**LAB : 2**

**OBJECTIVE :**

Working with J2ME Features

1. Develop an application that uses GUI components, Font and Colours
2. Develop Calculator Application.

**Requrements :**

(a) Windows PC (Windows 7/8/10) / Mac

(b) JDK 1.5

(c) Java Wireless Toolkit 2.5.2

**Implementation :**

1. Develop an application that uses GUI components, Font and Colours

**GuiFont.java :**

import javax.microedition.lcdui.\*;

import javax.microedition.midlet.\*;

public class GuiFont extends MIDlet

implements CommandListener {

// display manager

Display display;

// main menu : a menu with items

List menu;

// list of choices

List choose;

// textbox

TextBox input;

// ticker

Ticker ticker = new Ticker("3.1\_GUI\_Components");

// alerts

final Alert soundAlert = new Alert("sound Alert");

// date

DateField date = new DateField("Today's date: ", DateField.DATE);

// form

Form form = new Form("Form for Stuff");

// today's form

Form today = new Form("Today's date");

// gauge

Gauge gauge = new Gauge("Progress Bar", false, 20, 9);

// text field

TextField textfield = new TextField(

"TextField Label", "Dummy Text", 50, 0);

//canvas

private DecodeCanvas decodeCanvas = null;

private boolean painting = false;

public static final boolean COLOR = false;

public static final boolean DEBUG = false;

//colors

public static final int WHITE = 0xFFFFFF;

public static final int BLACK = 0x000000;

public static final int LIGHT\_GRAY = 0xAAAAAA;

public static final int DARK\_GRAY = 0x555555;

public static final int RED = 0xFF0000;

public static final int GREEN = 0x00FF00;

public static final int BLUE = 0x0000FF;

// command

static final Command backCommand = new Command("Back", Command.BACK, 0);

static final Command mainMenuCommand = new Command("Main", Command.SCREEN, 1);

static final Command exitCommand = new Command("Exit", Command.STOP, 2);

String currentMenu;

// constructor.

public GuiFont() {

}

/\*\*

\* Start the MIDlet by creating a list of

\* items and associating the

\* exit command with it.

\*/

public void startApp() throws

MIDletStateChangeException {

display = Display.getDisplay(this);

menu = new List("Test Components", Choice.IMPLICIT);

menu.append("Test TextBox", null);

menu.append("Test List", null);

menu.append("Test Alert", null);

menu.append("Test Date", null);

menu.append("Test Form", null);

menu.append("Font and color", null);

menu.addCommand(exitCommand);

menu.setCommandListener(this);

menu.setTicker(ticker);

mainMenu();

// form

form.append(gauge);

form.append(textfield);

// today

today.append(date);

}

public void pauseApp() {

display = null;

choose = null;

menu = null;

ticker = null;

form = null;

today = null;

input = null;

gauge = null;

textfield = null;

}

public void destroyApp(boolean unconditional) {

notifyDestroyed();

}

// main menu

void mainMenu() {

display.setCurrent(menu);

currentMenu = "Main";

}

// \* Test the TextBox component.

public void testTextBox() {

input = new TextBox

("Enter Some Text:", "", 10, TextField.ANY);

input.setTicker(new Ticker(

"Testing TextBox"));

input.addCommand(backCommand);

input.setCommandListener(this);

input.setString("");

display.setCurrent(input);

currentMenu = "input";

}

/\*\*

\* Test the List component.

\*/

public void testList() {

choose = new List("Choose Items",

Choice.MULTIPLE);

choose.setTicker(new Ticker(

"Testing List"));

choose.addCommand(backCommand);

choose.setCommandListener(this);

choose.append("Item 1", null);

choose.append("Item 2", null);

choose.append("Item 3", null);

display.setCurrent(choose);

currentMenu = "list";

}

/\*\*

\* Test the Alert component.

\*/

public void testAlert() {

soundAlert.setType(AlertType.ERROR);

//soundAlert.setTimeout(20);

soundAlert.setString("\*\* Alert Demo \*\* ");

display.setCurrent(soundAlert);

}

/\*\*

\* Test the DateField component.

\*/

public void testDate() {

java.util.Date now = new java.util.Date();

date.setDate(now);

today.addCommand(backCommand);

today.setCommandListener(this);

display.setCurrent(today);

currentMenu = "date";

}

/\*\*

\* Test the Form component.

\*/

public void testForm() {

form.addCommand(backCommand);

form.setCommandListener(this);

display.setCurrent(form);

currentMenu = "form";

}

/\*\*

\* Font

\*/

public void gui() {

//display = Display.getDisplay(this);

decodeCanvas = new DecodeCanvas(this);

decodeCanvas.addCommand(backCommand);

decodeCanvas.setCommandListener(this);

Display.getDisplay(this).setCurrent(decodeCanvas);

decodeCanvas.repaint();

currentMenu = "gui";

}

/\*\*

\* Handle events.

\*/

public void commandAction(Command c,

Displayable d) {

String label = c.getLabel();

if (label.equals("Exit")) {

destroyApp(true);

} else if (label.equals("Back")) {

if (currentMenu.equals("list")

|| currentMenu.equals("input")

|| currentMenu.equals("date")

|| currentMenu.equals("form")

|| currentMenu.equals("gui")) {

// go back to menu

mainMenu();

}

} else {

List down = (List)display.getCurrent();

switch (down.getSelectedIndex()) {

case 0: testTextBox(); break;

case 1: testList(); break;

case 2: testAlert(); break;

case 3: testDate(); break;

case 4: testForm(); break;

case 5: gui(); break;

}

}

}

class DecodeCanvas extends Canvas {

private GuiFont parent = null;

private int width = getWidth();

private int height = getHeight();

public DecodeCanvas(GuiFont parent) {

this.parent = parent;

}

public void paint(Graphics g) {

g.setColor(WHITE);

g.fillRect(0, 0, width, height);

Font f1 = Font.getFont(Font.FACE\_SYSTEM, Font.STYLE\_PLAIN, Font.SIZE\_LARGE);

Font f2 = Font.getFont(Font.FACE\_SYSTEM, Font.STYLE\_BOLD, Font.SIZE\_MEDIUM);

Font f3 = Font.getFont(Font.FACE\_SYSTEM, Font.STYLE\_ITALIC, Font.SIZE\_SMALL);

int yPos = 0;

if (COLOR)

g.setColor(BLUE);

else

g.setColor(LIGHT\_GRAY);

g.fillRect(0, yPos, width, f1.getHeight());

if (COLOR)

g.setColor(WHITE);

else

g.setColor(BLACK);

g.setFont(f1);

g.setColor(RED);

g.drawString("BIG FONT : Plain", 0, yPos, Graphics.LEFT | Graphics.TOP);

yPos = yPos + f1.getHeight() + 10;

g.setFont(f2);

// g.drawLine(0, f1.getHeight() + yPos - 1, width, f1.getHeight() + yPos - 1);

g.setColor(GREEN);

g.drawString("MEDIUM FONT : Bold", 0, yPos, Graphics.LEFT | Graphics.TOP);

//g.drawLine(0, f2.getHeight() + yPos - 1, width, f2.getHeight() + yPos - 1);

yPos = yPos + f1.getHeight() + 10;

g.setFont(f3);

g.setColor(BLUE);

g.drawString("SMALL FONT : Italic", 0, yPos, Graphics.LEFT | Graphics.TOP);

yPos = yPos + f1.getHeight() + 10;

g.drawLine(0, f3.getHeight() + yPos - 1, width, f3.getHeight() + yPos - 1);

painting = false;

}

}

}

**Output :**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

1. Develop an application that uses GUI components, Font and Colours

**Calculator.java :**

import javax.microedition.lcdui.\*;

import javax.microedition.midlet.MIDlet;

public final class Calculator extends MIDlet implements CommandListener, ItemCommandListener {

private static final int NUM\_SIZE = 20;

private final Command exitCmd = new Command("Exit", Command.EXIT, 2);

private final Command add = new Command("Add", Command.ITEM, 1);

private final Command sub = new Command("Sub", Command.ITEM, 1);

private final Command mul = new Command("Mul", Command.ITEM, 1);

private final Command div = new Command("Div", Command.ITEM, 1);

private final TextField textFiled\_1 = new TextField("Num 1 ", "", NUM\_SIZE, TextField.DECIMAL);

private final TextField textField\_2 = new TextField("Num 2 ", "", NUM\_SIZE, TextField.DECIMAL);

private final TextField tr = new TextField("Result", "", NUM\_SIZE, TextField.UNEDITABLE);

private final Alert alert = new Alert("Error", "", null, AlertType.ERROR);

private boolean isInitialized = false;

protected void startApp() {

if (isInitialized) {

return;

}

Form form = new Form("Calculator");

form.append(textFiled\_1);

form.append(textField\_2);

StringItem itema = new StringItem(" Add ", "", Item.BUTTON);

itema.setDefaultCommand(add);

itema.setItemCommandListener(this);

form.append(itema);

StringItem items = new StringItem("Subtract", "", Item.BUTTON);

items.setDefaultCommand(sub);

items.setItemCommandListener(this);

form.append(items);

StringItem itemm = new StringItem("Multiply", "", Item.BUTTON);

itemm.setDefaultCommand(mul);

itemm.setItemCommandListener(this);

form.append(itemm);

StringItem itemd = new StringItem(" Divide ", "", Item.BUTTON);

itemd.setDefaultCommand(div);

itemd.setItemCommandListener(this);

form.append(itemd);

form.append(tr);

form.addCommand(exitCmd);

form.setCommandListener(this);

Display.getDisplay(this).setCurrent(form);

isInitialized = true;

}

protected void destroyApp(boolean unconditional) {}

protected void pauseApp() {}

public void commandAction(Command c, Item item) {

double res = 0.0;

double n1 = getNumber(textFiled\_1, "First");

double n2 = getNumber(textField\_2, "Second");

try {

if (c == add)

res = n1 + n2;

if (c == sub)

res = n1 - n2;

if (c == mul)

res = n1 \* n2;

if (c == div)

res = n1 / n2;

} catch (NumberFormatException e) {

return;

} catch (ArithmeticException e) {

alert.setString("Divide by zero.");

Display.getDisplay(this).setCurrent(alert);

return;

}

String res\_str = Double.toString(res);

if (res\_str.length() > tr.getMaxSize()) {

tr.setMaxSize(res\_str.length());

}

tr.setString(res\_str);

}

public void commandAction(Command c, Displayable d) {

if (c == exitCmd) {

destroyApp(false);

notifyDestroyed();

return;

}

}

private double getNumber(TextField t, String type) throws NumberFormatException {

String s = t.getString();

if (s.length() == 0) {

alert.setString("No " + type + " Argument");

Display.getDisplay(this).setCurrent(alert);

throw new NumberFormatException();

}

double n;

try {

n = Double.parseDouble(s);

} catch (NumberFormatException e) {

alert.setString(type + " argument is out of range.");

Display.getDisplay(this).setCurrent(alert);

throw e;

}

return n;

}

}

**Output :**

|  |  |
| --- | --- |
|  |  |
|  |  |