Lecture 05
Two Dimensional Graphics
Part 2

Working with Text APIs

The java.awt.Font Class

Reference: Java-Tutorials/tutorial-2015/2d/text/index.html

Contents

- Font Concepts
- Physical and Logical Fonts
- The java.awt.Font class
- Measuring Text
- Demonstrations

1- Font Concepts

- **Font face** is a set of fonts available on the system.
- A ***Font object*** represents an instance of a font face.
- Examples of common font faces include Helvetica Bold and Courier Bold Italic.
- There are three different names that you can get from a Font object: its logical name, family name and font face name:
  - *Logical name* is a name mapped onto a physical font, which is one of the specific fonts available on the system
  - *Family name* is the name of the font family that determines the typographic design across several faces, such as Helvetica.
  - *Font face name* refers to an actual font installed on a system. This is the name you should use when specifying a font. It's often referred to as just the *font name*

2- Physical and Logical Fonts

- **Physical fonts** are the actual font libraries consisting of, for example, TrueType or PostScript Type 1 fonts. The physical fonts may be Time, Helvetica, Courier, or any number of other fonts, including international fonts.
- **Logical fonts** are the following five font families: Serif, SansSerif, Monospaced, Dialog, and DialogInput. These logical fonts are not actual font libraries. Instead, the logical font names are mapped to physical fonts by the Java runtime environment.
- ***Lucida*** font is a physical font. Oracle's JREs contain this family of physical fonts, which is also licensed for use in other implementations of the Java platform. These fonts are physical fonts, but do not depend on the host operating system.

Physical and Logical Fonts…

- **Bundling Physical Fonts, style PLAIN, in a file with your Application**

```
try { //Returned font is of pt size 1
  Font font = Font.createFont(Font.TRUETYPE_FONT, new File("A.ttf"));
 //Derive and return a 12 pt version: Need to use float otherwise  it would be interpreted as style
    return font.deriveFont(12f);
}
catch (IOException|FontFormatException e) {
// Handle exception
}
```

3- The java.awt.Font Class

- public class **Font** extends Object implements Serializable
- Fields:
  - Some constants for font styles, basic fonts
  - Font name, style, size
- Common constructor:

  Font(String name, int style, int size)
- Getters, setter

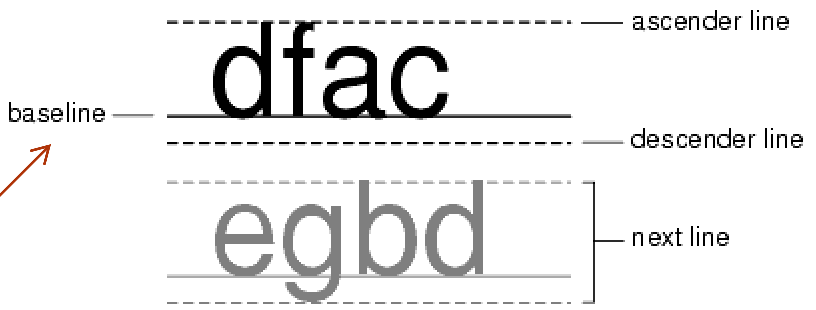
4- Measuring Text

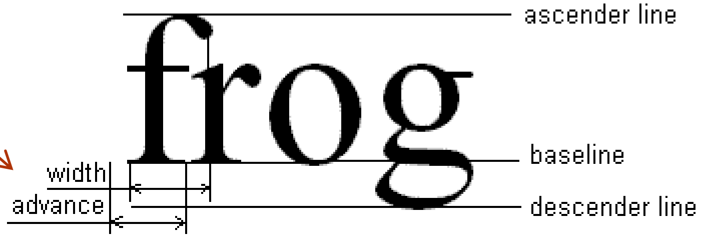
To properly measure text, you need to learn a few methods and some mistakes to avoid. Font metrics are measurements of text rendered by a Font object such as the height of a line of text in the font. The most common way to measure text is to use a FontMetrics instance which encapsulates this metrics information. For example:

```
// get metrics from the graphics
FontMetrics metrics = graphics.getFontMetrics(font);
// get the height of a line of text in this
// font and render context
int hgt = metrics.getHeight();
// get the advance of my text in this font
// and render context
int adv = metrics.stringWidth(text);
// calculate the size of a box to hold the
// text with some padding.
Dimension size = new Dimension(adv+2, hgt+2);
```

4- Measuring Text

- A LineMetrics object encapsulates the measurement information associated with a Font, such as its ascent, descent, and leading.
- The most common way to measure text is to use a FontMetrics instance which encapsulates this metrics information





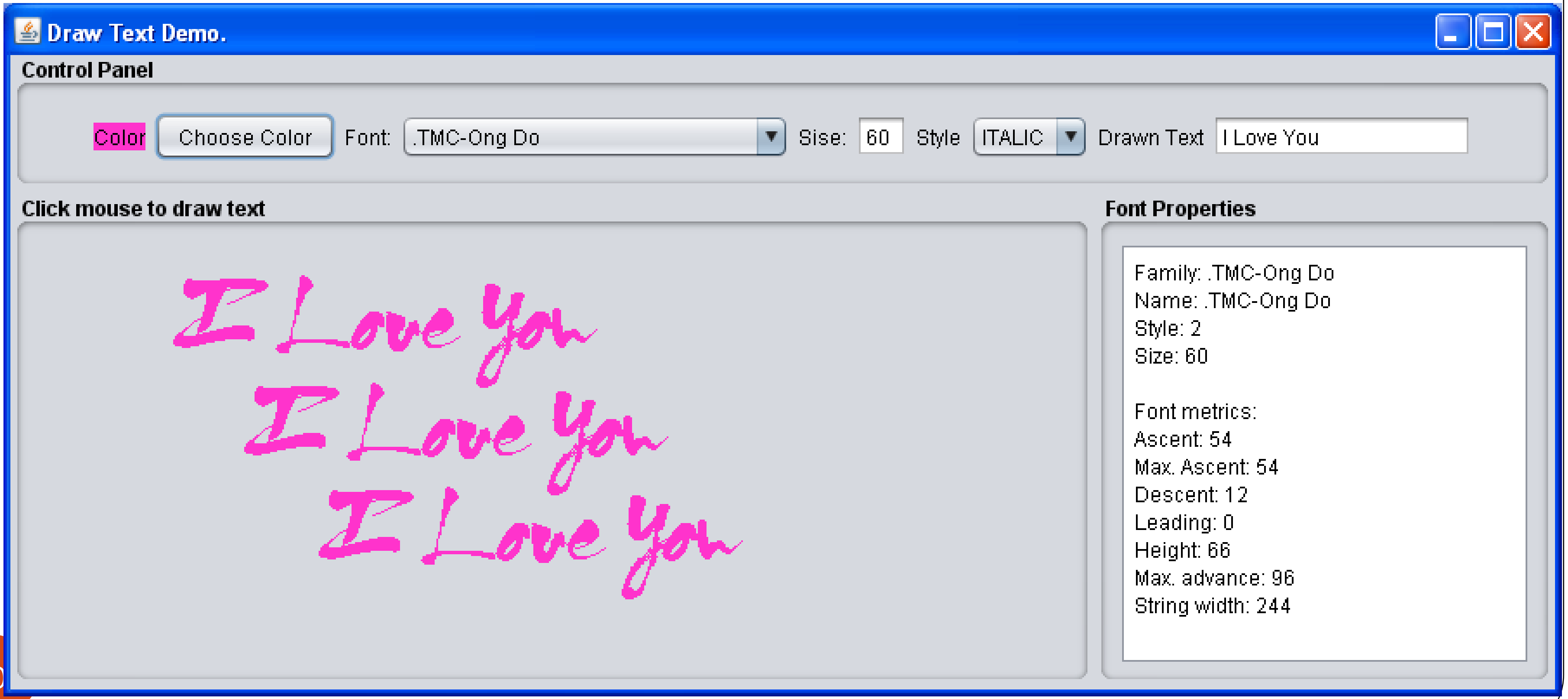
Demo 1- Draw Text

- This demonstration depict how to:
  - Get system fonts
  - Create a font object
  - Assign a font object to graphics object
  - Draw text on a component
  - Get font metrics

Demo 1- Draw Text

- This demonstration depicts how to:

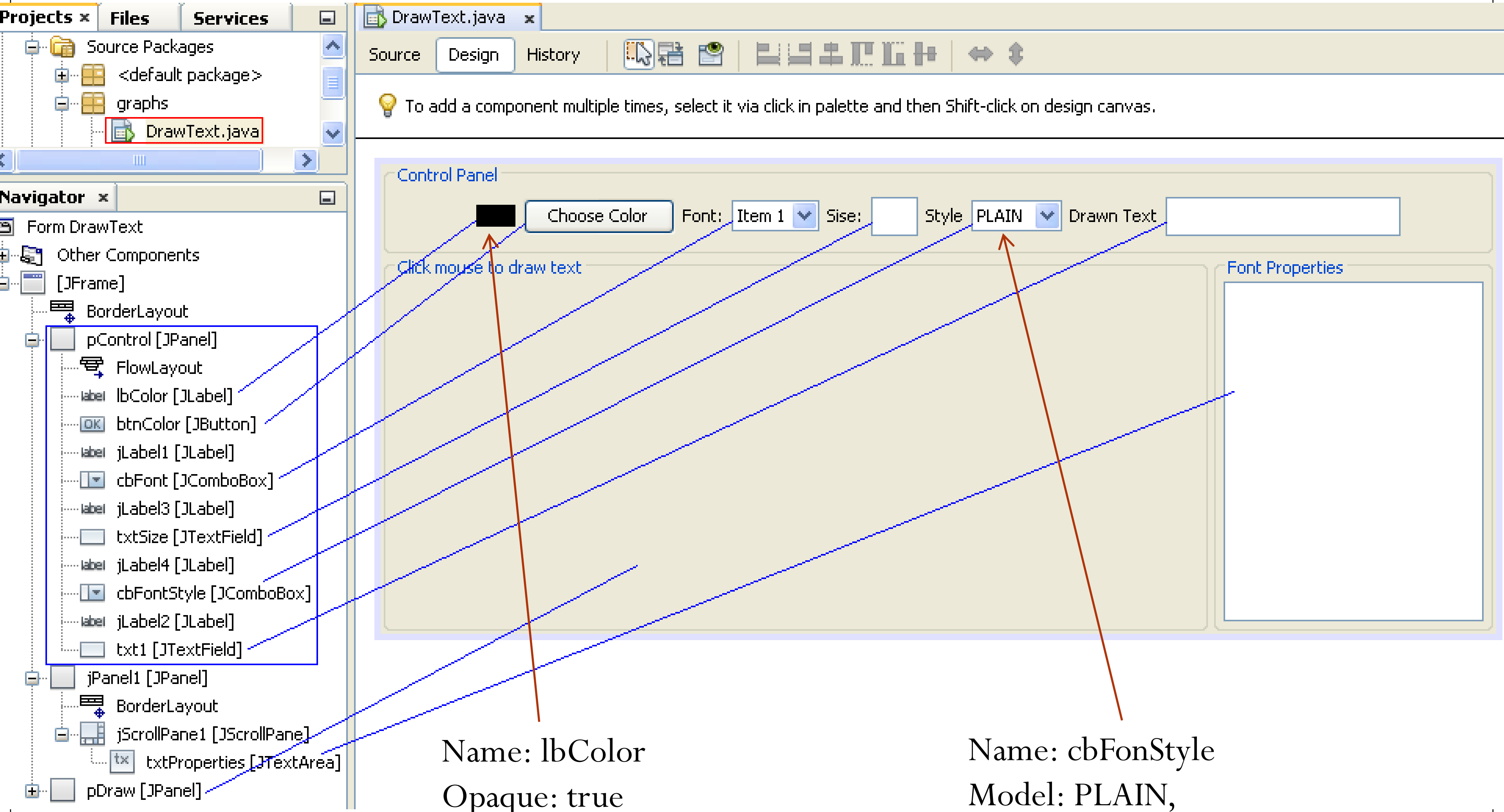
- Get system fonts
- Create a font object
- Assign a font object to graphics object
- Draw text on a component
- Get font metrics



Family: .TMC-Ong Do
Name: .TMC-Ong Do
Style: 2
Size: 60

Font metrics:
Ascent: 54
Max. Ascent: 54
Descent: 12
Leading: 0
Height: 66
Max. advance: 96
String width: 244

Demo 1: Design



Projects × Files Services

Source Packages
<default package>
graphs
DrawText.java

Navigator ×

Form DrawText
Other Components
[JFrame]
BorderLayout
pControl [JPanel]
FlowLayout
lbColor [JLabel]
btnColor [JButton]
jLabel1 [JLabel]
cbFont [JComboBox]
jLabel3 [JLabel]
txtSize [JTextField]
jLabel4 [JLabel]
cbFontStyle [JComboBox]
jLabel2 [JLabel]
txt1 [JTextField]
jPanel1 [JPanel]
BorderLayout
jScrollPane1 [JScrollPane]
txtProperties [JTextArea]
pDraw [JPanel]

DrawText.java ×

Source Design History

To add a component multiple times, select it via click in palette and then Shift-click on design canvas.

Control Panel

Choose Color Font: Item 1 Sise: Style PLAIN Drawn Text

Click mouse to draw text

Font Properties

Name: lbColor
Opaque: true
Text: Color
Background: Black

Name: cbFonStyle
Model: PLAIN,
BOLD, ITALIC

Demo 1: Code

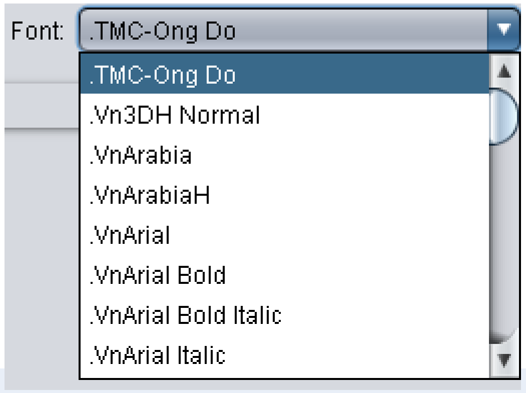
```
/* DrawText.java- Demo. for drawing text */
package graphs;
import java.awt.Color;
import java.awt.Font;
import java.awt.FontMetrics; // for getting font metric
import java.awt.GraphicsEnvironment; // for getting system fonts
import java.awt.Graphics;
import javax.swing.JColorChooser;

public class DrawText extends javax.swing.JFrame {
    Graphics g = null; // Graphcs object for drawing
    Color color= null; // current used color
    Font font= null;   // current used font
    int fontStyle = Font.PLAIN; // current font properies
    int fontSize= 15;
    String fontName= Font.SANS_SERIF;
    String drawnText=""; // text will be drawn
```

FPT Fpt University

TRƯỜNG ĐẠI HỌC FPT

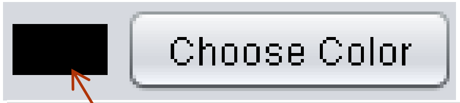
Demo 1: Code

```java
public DrawText() { // constructor
    initComponents();
    this.setSize(900, 400);
    loadFonts(); // load system fonts to combobox cbFont
    g= this.pDraw.getGraphics(); // get graphics object
    color= Color.BLACK; // set up default color
    g.setColor(color);
    // setup default font
    font= new Font(fontName, fontStyle, fontSize);
    this.showFontProperties(); // view font properties
}
private void loadFonts(){ // load system fonts to combobox
    // Get all system font installed
    GraphicsEnvironment env = GraphicsEnvironment.getLocalGraphicsEnvironment();
    Font[] fonts = env.getAllFonts();
    // load fonts to combobox cbFont
    this.cbFont.removeAllItems();
    String fName; // font name
    for (Font f: fonts)
    {   fName = f.getFontName();
        this.cbFont.addItem(fName);
    }
}
```

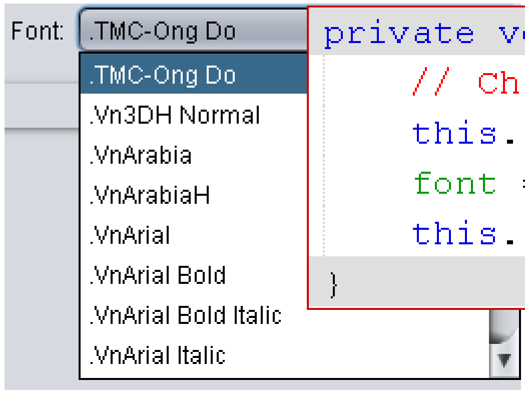

Font: .TMC-Ong Do

.TMC-Ong Do
.Vn3DH Normal
.VnArabia
.VnArabiaH
.VnArial
.VnArial Bold
.VnArial Bold Italic
.VnArial Italic

Demo 1: Code

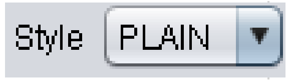


```
private void btnColorActionPerformed(java.awt.event.ActionEvent evt) {
    // Get color from color chooser
    Color newColor= JColorChooser.showDialog(this, "Choose a color", color);
    if (newColor!=null) color= newColor;
    lbColor.setBackground(color);
}
```

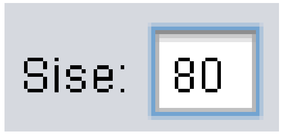


```
private void cbFontActionPerformed(java.awt.event.ActionEvent evt) {
    // Change font name
    this.fontName= ((String)cbFont.getSelectedItem());
    font = new Font(fontName, fontStyle, fontSize);
    this.showFontProperties();
}
```

Demo 1: Code



```
private void cbFontStyleActionPerformed(java.awt.event.ActionEvent evt) {
    // Change font style
    String str = (String)(cbFontStyle.getSelectedItem());
    if (str.equalsIgnoreCase("Bold")) fontStyle= Font.BOLD;
    else if (str.equalsIgnoreCase("Italic")) fontStyle= Font.ITALIC;
    else fontStyle= Font.PLAIN;
    font = new Font(fontName, fontStyle, fontSize);
    this.showFontProperties();
}
```

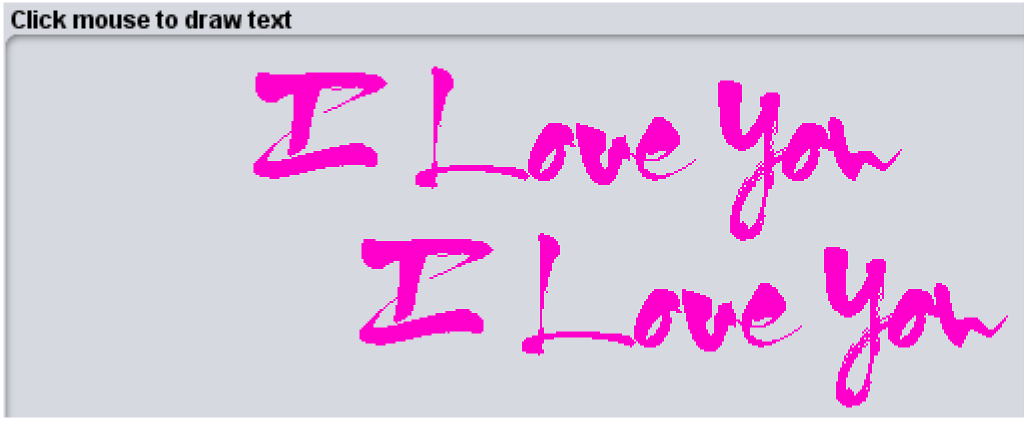


```
private void txtSizeFocusLost(java.awt.event.FocusEvent evt) {
    // Change font size
    fontSize= Integer.parseInt(txtSize.getText());
    font = new Font(fontName, fontStyle, fontSize);
    g.setFont(font);
    this.showFontProperties();
}
```

```
private void txtSizeActionPerformed(java.awt.event.ActionEvent evt) {
    // Change font size
    fontSize= Integer.parseInt(txtSize.getText());
    font = new Font(fontName, fontStyle, fontSize);
    g.setFont(font);
    this.showFontProperties();
}
```

When user enters preferred size then strikes the ENTER key

Demo 1: Code

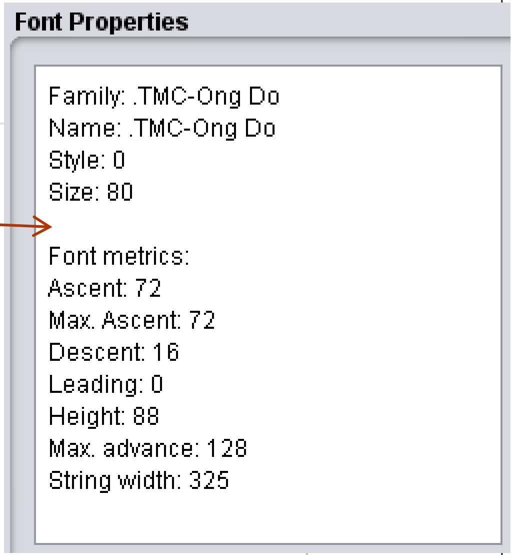


```java
private void pDrawMouseClicked(java.awt.event.MouseEvent evt) {
    // Draw string at the mouse position
    int x= evt.getX();
    int y= evt.getY();
    g.setFont(font);
    g.setColor(color);
    drawnText = txt1.getText();
    g.drawString(drawnText, x, y);
    showFontProperties();
}
```

TRƯỜNG ĐẠI HỌC FPT

Demo 1: Code

```
private void showFontProperties(){
    txtProperties.setText("");
    txtProperties.append("Family: " + font.getFamily()+ "\n");
    txtProperties.append("Name: " + font.getName()+ "\n");
    txtProperties.append("Style: " + font.getStyle() + "\n");
    txtProperties.append("Size: " + font.getSize() + "\n");
    if (g!=null) {
      FontMetrics fm = g.getFontMetrics();
      txtProperties.append("\nFont metrics:\n");
      txtProperties.append("Ascent: " + fm.getAscent() + "\n");
      txtProperties.append("Max. Ascent: " + fm.getMaxAscent() + "\n");
      txtProperties.append("Descent: " + fm.getDescent()+ "\n");
      txtProperties.append("Leading: " + fm.getLeading()+ "\n");
      txtProperties.append("Height: " + fm.getHeight() + "\n");
      txtProperties.append("Max. advance: " + fm.getMaxAdvance() + "\n");
      if (drawnText.length()>0)
      txtProperties.append("String width: " + fm.stringWidth(drawnText)+ "\n");
    }
}
```


Font Properties

```
Family: .TMC-Ong Do
Name: .TMC-Ong Do
Style: 0
Size: 80

Font metrics:
Ascent: 72
Max. Ascent: 72
Descent: 16
Leading: 0
Height: 88
Max. advance: 128
String width: 325
```

Summary

- Font Concepts
- Physical and Logical Fonts
- The java.awt.Font class
- Measuring Text
- Demonstrations

TRƯỜNG ĐẠI HỌC FPT

Thank You