function calls for the Moves script which is also the container of all possible moves for every figure. Also Moves script highlights the possible moves for piece.

Piece.Update(Piece target) – This function allows to move piece to a different position by switching values of itself to values of the piece which should move to its position.

```
public void Update(Piece target){
    this.button.setBackground(target.button.getBackground());
    this.button.removeActionListener(actionListener);
    this.button.setForeground(target.button.getForeground());
    this.button.setText(target.button.getText());
    this.color = target.color;
    this.rank = target.rank;
    this.IsEnemy = target.IsEnemy;
    if(target.IsEnemy)
        actionListener = new EmptyPieceActionListener(this);
    else
        actionListener = new DefaultActionListener(this);
    this.button.addActionListener(actionListener);
}
```

Function that update piece by switching values in containers