# “SWING” JAVA GUIDE FOR PD GAME

NOTE:THIS IS A PARTIAL GUIDE

public class PDGameGUI extends JFrame implements ActionListener, ListSelectionListener {

//instance variables for PDGameGUI, are up here so all methods below can access them

private final DefaultListModel<String> listModel = new DefaultListModel<String>();

private final JList<String> finishedGamesList; //jlist will be put in the listmodel

private final JTextField computerStrategyTF = new JTextField(10);

private final JLabel computerStrategyL = new JLabel("Computer Strategy-Combo Box");

private final JButton startB = new JButton("Start New Game");

private final JComboBox<Object> computerStrategyCB;

private int computerStrategy = 1;

private final JLabel decisionL = new JLabel("Your decision this round?");

private final JTextArea gameResultsTA = new JTextArea(15, 30);

private PDGame currentPDGame = null;

private String gameStartTime = null;

private final HashMap<String, GameStatsptr> stats = new HashMap<>();

public static **void main(String[] args**) {

createAndShowGUI();

}

public static void **createAndShowGUI()** {

// Create and set up the window.

PDGameGUI pdg1 = new PDGameGUI(); //call constructor below to set the window to user

pdg1.addListeners(); //method will add listeners to buttons

pdg1.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Display the window and pack together all the panels, etc

pdg1.pack();

pdg1.setVisible(true);

}

**//CONSTRUCTOR WHERE YOU BUILD THE SWING INTERFACE (partial code follows)**

public PDGameG2() {

super("JIM Prisoner's Dilemma");

currentPDGame = new PDGame();

//setSize(800, 800);

//setPreferredSize(600,800);

setLayout(new BorderLayout());

// Set up left yellow panel

JPanel panel1 = new JPanel(new BorderLayout());

"JList of Games holding a List Model"));

add(panel1, BorderLayout.WEST); //add to frame

// Set up top left "list"

finishedGamesList = new JList<String>(listModel);

finishedGamesList.setFont(new Font("SansSerif", Font.BOLD, 24));

finishedGamesList.setVisibleRowCount(10);

finishedGamesList.setFixedCellWidth(550);

finishedGamesList.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

panel1.add(new JScrollPane(finishedGamesList), BorderLayout.NORTH);

//fill in other panels, grids etc

//COMBO BOX ON RIGHT

Object[] strategyArray = currentPDGame.getStrategies().toArray();//convert AL to array

computerStrategyCB = new JComboBox<Object>(strategyArray);

computerStrategyCB.setEditable(false);

computerStrategyCB.setSelectedIndex(0);

panel5.add(computerStrategyCB);

} //end constructor

//hook up listeners to buttons

public **void addListeners() {**

startB.addActionListener(this); // do this for all buttons ETC….

}

@Override **//HANDLES WHT BUTTON WAS CLICKED and calls appropriate method**

public void actionPerformed(ActionEvent e) {

if (e.getSource() == startB) {

startGame();

} else if (e.getSource() == silentB) {

cooperate();

} else if (e.getSource() == betrayB) {

betray();

} else if (e.getSource() == computerStrategyCB) {

computerStrategy = computerStrategyCB.getSelectedIndex() + 1;

}

}

public void startGame() {

currentPDGame = new PDGame();

currentPDGame.setStrategy(computerStrategy);

gameStartTime = (new Date()).toString();

stats.put(gameStartTime, currentPDGame.getStats());

promptPlayer(); etc

}

@Override

**//user has clicked on a finished game in upper left list box,**

**// show results from game**

**public void valueChanged(ListSelectionEvent e**) {

if (!finishedGamesList.isSelectionEmpty()) {

searchKey = (String) finishedGamesList.getSelectedValue(); //get out time of game and look up in hash map

roundsTF.setText(new Integer(gameStatsInfo.getRounds()).toString());

roundsTF.setFont( new Font("SansSerif", Font.BOLD, 24));

playerSentenceTF.setFont( new Font("SansSerif", Font.BOLD, 24));

playerSentenceTF.setText(String.format("%d %s", playerSentenceYrs,

((playerSentenceYrs > 1) ? " years" : " year")));

}

}

**Creating custom colors**

Color c1 = new Color(211, 211, 55); //higher numbers means lighter colors

myPanel.setBackground(c1) //sets the color