CSC 171 LAB 10

Lab Instructions

The goal of this lab is to give you practice with the final modifier, GUI events, throwing exceptions, handling exceptions.

For this lab, implement all of the instructions below in one or more appropriately named packages. For the exercises in this lab it is acceptable to implement your “test” as a main method within the same class.

1. A class to represent a Car would have many different instance variables, including a unique VIN (vehicle identification number), paint color, mileage, interior color, number of valves in the engine, and so on. Design your own class for a car that includes variables for at least those features. Use the final modifier as appropriate to prevent some of the instance variables from being reassigned after they are assigned a value. Include comments for each variable to indicate why you decided to make it final, or why you decided not to. Create the necessary constructors, and provide accessors and mutators as appropriate.

2. It’s the last lab, so let’s do something with a checkerboard! In Lab06 you created classes called Square and Checkerboard. Dig that code out (or use the code at the end of this lab) and update it so that final is used to modify instance variables in both classes. Include comments for each variable to indicate why you did or did not decide to make it final. Your code should still compile and function properly when you have completed the modifications.

3. Implement a simple GUI that includes a frame with two buttons and a label. The text on the button should say “Drink me!” and the text on the other should say “Eat me!” The label should be blank by default. If the first button is pressed, the label should change to say “You got smaller!” If the second button is pressed, the label should change to say “You got bigger!”

4. Begin by copying and pasting your code from section 3 into a new class. Remove the label from your UI. Now, when the user clicks the “Eat me!” button, increase the height and width of your frame by 10%. If the user clicks “Drink me!” decrease the height and width of your frame by 10%.

5. Implement a simple UI with a text field, a button, and a label. Use the layout of your choice. The text on the button should read “Clone”. When the the button is clicked, set the text on the JLabel to be whatever has been typed into the text field.

6. Implement a simple UI that has the following components: a label that says “Choose your favorite color”, at least 5 radio buttons that present color choices, a button that says “Choose!”, and a blank label. Use a ButtonGroup to insure that the user is only able to choose one color at a time. When the “Choose!” button is pressed, update the blank label with a string similar to “You chose the color RED!” (replace “RED” with the color that the user choose).

7. In Lab 09 you created a class called “Bullseye” that extended JPanel. Reuse that code here. In the event that you lost it (or never wrote it) you may refer to the paint method at the end of this lab. Use an appropriate layout manager to add your Bullseye to a frame and a label either above it or below it. Then implement an event listener that responds to mouse clicks. If the mouse is clicked in the bullseye (the red circle), update the text on the label to say “BULLSEYE!” Otherwise, update the text on the label to say “MISSED!”

8. Write a method that accepts a String filename as a parameter. Use the filename to create an instance of the **`java.io.FileInputStream`** class. The constructor of this class will throw an exception if a file with the specified name does not exist. Catch the exception and print a message saying “file does not exist.” If no exception is thrown, print a message that says “file found.” **Note**: by default Java looks for files in the working directory (the directory from which you ran the program). In Eclipse, this is the root folder of your project.

9. Start by copying and pasting the code from section 8. Modify your code to initially assign a **`null`** value to the **`FileInputStream`** outside of your **`try`** block. Rethrow the exception rather than catching it. Add a **`finally`** block that calls the close method on the **`FileInputStream`** *if it is not null*.

10. Recall the “DivideByZeroException” used as an example in class that is thrown when the “divide” method is called on a simple calculator with a denominator of 0. Create a custom exception of your own invention that extends the **`java.lang.Exception`** class. In a separate class, write a method that throws the exception under some condition. Write a test that demonstrates calling the method once when the exception is thrown (print the stack trace of the exception), and once when the exception is not thrown.

HAND IN

Before handing in, create two additional files in your lab directory:

1. Create a README that contains the following:
    a. Your contact information (name, class, lab session), TA name, and assignment number.

   b. A brief (one paragraph at most) description of the assignment.
   c. Instructions explaining how to run your code. While it is possible that some TAs will compile, run, and test your code in Eclipse, it is also possible that they will want to compile and run it from the command line. You should include instructions on how to compile and run each of the executable Java classes.
2. Create a file titled "SampleOutput" showing the results of running your code. (You may copy and paste from the Eclipse console). If appropriate, please include a comment above the output for each section (e.g. "#OUTPUT FOR SECTION 1") and put a few blank lines between each section.

Hand in by uploading the compressed (i.e. "zipped") folder containing your assignment files to Blackboard.

```
public class Square {
	private String color;
	private boolean occupied;

	public String getColor() {
		return color;
	}

	public boolean getOccupied() {
		return occupied;
	}

	public void setOccupied( boolean occupied ) {
		this.occupied = occupied;
	}

	public void setColor( String color ) {
		this.color = color;
	}

	public String toString() {
		return "|" + getColor().charAt( 0 ) +
			(occupied ? "*" : " ") + "|";
	}
}

public class Checkerboard {
	private Square[][] squares;

	public Checkerboard( int rows, int cols ) {
        squares = new Square[rows][cols];

        for( int i=0; i<rows; i++ ) {
            for( int j=0; j<cols; j++ ) {
                int sequence = i + j;
                boolean even = ( sequence % 2 ) == 0;
                String color = ( even ? "Black" : "Red" );
                boolean occupied =
                    ( even && ( i < 3 || i >= rows - 3 ));

                Square square = new Square( color, occupied );
                squares[i][j] = square;
            }
        }
    }

    public static void main( String[] args ) {
        Checkerboard board = new Checkerboard( 8, 8 );
        System.out.println( board );
    }

    public Square[][] getSquares() {
        return squares;
    }

    public void setSquares( Square[][] squares ) {
        this.squares = squares;
    }

    @Override
    public String toString() {
        String output = "";
        for( int i=0; i<squares.length; i++ ) {
            output = output + rowToString( squares[i] ) + "\n";
        }
        return output;
    }

    private String rowToString( Square[] row ) {
        String output = "";
        for( Square square : row ) {
            output = output + square;
        }
```

```
            return output;
        }
    }

    // paint method for Bullseye
    @Override
    public void paint( Graphics g ) {
        Dimension dim = getSize();
        g.setColor( Color.blue );
        int width = dim.width;
        int height = dim.height;
        g.fillRect( 0, 0, width, height );
        int wchange = width / 20;
        int hchange = height / 20;
        int x = 0;
        int y = 0;
        boolean black = true;

        for( int i=0; i<9; i++ ) {
            g.setColor( black ? Color.black : Color.white );
            black = !black;

            g.fillOval( x, y, width, height );
            x = x + wchange;
            y = y + hchange;
            width = width - wchange * 2;
            height = height - hchange * 2;
        }

        g.setColor( Color.red );
        g.fillOval( x, y, width, height );
    }
```