**Quiz 1)** Java programs go through 5 phases: Edit, Compile, Load, Verify, Execute. The class loader reads: .class files. Java is compiled into: ByteCodes. Suppose someone creates a Java class called HelloWorld and names the file HelloWorldEx.java. What is the problem? The class name must match the filename so this won’t work. The process of building an object of a class before a program can perform the tasks that the class’s methods define is called: Instantiation. If you want the user to provide input to your application via the keyboard: Use the Scanner class methods from Java’s class libraries. An import declaration: Is used to help the compiler locate a class that’s used in an application. Class names in Java, by convention: Start with a capital letter and use camelcase. Java’s API: Contains a set of predefined class, is also called that Java class library, and Groups related classes into packages. Which of the following is TRUE about the Java virtual machine: It contains the Java Runtime Engine, the APIs, and the Java class libraries. **Quiz 2)** In a Swing application using BorderLayout, which of the following is the correct statement to place a label in the top area of a panel? myPanel.add(lblGreet,BorderLayout.NORTH). Which of the following is an appropriate class statement for a Swing application? Class Ex1 extends JFrame. In a Swing application, components are added: to the content pane. Java Swing has several top-level container classes.  These include: JFrame, JDialog, JApplet, NOT JPane. Which of the following is NOT TRUE about JFrame’s ‘EXIT\_ON\_CLOSE’? It is: a field, a constant, sets default behavior when a frame is closed, it is NOT a method. In the following statement, the JFrame on the right side of the assignment statement: invokes the constructor. Java programs that use Swing must have at least two top-level containers – a frame and a content pane: False. Every top-level container: can have a menu bar AND has a content pane. What must be done when you want to handle an event from the user (i.e. a click or double-click)? Add “implements EventListener” to the class heading. **Quiz 3)** The constructor of a class: is required to be called for every object that is created. Which of the following is the appropriate entry in a UML diagram for a default constructor of class Employee? <<constructor>> Employee(name: String). If a class is written without a constructor: the compiler provides a default constructor with parameters to initialize the members of the class. Suppose a class called Employee contains 3 data members – ID, name, and salary.  What can be said about those when an instance of Employee called e1 is created? ID and name are initialized automatically to the null string, and salary is initialized to 0.0 automatically. Which of the following are TRUE about Java’s double type? they require twice as much memory as float variables, the provide 15 significant digits, the compiler treats floating-point literals in the source code as double by default, NOT in contrast to floating point numbers, doubles are always precise. Which of the following is the appropriate entry in a UML diagram for a data member of class Employee named salary that is encapsulated within the class? +salary(name : Employee). If the programmer provides a constructor to a class, the compiler does not provide a default constructor: True. Which of the following is TRUE about the code shown? public static void main( String[ ] args){ int x; }: x is NOT an instance variable, NOT initialized to 0 by default, NOT a local variable initialized to 0 by default. Members of a class are typically: declared private. For precise floating-point numbers, Java provides a class in the package java.math called: BigDecimal.

**import** java.io.\*;

**class** PrintFile {

**public** **static** **void** main(String[] args) {

**for** (**int** ct=0; ct < args.length; ct++)

{*displayFile*(args[ct]);

}//end for

}//end of main method

**private** **static** **void** displayFile(String fileName)

{ System.*out*.println(fileName + ":");

System.*out*.println();

FileReader myFile = **null**; //NOTE--important to declare this OUTSIDE of try block

**try**

{myFile = **new** FileReader(fileName); **int** c;

**while** ((c = myFile.read()) != -1)

{ System.*out*.print((**char**)c);

} //end while

}//end try

**catch** (FileNotFoundException noFoundEx)

{ System.*out*.println("Could not open " + fileName);

}

**catch** (IOException ioEx)

{ System.*out*.println("Error reading from " + fileName);

}

**finally**

{ **if** (myFile != **null**)

{ **try** { myFile.close(); }

**catch** (IOException ioEx) {;}

}

} //end finally

} //end of displayFile method

}//end PrintFile

import javax.swing.AbstractButton;

import javax.swing.JButton;

import javax.swing.JPanel;

import javax.swing.JFrame;

import javax.swing.ImageIcon;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

/\*

\* ButtonDemo.java requires the following files:

\* images/right.gif

\* images/middle.gif

\* images/left.gif

\*/

public class ButtonDemo extends JPanel

implements ActionListener {

protected JButton b1, b2, b3;

public ButtonDemo() {

ImageIcon leftButtonIcon = createImageIcon("images/right.gif");

ImageIcon middleButtonIcon = createImageIcon("images/middle.gif");

ImageIcon rightButtonIcon = createImageIcon("images/left.gif");

b1 = new JButton("Disable middle button", leftButtonIcon);

b1.setVerticalTextPosition(AbstractButton.CENTER);

b1.setHorizontalTextPosition(AbstractButton.LEADING); //aka LEFT, for left-to-right locales

b1.setMnemonic(KeyEvent.VK\_D);

b1.setActionCommand("disable");

b2 = new JButton("Middle button", middleButtonIcon);

b2.setVerticalTextPosition(AbstractButton.BOTTOM);

b2.setHorizontalTextPosition(AbstractButton.CENTER);

b2.setMnemonic(KeyEvent.VK\_M);

b3 = new JButton("Enable middle button", rightButtonIcon);

//Use the default text position of CENTER, TRAILING (RIGHT).

b3.setMnemonic(KeyEvent.VK\_E);

b3.setActionCommand("enable");

b3.setEnabled(false);

//Listen for actions on buttons 1 and 3.

b1.addActionListener(this);

b3.addActionListener(this);

b1.setToolTipText("Click this button to disable the middle button.");

b2.setToolTipText("This middle button does nothing when you click it.");

b3.setToolTipText("Click this button to enable the middle button.");

//Add Components to this container, using the default FlowLayout.

add(b1);

add(b2);

add(b3);

}

public void actionPerformed(ActionEvent e) {

if ("disable".equals(e.getActionCommand())) {

b2.setEnabled(false);

b1.setEnabled(false);

b3.setEnabled(true);

} else {

b2.setEnabled(true);

b1.setEnabled(true);

b3.setEnabled(false);

}

}

/\*\* Returns an ImageIcon, or null if the path was invalid. \*/

protected static ImageIcon createImageIcon(String path) {

java.net.URL imgURL = ButtonDemo.class.getResource(path);

if (imgURL != null) {

return new ImageIcon(imgURL);

} else {

System.err.println("Couldn't find file: " + path);

return null;

}

}

/\*\*

\* Create the GUI and show it. For thread safety,

\* this method should be invoked from the

\* event-dispatching thread.

\*/

private static void createAndShowGUI() {

//Create and set up the window.

JFrame frame = new JFrame("ButtonDemo");

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

//Create and set up the content pane.

ButtonDemo newContentPane = new ButtonDemo();

newContentPane.setOpaque(true); //content panes must be opaque

frame.setContentPane(newContentPane);

//Display the window.

frame.pack();

frame.setVisible(true);

}

public static void main(String[] args) {

//Schedule a job for the event-dispatching thread:

//creating and showing this application's GUI.

javax.swing.SwingUtilities.invokeLater(new Runnable() {

public void run() {

createAndShowGUI();

}

});

}

}

Bottom of Form