JAVA GUI SWING

**JFrame Functions:**

JFrame frame = new JFrame(); //creates a frame

Frame.setTitle(“Hello Swing”); // sets title of frame

Frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); //exit out of application

Frame.setResizable(false); //prevents frame from being resized

Frame.setSize(450,450); // sets size of frame;

Frame.setVisible(true); // makes frame visible

**To change the icon of the frame:**

ImageIcon image = new ImageIcon("C:\\Users\\hp\\Desktop\\hci logo.png"); //pass the image path in the constructor

frame.setIconImage(image.getImage()); //sets the frame icon

**To change the bg color of frame:**

frame.getContentPane().setBackground(Color.red); //changes to color to red

frame.getContentPane().setBackground(new Color(0xB095FB)); //using hexadecimal code

frame.pack(); //sets the size of frame according to the components

**JLabel Functions:**

Labels are used to display text or images

JLabel label = new JLabel(); // creates a new label

label.setText("Bro, do you even code?"); //adds text to label

JLabel label = new JLabel(“Bro, do you even code?”); //can be done this this also

# Then add it to frame

Frame.add(label);

**To add image:**

#### Hello

#### Hello

**hello**

ImageIcon image = new ImageIcon("C:\\Users\\hp\\Desktop\\hci logo.png");

label.setIcon(image);

label.setHorizontalTextPosition(JLabel.CENTER); //sets the text on the left, center or right of the image

label.setVerticalTextPosition(JLabel.TOP); //sets the text on the top, center or bottom of the image

label.setForeground(Color.BLUE); //changes color of text

label.setFont(new Font("Cambria", Font.PLAIN, 20)); //changes font style – paramters are name of font, style and size

label.setIconTextGap(50); //sets a gap between image and the text

label.setBackground(Color.red); //sets the bg color of the labels

label.setOpaque(true); //makes the bg color of the label visible

**to add border:**

Border border = BorderFactory.createLineBorder(Color.yellow,3); //creates a border – parameters are color, width

label.setBorder(border); //sets the border for label

label.setVerticalAlignment(JLabel.TOP); //vertically aligns the label

label.setHorizontalAlignment(JLabel.CENTER); //horizontally aligns the label

**to set label in a limited space:**

frame.setLayout(null); //set the frame layout to null

label.setBounds(10, 10, 250, 250); // set the size and position of the label – x-axis, y-axis, width, height

## JPanel Functions

It is a component that functions as a container to hold other components

JPanel redPanel = new JPanel(); // creates a Panel

redPanel.setBackground(Color.red); // sets panel bg red

redPanel.setBounds(20, 20, 400, 400); // sets the position and size of panel (x,y,width,height)

**adding components to panel:**

JLabel label= new JLabel("Labels"); // creating a label

label.setFont(new Font("Cambria",Font.BOLD,20)); //setting its font

redPanel.add(label); //adding it to the panel

**to set alignment of components:**

redPanel.setLayout(new BorderLayout()); //set the layout to BorderLayout

label.setVerticalAlignment(JLabel.CENTER);

label.setHorizontalAlignment(JLabel.CENTER); // then change alignment of component

*if not using borderLayout, set the layout of panel to null and setBounds for the label*

redPanel.setLayout(null);

label.setBounds(10, 10, 70, 100);

**JButton Functions:**

It is a button that performs an action when clicked

JButton button = new JButton(); // creates a button

button.setText("I am a button"); // sets the text of button

button.setFocusable(false); //removes the border of the text from the button

ImageIcon image = new ImageIcon("C:\\Users\\hp\\Desktop\\images.jpeg"); //creates image icon

button.setIcon(image); //sets the image on the button

button.setFont(new Font("Comic Sans", Font.BOLD, 14)); //set the font of button text

*\*all functions same as JLabel*

JButton button = new JButton();

button.setBounds(50, 50, 250, 150);

button.setText("I am a button");

button.setFocusable(false);

button.setIcon(image);

button.setFont(new Font("Comic Sans", Font.BOLD, 14));

button.setHorizontalTextPosition(JButton.CENTER);

button.setVerticalTextPosition(JButton.TOP);

button.setForeground(Color.red);

button.setBackground(Color.BLUE);

button.setBorder(BorderFactory.createEtchedBorder());

### BorderLayout

Places components in 5 areas: north, south, east, west and center

All extra space is placed in the center

By Default, frames use a borderLayout

JFrame frame = new JFrame();

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

frame.setSize(450,450);

frame.setVisible(true);

frame.setLayout(new BorderLayout(10,10)); //set layout and parameters are the margin btw   
components

JPanel panel1 = new JPanel();

JPanel panel2 = new JPanel();

JPanel panel3 = new JPanel();

JPanel panel4 = new JPanel();

JPanel panel5 = new JPanel();

panel1.setBackground(Color.red);

panel2.setBackground(Color.BLUE);

panel3.setBackground(Color.GREEN);

panel4.setBackground(Color.YELLOW);

panel5.setBackground(Color.BLACK);

panel1.setPreferredSize(new Dimension(100,100)); //set size of panel

panel2.setPreferredSize(new Dimension(100,100));

panel3.setPreferredSize(new Dimension(100,100));

panel4.setPreferredSize(new Dimension(100,100));

panel5.setPreferredSize(new Dimension(100,100));

//----------------sub panels ------------------ //a panel can have a borderLayout as well

JPanel panel6 = new JPanel();

JPanel panel7 = new JPanel();

JPanel panel8 = new JPanel();

JPanel panel9 = new JPanel();

JPanel panel10 = new JPanel();

panel6.setBackground(Color.CYAN);

panel7.setBackground(Color.GRAY);

panel8.setBackground(Color.PINK);

panel9.setBackground(Color.WHITE);

panel10.setBackground(Color.MAGENTA);

panel5.setLayout(new BorderLayout(10,10));

panel6.setPreferredSize(new Dimension(50,50));

panel7.setPreferredSize(new Dimension(50,50));

panel8.setPreferredSize(new Dimension(50,50));

panel9.setPreferredSize(new Dimension(50,50));

panel10.setPreferredSize(new Dimension(50,50));

panel5.add(panel6, BorderLayout.NORTH);

panel5.add(panel7, BorderLayout.SOUTH);

panel5.add(panel8, BorderLayout.EAST);

panel5.add(panel9, BorderLayout.WEST);

panel5.add(panel10, BorderLayout.CENTER);

//--------------sub panels--------------

frame.add(panel1, BorderLayout.NORTH); //when adding, set the position

frame.add(panel2, BorderLayout.SOUTH);

frame.add(panel3, BorderLayout.EAST);

frame.add(panel4, BorderLayout.WEST);

frame.add(panel5, BorderLayout.CENTER);

## FlowLayout

Places the components in a row until no place is left and then moves to the next row.

JFrame frame = new JFrame();

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

frame.setSize(450,450);

frame.setLayout(new FlowLayout(FlowLayout.LEADING, 10,10));

// first parameter is the alignment of the components, and the second and third is horizontal and vertical spacing between the components

frame.setVisible(true);

frame.add(new JButton("1"));

frame.add(new JButton("2"));

frame.add(new JButton("3"));

frame.add(new JButton("4"));

frame.add(new JButton("5"));

frame.add(new JButton("6"));

frame.add(new JButton("7"));

frame.add(new JButton("8"));

## GridLayout

Places components in a grid (i.e rows and cols). Each component takes all the space in a cell. All the cells are of same size

JFrame frame = new JFrame();

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

frame.setSize(450,450);

frame.setLayout(new GridLayout(3,3,10,10)); //args(rows, cols, horizontal\_margin, vertical\_margin)

frame.setVisible(true);

frame.add(new JButton("1"));

frame.add(new JButton("2"));

frame.add(new JButton("3"));

frame.add(new JButton("4"));

frame.add(new JButton("5"));

frame.add(new JButton("6"));

frame.add(new JButton("7"));

frame.add(new JButton("8"));

## JLayeredPane

This is a swing container that provides a third dimension for positioning components i.e the depth or z-index

It arranges the components in layers or on top of each other.

There are 5 layers in JLayeredPane (button to top):

1. Default
2. Palette
3. Modal
4. PopUp
5. Drag

JLabel label1 = new JLabel();

label1.setOpaque(true);

label1.setBackground(Color.red);

label1.setBounds(50, 50, 200, 200);

JLabel label2 = new JLabel();

label2.setOpaque(true);

label2.setBackground(Color.GREEN);

label2.setBounds(100, 100, 200, 200);

JLabel label3 = new JLabel();

label3.setOpaque(true);

label3.setBackground(Color.blue);

label3.setBounds(150, 150, 200, 200);

JLayeredPane pane = new JLayeredPane(); // creates a layeredPane

pane.setBounds(0, 0, 500, 500); //set size

pane.add(label1, JLayeredPane.DEFAULT\_LAYER); //add the labels by mentioning on which layer it   
should be

pane.add(label2, JLayeredPane.DEFAULT\_LAYER);

pane.add(label3, JLayeredPane.DRAG\_LAYER);

*can also set the layer like this (higher number on top):*

pane.add(label1, Integer.valueOf(0));

pane.add(label2, Integer.valueOf(3));

pane.add(label3, Integer.valueOf(2));

JFrame frame = new JFrame();

frame.add(pane);

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

frame.setSize(450,450);

frame.setLayout(null);

frame.setVisible(true);

**Click button to open new window:**

We need to create different classes for different windows. We use the ActionListener class to perform the action and add the addActionListener function to the button.

We can call this class in the main class.

public class PopUp implements ActionListener{

JFrame frame = new JFrame();

JButton button = new JButton();

PopUp(){

button.setText("Click me");

button.setFocusable(false);

button.setBounds(100, 100, 100, 50);

button.addActionListener(this); //creates the action

frame.add(button);

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

frame.setLayout(null);

frame.setSize(450,450);

frame.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

if(e.getSource()==button)

{

frame.dispose(); //closes the first window on clicking the button

MyFrame jj = new MyFrame(); //generates the second window

}

}

}

## JOptionPane

This pops up a dialog box. Different types of dialog boxes can be generated.

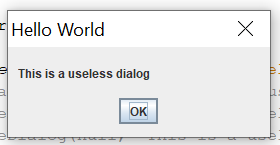
1. Standard box that informs user of something

JOptionPane.showMessageDialog(null, "This is a useless dialog", "Hello World", JOptionPane.PLAIN\_MESSAGE);

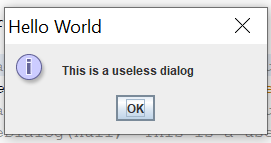
* This create a dialog box. The parameters are – parent component, message, title and dialog type
* Plain message is the type in which the message has no formatting.
* The dialog box’s look and feel depends on the type we choose.

The types are:

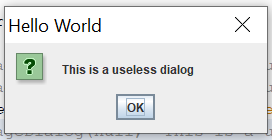
JOptionPane.showMessageDialog(null, "This is a useless dialog", "Hello World", JOptionPane.PLAIN\_MESSAGE);



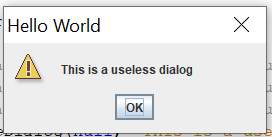
JOptionPane.showMessageDialog(null, "This is a useless dialog", "Hello World", JOptionPane.INFORMATION\_MESSAGE);



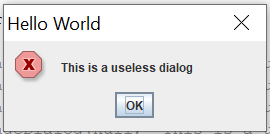
JOptionPane.showMessageDialog(null, "This is a useless dialog", "Hello World", JOptionPane.QUESTION\_MESSAGE);



JOptionPane.showMessageDialog(null, "This is a useless dialog", "Hello World", JOptionPane.WARNING\_MESSAGE);



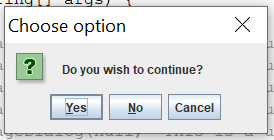
JOptionPane.showMessageDialog(null, "This is a useless dialog", "Hello World", JOptionPane.ERROR\_MESSAGE);



1. A dialog box that user that choose yes, no or cancel from

* Same parameters as previous one

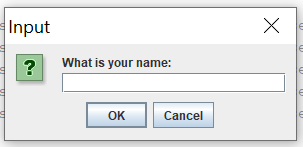
JOptionPane.showConfirmDialog(null, "Do you wish to continue?", "Choose option", JOptionPane.YES\_NO\_CANCEL\_OPTION);



This returns 3 values, 0 for yes, 1 for no, 2 for cancel and -1 for cross.

1. Dialog box in which user can enter something

JOptionPane.showInputDialog("What is your name: ");

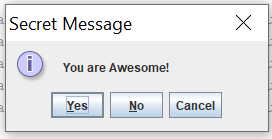


1. This combines all the three into one

JOptionPane.showOptionDialog(parentComponent, message, title, OptionType, MessageType, icon, options, initialValue);

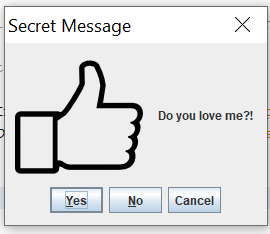
OptionType – the buttons that we need the dialog to have

JOptionPane.showOptionDialog(null, "You are Awesome!", "Secret Message", JOptionPane.YES\_NO\_CANCEL\_OPTION, JOptionPane.INFORMATION\_MESSAGE, null, null, 0);



ImageIcon image = new ImageIcon("C:\\\\Users\\\\hp\\\\Desktop\\\\20664.png");

JOptionPane.showOptionDialog(null, "Do you love me?!", "Secret Message", JOptionPane.YES\_NO\_CANCEL\_OPTION, JOptionPane.INFORMATION\_MESSAGE, image, null, 0);

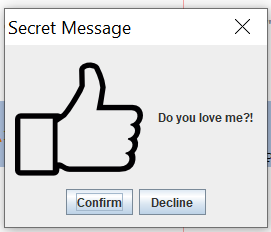


*To add our own options, we create a string array of the options and pass it in the args.*

String option[]= {"Confirm", "Decline"};

ImageIcon image = new ImageIcon("C:\\\\Users\\\\hp\\\\Desktop\\\\20664.png");

JOptionPane.showOptionDialog(null, "Do you love me?!", "Secret Message", JOptionPane.YES\_NO\_CANCEL\_OPTION, JOptionPane.INFORMATION\_MESSAGE, image, option, 0);



## JTextField

This is a text box in which user can enter data

public class Textbox extends JFrame implements ActionListener{

JFrame frame = new JFrame();

JButton submit = new JButton();

JTextField username = new JTextField();

Textbox()

{

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

this.setLayout(new FlowLayout());

username.setPreferredSize(new Dimension(250,40)); //set dimension or size of textfield

username.setFont(new Font("Cambria", Font.BOLD, 20));

username.setForeground(Color.white);

username.setBackground(Color.black);

username.setCaretColor(Color.blue); //sets the color of the line in the textbox

username.setText("Username"); //to set default text in the textbox

submit.setText("Submit");

submit.addActionListener(this); //does action when button is clicked

this.add(submit);

this.add(username);

this.pack();

this.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

if(e.getSource()==submit)

{

System.out.println("Welcome "+username.getText()); //extracts the text that was entered in   
the text field

username.setEditable(false); //disables the text field from being edited again

submit.setEnabled(false); //disables the button from being clicked again

}

}

}

## JCheckBox

public class Checkbox extends JFrame implements ActionListener{

JButton button = new JButton("Submit");

JCheckBox checkbox = new JCheckBox(); //creates a checkbox

Checkbox()

{

checkbox.setText("I am not a robot"); //adds text to checkbox

checkbox.setFocusable(false); //removes the outline from the checkbox

checkbox.setFont(new Font("Cambria", Font.BOLD, 20)); //sets font font and size of checkbox

checkbox.setIcon(defaultIcon); // sets the icon for the checkbox instead of the box

checkbox.setSelectedIcon(selectedIcon); //sets the icon for when the checkbox is selected

button.addActionListener(this);

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

this.setLayout(new FlowLayout());

this.add(button);

this.add(checkbox);

this.pack();

this.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

if(e.getSource()==button)

{

System.out.println(checkbox.isSelected());

}

}

}

## JRadioButtons

One or more buttons in a group in which only one will be selected at a time

We need to place the radio buttons in a ButtonGroup so only one can be selected at a time.

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

frame.setLayout(new FlowLayout());

JRadioButton pizza;

JRadioButton burger;

JRadioButton pasta;

pizza = new JRadioButton("Pizza");

burger = new JRadioButton("Burger");

pasta = new JRadioButton("Pasta");

ButtonGroup group = new ButtonGroup();

group.add(pizza);

group.add(burger);

group.add(pasta);

frame.add(pizza);

frame.add(burger);

frame.add(pasta);

frame.pack();

frame.setVisible(true);

## Combo Boxes

A component that combines a button or text in a dropdown list

JFrame frame = new JFrame();

String animals[]= {"Dog", "Cat","Parrot"};

JComboBox list = new JComboBox(animals); //pass a string array of the list

list.addItem("Horse"); //can also add item using function

list.insertItemAt("Swam", 1); //to insert at a certain index

frame.add(list);

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

frame.setLayout(new FlowLayout());

frame.pack();

frame.setVisible(true);

## Sliders

User can enter a value by adjusting the slider.

public class Slider implements ChangeListener{

JFrame frame;

JPanel panel;

JLabel label;

JSlider slider;

Slider()

{

frame = new JFrame("Slider Demo");

panel= new JPanel();

label = new JLabel();

slider = new JSlider(0,100,50); //initial value,end value, starting value

slider.setPreferredSize(new Dimension(400,200));

slider.setPaintTicks(true); //sets the small lines on the slider

slider.setMinorTickSpacing(10); //adds the small line after every 10 units

slider.setPaintTrack(true); //sets the large lines on the slider

slider.setMajorTickSpacing(25); //adds the large line after every 25 units

slider.setPaintLabels(true); //adds numbers to the lines

slider.setFont(new Font("Cambria", Font.PLAIN,14));

slider.setOrientation(SwingConstants.VERTICAL); //changes the slider from horizontal to vertical

label.setText("Degree Celceuis = "+ slider.getValue());

label.setFont(new Font("Cambria", Font.PLAIN,20));

panel.add(slider);

panel.add(label);

frame.add(panel);

frame.setSize(420,420);

frame.setVisible(true);

}

## ProgressBar

Let’s the user know about the progress of the operation.

## JFileChooser

Lets user choose a file from the computer. Helpful for opening and saving files.

JFileChooser fc = new JFileChooser(); //launches an instance of the file chooser

Fc.showOpenDialog(null); //selects file to open, parament is some parent, this returns an integer value as a response. When you choose and open a file it returns 0, when you cancel the dialog it returns 1, and when you click cross it returns 1 as well.

To check if a file is choosen, you can either check if response is 0 or you can use the following:

If(response == JFileChooser.APPROVE\_OPTION)

{

File file = new File(fc.getSelectedFile().getAbsolutePath());

// if a file is choosen it creates a new file and sets the path of the file as the file choosen. The file variable now has the path of the file that was opened.

}

To open save dialog:

Fc.showSaveDialog(null); //this will select file to save

To select the default location in which the file chooser opens:

Fc.setCurrentDirectory(new File(“.”)); //parameters is the path to the directory. “.” Selects the path for the current project.