Labs

Lab 10.1: Passing Strings to Applets

What is the purpose?

You will rewrite Listing 16.4, DisplayMessage.html, and Listing 16.5, DisplayMessage.java, on pages 547-548 of your book to display a message with a standard color, font, and size into an applet. The message, x, y, color, fontname, and fontsize are parameters in the <applet> tag, as shown in Figure 10-1-1.

```
<applet
  code = "DisplayMessage.class"
  width = 300
  height = 100
  alt="You must have a Java-enabled browser to view the applet"
>
  <param name = MESSAGE value = "Welcome to Java" />
  <param name = X value = 40 />
  <param name = Y value = 50 />
  <param name = COLOR value = "red" />
  <param name = FONTNAME value = "Monospaced" />
  <param name = FONTSIZE value = 20 />
</applet>
```

Figure 10-1-1

What are the steps?

- Task 1:

  Procedure

  1. Modify the DisplayMessage.html on page 547 with the code shown in Figure 10-1-2.

```
<param name = MESSAGE value = "Welcome to Java" />
<param name = X value = 40 />
<param name = Y value = 50 />
<param name = COLOR value = "red" />
<param name = FONTNAME value = "Monospaced" />
<param name = FONTSIZE value = 20 />
```

Figure 10-1-2

  2. Modify the DisplayMessage.java on page 547 with the code shown in Figure 10-1-3.

```
// Get parameter values from the HTML file
color = getParameter("COLOR");
fontName = getParameter("FONTNAME");
```

```
        fontSize = Integer.parseInt(getParameter("FONTSIZE"));

        messagePanel.setForeground(Color.red);

        messagePanel.setFont(new Font(fontName, Font.BOLD, fontSize));
```

Figure 10-1-3

3. Compile the DisplayMessage.java file using javac command.
4. Browse the DisplayMesssage.html in any Web browser to view the applet.
5. Save a screenshot of the output similar to Figure 10-1-4 and submit it to your instructor.

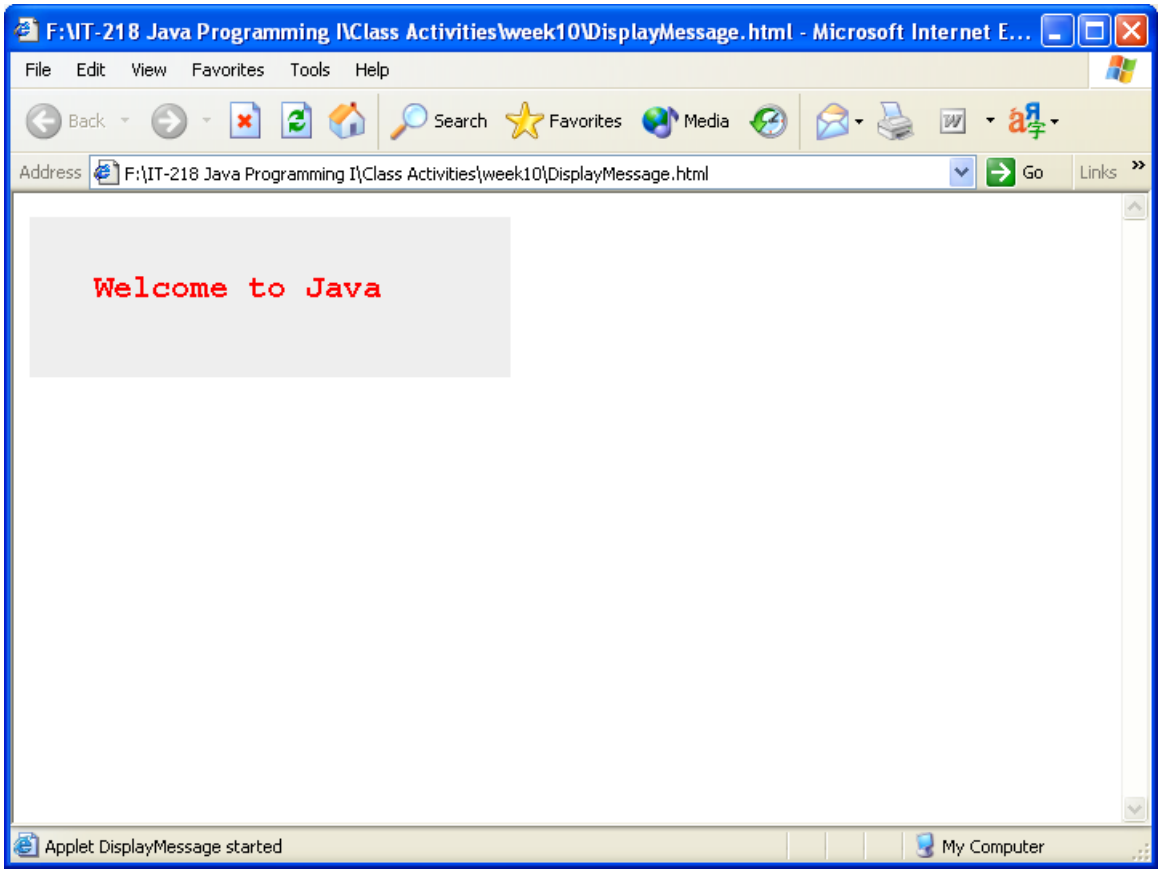

Figure 10-1-4

Did it work?

Were you able to—

- Embed DisplayMessage class into an applet?
- Display DisplayMessage class in an applet with DisplayMessage.html?
- Display a message with DisplayMessage.html?

Lab 10.2: Modified Tic Tac Toe

What is the purpose?

Rewrite the program in Listing 17.7, TicTacToe.java, on pages 572-577 of your book with the following modifications:

- Declare Cell as a separate class rather than an inner class.
- Add a button named New Game. The New Game button starts a new game.

What are the steps?

- Task 1:

Procedure

1. Modify the TicTacToe.java on pages 552-554.
2. Declare Cell as a separate class rather than an inner class. Replace the inner class Cell by the code shown in Figure 10-2-1 or create your own.

```
// Now it is a standalone class
class Cell extends JPanel implements MouseListener {

  // Reference to parent
  TicTacToe parent;

  public Cell(TicTacToe parent) {
    this.parent = parent;

  }
}
```

Figure 10-2-1

3. Add a button named New Game. The New Game button starts a new game.
4. Modify using code pictured in Figure 10-2-2 or create your own.

```
// Create a button to start a new game
private JButton jbtNew = new JButton("New Game");

// Place the panel and the label to the applet
this.getContentPane().add(jbtNew, BorderLayout.NORTH);

jbtNew.addActionListener(this);

public void actionPerformed(ActionEvent e) {
  if (e.getSource() == jbtNew) {

  }
```

```
}
```

Figure 10-2-2

5. Compile the TicTacToe.java file using the javac command.
6. Execute the TicTacToe.class using the java command.
7. Save screenshots of your output similar to Figures 10-2-3 and 10-2-4 and submit to your instructor.

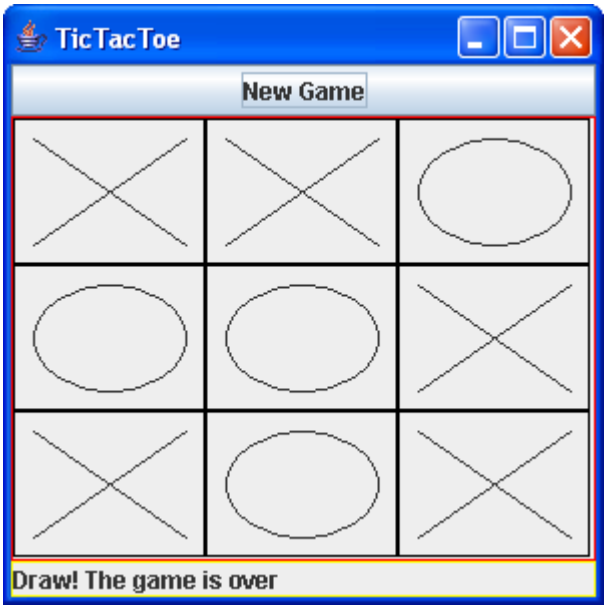

Figure 10-2-3

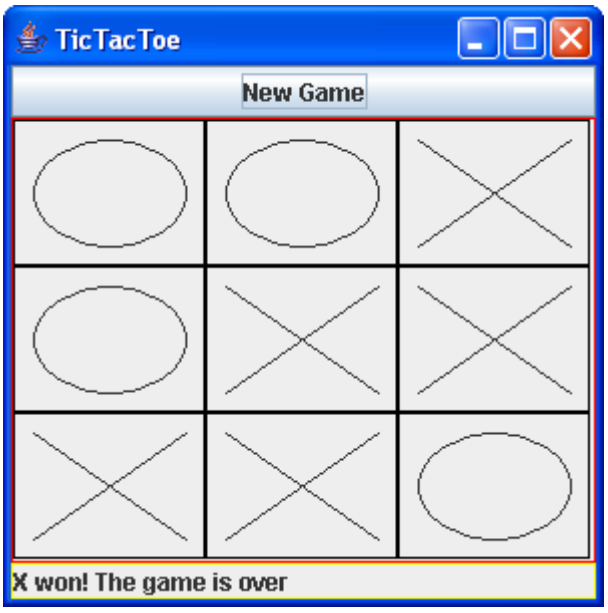

Figure 10-2-4

Did it work?

Were you able to—

- Implement the TicTacToe class with a standalone class instead of an inner class?
- Restart the game with a "New Game" button?