Lab Exercises

Lab Exercise 1 — Guess Game

Name: ______________________ **Date:** ______________________

Section: ______________________

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into six parts:

1. Lab Objectives
2. Description of Problem
3. Sample Output
4. Program Template (Fig. L 14.1 and Fig. L 14.2)
5. Problem-Solving Tips
6. Follow-Up Questions and Activities

The program template represents a complete working Java program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with Java code. Compile and execute the program. Compare your output with the sample output provided. Then answer the follow-up questions. The source code for the template is available at `www.pearsonhighered.com/deitel`.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 14 of *Java How to Program: 8/e*. In this lab, you will practice:

- Designing a GUI.
- Processing events.
- Creating and manipulating GUI components.

The follow-up questions and activities also will give you practice:

- Using various GUI methods to manipulate components.
- Adding additional components to a GUI.

Problem Description

Write an application that plays “guess the number” as follows: Your application chooses the number to be guessed by selecting an integer at random in the range 1–1000. The application then displays the following in a label:

```
I have a number between 1 and 1000. Can you guess my number?
Please enter your first guess.
```

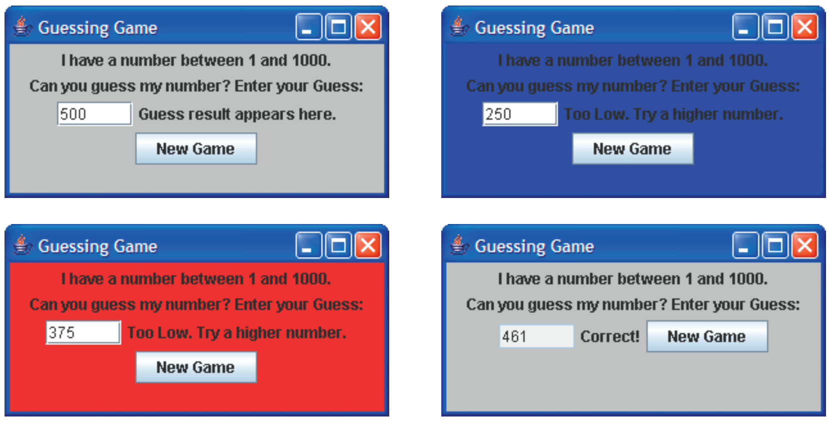
A `JTextField` should be used to input the guess. As each guess is input, the background color should change to either red or blue. Red indicates that the user is getting “warmer,” and blue indicates that the user is getting “colder.” A `JLabel` should display either `"Too High"` or `"Too Low"` to help the user zero in on the correct answer. When the user gets the correct answer, `"Correct!"` should be displayed, and the `JTextField` used for input should be changed to be uneditable. A `JButton` should be provided to allow the user to play the game again. When the `JButton` is clicked, a new random number should be generated and the input `JTextField` changed to be editable.

Lab Exercises

Name:

Lab Exercise 1 — Guess Game

Sample Output



I have a number between 1 and 1000.
Can you guess my number? Enter your Guess:
500 Guess result appears here.

I have a number between 1 and 1000.
Can you guess my number? Enter your Guess:
250 Too Low. Try a higher number.

I have a number between 1 and 1000.
Can you guess my number? Enter your Guess:
375 Too Low. Try a higher number.

I have a number between 1 and 1000.
Can you guess my number? Enter your Guess:
461 Correct!

Program Template

```java
// Lab Exercise 1 Solution: GuessGameFrame.java
// Guess the number
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Graphics;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.Random;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.JLabel;
import javax.swing.JButton;

public class GuessGameFrame extends JFrame
{
   private static Random generator = new Random();
   private int number; // number chosen by application
   private int guessCount; // number of guesses
   private int lastDistance; // distance between last guess and number
   private JTextField guessInputJTextField; // for guessing
   private JLabel prompt1JLabel; // first prompt to user
   private JLabel prompt2JLabel; // second prompt to user
   private JLabel messageJLabel; // displays message of game status
   private JButton newGameJButton; // creates new game
   private Color background; // background color of application

   // set up GUI and initialize values
   public GuessGameFrame()
   {
      /* Write a line of code that calls the superclass constructor and sets the title
         of this application to "Guessing Game" */

      guessCount = 0; // initialize number of guesses to 0
      background = Color.LIGHT_GRAY; // set background to light gray

```

Fig. L 14.1 | GuessGameFrame.java. (Part 1 of 3.)

Lab Exercises

Name:

Lab Exercise 1 — Guess Game

```java
        prompt1JLabel = new JLabel(
           "I have a number between 1 and 1000." ); // describe game
        prompt2JLabel = new JLabel(
           "Can you guess my number? Enter your Guess:" ); // prompt user

        guessInputJTextField = new JTextField( 5 ); // to enter guesses
        guessInputJTextField.addActionListener( new GuessHandler( ) );
        messageJLabel = new JLabel( "Guess result appears here." );

        /* Write a statement that creaters the "New Game" button */
        newGameJButton.addActionListener(

           new ActionListener() // anonymous inner class
           {
              public void actionPerformed( ActionEvent e )
              {
                 /* Write code that resets the application to an appropriate state
                    to start a new game. Reset the background color to light gray,
                    set the JTextFields to their initial text, call method
                    theGame and repaint the GuessGame JFrame */
              } // end method actionPerformed
           } // end anonymous inner class
        ); // end call to addActionListener

        /* Write code that will set the layout of the container to a Flowlayout,
           then add all the GUI components to the container */
      theGame(); // start new game
     } // end GuessGameFrame constructor

     // choose a new random number
     public void theGame()
     {
        /* Write a statement that sets instance variable number to a random number
           between 1 and 1000 */
     } // end method theGame

     // change background color
     public void paint( Graphics g )
     {
        super.paint( g );
        getContentPane().setBackground( background ); // set background
     } // end method paint

     // react to new guess
     public void react( int guess )
     {
        guessCount++; // increment guesses
        /* Write code that sets instance variable currentDistance to 1000. This
           variable's value will be used to determine if th ebackground color
           should be set to red or blue to indicate that the last guess was getting
           closer to or further from the actual number. */

        // first guess
        if ( guessCount == 1 )
        {
```

Fig. L 14.1 | `GuessGameFrame.java`. (Part 2 of 3.)

Lab Exercises

Name:

Lab Exercise 1 — Guess Game

```
            /* Write code to set instance variable lastDistance to the absolute value
               of the difference between variables guess and number. This value will
               be used with subsequent guesses to help set the background color. */

            if ( guess > number )
               messageJLabel.setText( "Too High. Try a lower number." );
            else
               messageJLabel.setText( "Too Low. Try a higher number." );
         } // end if
         else
         {
            /* Write code that sets instance variable currentDistance to the absolute
               value of the difference between variables guess and number. This
               variable's value will be compared with lastDistance to determine the
               background color. */

            // guess is too high
            if ( guess > number )
            {
               messageJLabel.setText( "Too High. Try a lower number." );

               /* Write code that sets Color variable background to red if the
                  currentDistance is less than or equal to lastDistance; otherwise,
                  set background to blue. Then assign currentDistance to lastDistance. */
            } // end if
            else if ( guess < number ) // guess is too low
            {
               messageJLabel.setText( "Too Low. Try a higher number." );
               background = ( currentDistance <= lastDistance ) ?
                  Color.RED : Color.BLUE;
               lastDistance = currentDistance;
            } // end else if
            else // guess is correct
            {
               messageJLabel.setText( "Correct!" );

               /* Write code that sets Color variable background to red if the
                  currentDistance is less than or equal to lastDistance; otherwise,
                  set background to blue. Then assign currentDistance to lastDistance. */
            } // end else

            repaint();
         } // end else
      } // end method react

      // inner class acts on user input
      class GuessHandler implements ActionListener
      {
         public void actionPerformed( ActionEvent e )
         {
            /* Write code that will obtain the guess, convert it to an int and
               pass that value to the react method */
         } // end method actionPerformed
      } // end inner class GuessHandler
   } // end class GuessGameFrame
```

Fig. L 14.1 | `GuessGameFrame.java`. (Part 3 of 3.)

Lab Exercises

Name:

Lab Exercise 1 — Guess Game

```
// Lab Exercise 1 Solution: GuessGame.java
// Guess the number
import javax.swing.JFrame;

public class GuessGame
{
   public static void main( String args[] )
   {
      GuessGameFrame guessGameFrame = new GuessGameFrame();
      guessGameFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      guessGameFrame.setSize( 300, 150 ); // set frame size
      guessGameFrame.setVisible( true ); // display frame
   } // end main
} // end class GuessGame
```

Fig. L 14.2 | `GuessGame.java`.

Problem-Solving Tips

1. Use methods from the `JTextField` class to manipulate all `JTextField` components. For instance, method `setText` will set the text of the text field, and method `setEditable` will set whether the text field can be edited or not.
2. Method `setBackground` from class `JFrame` sets the background color of the `JFrame`.
3. Use method `nextInt` from class `Random` to generate a random number from 1 to 1000. You will need to scale the range of values produced by `random` by 1000 and shift the range by 1.
4. Use variables `lastDistance` and `currentDistance` to determine the distance of the guess from the actual number. If this distance gets larger between guesses, set the background color of the `JFrame` to blue. If this distance gets smaller or stays the same, set the background color to red.
5. If you have any questions as you proceed, ask your lab instructor for assistance.