Have you started PSA6?
A. I haven’t looked at it at all
B. I’ve started: I’ve read over the assignment and started thinking about it
C. I’ve read over the assignment AND THE CODE and started thinking about it
D. I’ve started writing some code
E. I’m done (moving on to extra credit)

Inner Classes in a simple GUI
1. An anonymous inner class is a nameless class that combines defining an inner class and instantiating it all in one step. True or false?

A. True, an anonymous inner class does all of the above.

B. False, an anonymous inner class must be instantiated in another step.
2. Why must you implement ALL of the methods in the WindowListener, even though you may not care about some of them?

A. All of the methods in WindowListener only return null, so they must be implemented.

B. They are all static methods, so they must be implemented.

C. They are all abstract because WindowListener is an interface, so they must be implemented.

D. You don’t have to implement the methods in WindowListener at all.
3. What kind of event is fired when you type on your keyboard?

A. MouseEvent

B. KeyEvent

C. WindowEvent

D. ButtonEvent
Making our GUI more interesting

Each oval represents a “space” in a 3-position “board”
Yellow indicates that no one has played.

If it is ‘X’s turn, clicks should turn the ovals blue
If it is ‘O’s turn, clicks should turn the ovals green

Turns will switch only when the user clicks “Switch turns”

Idea: We will create a new instance variable “board” and the Panels will paint based on the contents of the “board” variable
public class ConnectFourSimpleDemo extends JFrame {
    private char turn;
    private char[] board;
    private JLabel status;

    public ConnectFourSimpleDemo() {
        this.turn = 'X';  // Initialize turn to X
        this.board = new char[3];  // Initialize the board (empty)
        this.board[0] = ' ';  
        this.board[1] = ' ';  
        this.board[2] = ' ';

        JPanel displayBoard = makeDisplayBoard();
        JPanel buttonPanel = makeButtonPanel();

        this.status = new JLabel( "Welcome to Connect 4!  Turn is: " + this.turn );

        setLayout( new BorderLayout() );
        add( this.status, BorderLayout.NORTH );
        add( displayBoard, BorderLayout.CENTER );
        add( buttonPanel, BorderLayout.SOUTH );

        pack();
        setVisible( true );
    }

    Our method was too long,
    so I added helper methods
}
public class ConnectFourSimpleDemo extends JFrame {
    // ConnectFourSimpleDemo is a completely separate class, even though
    // both classes are defined in the same file.
}

class MyPanel extends JPanel {
    private int position;

    public MyPanel( int pos )
    {
        position = pos;
    }

    protected void paintComponent( Graphics g )
    {
        super.paintComponent(g);
        if ( board[position] == 'X' )
            g.setColor( Color.blue );
        else if ( board[position] == 'O' )
            g.setColor( Color.green );
        else
            g.setColor( Color.yellow );
        g.fillOval( 0, 0, getWidth(), getHeight() );
    }
}

Will this work?
A. Yes
B. No, because you cannot access the
   ConnectFourSimpleDemo
   member variable board from the MyPanel class
C. No, because the MyPanel
   constructor does not explicitly call the constructor of the
   superclass (JPanel)
ConnectFourSimpleDemo object (scope)

- **turn**: ‘X’
- **status**: Address of the JLabel object
- **board**: 

MyPanel object (scope)

- **position**: 

No connection between MyPanel object and CFSD object!
Now if Java can’t find a variable in the MyPanel object, it will look for it in the CFSD object!

Do not confuse this with subclasses! The MyPanel object exists within the scop of the CFSD object, but there is no subclass relation. The board variable is still in the CFSD object, NOT the MyPanel object.
Now if Java can’t find a variable in the MyPanel object, it will look for it in the CFSD object!

Do not confuse this with subclasses! The MyPanel object exists within the scope of the CFSD object, but there is no subclass relation. The board variable is still in the CFSD object, NOT the MyPanel object. A MyPanel exists within a CFSD, but it is NOT a CFSD.
public class ConnectFourInnerDemo extends JFrame {

    // CFID defined here

    class MyPanel extends JPanel {
        private int position; // A new variable in MyPanel
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            if (board[position] == 'X')
                g.setColor(Color.blue);
            else if (board[position] == 'O')
                g.setColor(Color.green);
            else g.setColor(Color.yellow);
            g.fillOval(0, 0, getWidth(), getHeight());
        }
        // other MyPanel methods defined here
    }

    class PanelClickListenerInner implements MouseListener {
        public void mouseClicked(MouseEvent e) {
            // Other methods defined here
        }
    }
}
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    if (board[position] == 'X')
        g.setColor(Color.blue);
    else if (board[position] == 'O')
        g.setColor(Color.green);
    else
        g.setColor(Color.yellow);
    g.fillOval(0, 0, getWidth(), getHeight());
}
What should go in the blank to make the panel turn blue when the user clicks on it and it’s X’s turn?

A. board[position] = turn;
B. this.board[this.position] = this.turn;
C. g.setColor(Color.blue);
D. if ( turn == ‘X’ ) g.setColor(Color.blue);
E. Something else
protected void paintComponent( Graphics g )
{
    super.paintComponent(g);
    if (board[position] == 'X' )
        g.setColor( Color.blue );
    else if ( board[position] == 'O' )
        g.setColor( Color.green );
    else
        g.setColor( Color.yellow );
    g.fillOval(0, 0, getWidth(), getHeight());
}

public void mouseClicked( MouseEvent e )
{
    repaint(); // Java automatically calls paintComponent.
    // calling object(restrict)
    repaint(); // where is g defined?
}

A. board[position] = turn;
B. this.board[this.position] = this.turn;
C. g.setColor( Color.blue ); // There's no
D. if ( turn == 'X' ) g.setColor( Color.blue );
E. Something else
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    if (board[position] == 'X')
        g.setColor(Color.blue);
    else if (board[position] == 'O')
        g.setColor(Color.green);
    else
        g.setColor(Color.yellow);
    g.fillOval(0, 0, getWidth(), getHeight());
}

public void mouseClicked(MouseEvent e) {
    board[position] = turn;
    repaint();
}
public class ConnectFourInnerDemo extends JFrame {
    // CFID defined here
    class MyPanel extends JPanel {
    private int position;  // A new variable in MyPanel
    // MyPanel methods defined here

    class PanelClickListenerInner implements MouseListener {
        public void mouseClicked( MouseEvent e ) {
            board[position] = turn;
            repaint();
        }
        // Other methods defined here
    }
    // Other methods defined here

    Why do we need to call repaint in the mouseClicked method? (discuss)
Adding the PanelClickListener

ConnectFourSimpleDemo object (extends JFrame)

board

Address of the board array

MyPanel object (extends JPanel)

PanelClickListener object
(implements MouseListener)

public void mouseClicked( ... )

position

0

status

Address of the JLabel object

Why won’t this code work?
A. It will work fine
B. The PanelClickListener must be instantiated inside a MyPanel object
C. A PanelClickListener is not a MouseListener
D. PanelClickListener objects cannot be instantiated using new because PanelClickListener is an interface
public class ConnectFourInnerDemo extends JFrame {
    // CFID defined here
    class MyPanel extends JPanel
    {
        // MyPanel defined here
    }

    // We will register this listener with the reset button
    class ResetListener implements ActionListener
    {
        public void actionPerformed( ActionEvent e )
        {
            // You will implement
        }
    }

    // We will register this listener with the switch turns button
    class SwitchListener implements ActionListener
    {
        public void actionPerformed( ActionEvent e )
        {
            // You will implement
        }
    }
}
class ResetListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        // See Friday’s code provided on the website
    }
}

class SwitchListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        // In CFID:
        private char turn;
        private char[] board;
        private JLabel status;
    }
}