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)

START early! REMEMBER there are NOT a lot of tutor hours Tuesday/Wednesday (labs are full)

MORE TIPS for PSA6 + APPLETS
1. What interface do we need to implement in a class to let objects of that class be a listener for an event?

A. ActionListener
B. JListener
C. JPanel
D. JFrame
E. Comparable
2. What is an Anonymous class?

A. An abstract class with no methods.
B. An inner class that doesn’t have a name.
C. A class where all variables and methods are private.
D. A class where all variables and methods are static.
3. Do we have to implement ALL of the methods specified in the WindowListener, even though you may not care about some of them?

A. Yes, we should implement all of them, because we should always care about all possible events.

B. Yes, we have to implement all of them, because WindowListener is an interface and they are all abstract methods.

C. Yes, they are all static methods, so they must be implemented.

D. No. If you don’t want to use some of those options, you don’t have to implement them.

E. No. You don’t have to implement the methods in WindowListener at all.
4. What kind of event is fired when you click on your mouse?

A. KeyEvent

B. MouseEvent

C. ButtonEvent

D. WindowEvent
Making our GUI more interesting

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

1. If it is ‘X’'s turn, clicks should turn the ovals **blue**
2. If it is ‘O’s turn, clicks should turn the ovals **green**
3. Turns will switch only when the user clicks “Switch turns”

*Idea: We create a new instance variable “board” and the Panels will paint based on the contents of the “board” variable.*

WE WANT SOME CONNECTION between what happens in some Components and what is shown in other Components
public class ConnectFourSimpleDemo extends JFrame {
    // ConnectFourSimpleDemo is a completely separate class, even though
    // both classes are defined in the same file.
} END of ConnectFourSimpleDemo

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)
IF we DEFINE CLASS CFSDemo and MyPanel separately:
No connection between MyPanel object and CFSD object!

I could pass a LOT of parameters to the constructor...
instead of just  

public MyPanel( int pos )

BUT . . . THERE IS A BETTER SOLUTION (next slide)
ConnectFourSimpleDemo object

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

MyPanel object

- **position**: 

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.
ConnectFourInnerDemo object

- turn: 'X'
- status: Address of the JLabel object
- board

MyPanel object

- position: 0
- PanelClickListener object

public void mouseClicked(
  // WRITES data to board
  ...
)

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());
}

// method from MyPanel object

// READS data from board

How many “inner” levels??
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 PanelClickListener implements MouseListener {
        public void mouseClicked( MouseEvent e ) {
            ______________________________;
            repaint();  // HINT: repaint calls to paintComponent
        }
        // We will register this listener
        // with the MyPanel objects
    }

    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;  \textit{WRITES data to board}
    B. g.setColor( Color.blue );
    C. if ( turn == ‘X’ ) g.setColor( Color.blue );
    D. 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();
}

From the PanelClickListener I can "see" both the fields from MyPanel object and CFID object 😊

CAREFUL! They are their fields, not mine... I can NOT use this
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 PanelClickListener implements MouseListener {
            public void mouseClicked(MouseEvent e) {
                board[position] = turn;
                repaint();
            }
            // WARNING: We need to register this listener
            // with the MyPanel objects !!

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

**ConnectFourInnerDemo object (extends JFrame)**

1st step: CREATE CODE ...

2nd step: CREATE ONE INSTANCE of our “LISTENER”

3rd step: add the listener to the component

```java
public ConnectFourInnerDemo()
{
    ...
    JPanel displayBoard = new JPanel();
    JPanel p1 = new MyPanel();
    p1.addMouseListener(new PanelClickListener());
    ...
}
```

Why won’t this code work properly?

A. The PanelClickListener must be instantiated inside a MyPanel object
B. A PanelClickListener is not a MouseListener
C. PanelClickListener objects cannot be instantiated using new because PanelClickListener is an interface

**MyPanel object (extends JPanel)**

(implies MouseListener)

public void mouseClicked( ... )

**board**

Address of the board array

**status**

Address of the JLabel object

**turn**

‘X’

**board**

Address of the board array

**MyPanel object (extends JPanel)**

Instantiate the listener INSIDE one object of the class where the listener class is defined (e.g., in the constructor. Remember the advantages of doing that there – ALL objects of that class are “born” with the listener already associated)
Adding the ActionListener for the buttons

public class ConnectFourInnerDemo extends JFrame {
    // CFID defined here
    class MyPanel extends JPanel //inner class of CFIDemo
    { // MyPanel defined here } //END OF MYPANEL

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

    // We will register this listener with the Switch turns button
    class SwitchListener implements ActionListener
    { // ANOTHER inner class of CFIDemo
        public void actionPerformed(ActionEvent e )
        {
            // You will implement
        }
    }
}
class ResetListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        // . . .
    }
}

class SwitchListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        // SOLUTION WE DID IN CLASS:
        if (turn == 'X'){
            turn = 'O';
        } else{
            turn = 'X';
        }
        status.setText("  Turn is: "+turn);
        repaint();
    }
}

NOTES: ActionListener How many methods do I need to implement?
http://docs.oracle.com/javase/7/docs/api/java/awt/event/ActionListener.html

class SwitchListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        // SOLUTION WE DID IN CLASS:
        if (turn == 'X'){
            turn = 'O';
        } else{
            turn = 'X';
        }
        status.setText("  Turn is: "+turn);
        repaint();
    }
}

Because they INNER class of CFIDemo, I can see these variables:
private char turn;
private char[] board;
private JLabel status;
public class ConnectFourInnerDemo extends JFrame {
    // CFID defined here
    ... 
    // registering the listener ...
    // COULD I USE ANONYMOUS CLASS INSTEAD?
    jbtSwitch.addActionListener( new SwitchListener() );

    class MyPanel extends JPanel //inner class of CFIDemo
    { // MyPanel defined here } //END OF MYPANEL

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

    // We will register this listener with the Switch turns button
    class SwitchListener implements ActionListener
    { // ANOTHER inner class of CFIDemo
        public void actionPerformed(ActionEvent e){
            // You will implement
        }
    }

    IN PSA6, BEFORE IMPLEMENTING ANYTHING THINK if IT’s ALREADY DONE IN ANY OF YOUR PSA4 (ConnectFourBoard) CLASS!
    Remember you’ll use a ConnectFourBoard NOT the char[] in these examples
PSA 6 summary

Only one new method with regard to PSA4 (returns the char in that position of the board) – THIS IS INCLUDED in our solution .class
char getContents (int row, int column)

The MAIN elements are “similar” to the example in class
• ConnectFour extends JFrame: it’s your “canvas”, where you’ll add everything

• BoardCell
  • extends JPanel: it’ll represent one of the “cells” of the game (ovals/circles...)
  • it’s an inner class of ConnectFour

• PlayListener
  • implements MouseListener: manages what happens when you click in a BoardCell
  • it’s an inner class of BoardCell
Next week – Key events - KeyListener

Applets and Graphic letters!
public class AppletDemo extends JApplet {

    /** A message that will reflect what the user types */
    private JLabel message;
    private JTextField textfield;

    public void init() {
        System.out.println( "Initializing the applet" );
        message = new JLabel( "Type something in the field below" );
        message.setFont( new Font( "Serif", Font.PLAIN, 40 ) );
        textfield = new JTextField();
        textfield.addKeyListener( new MessageListener() );

        this.setLayout( new BorderLayout() );
        this.setSize(new Dimension(700,500));
        add( message, BorderLayout.CENTER );
        add( textfield, BorderLayout.SOUTH );
        setVisible( true );
    }

    Sketch what this applet will look like.
Once it’s FINISHED and working, you can run it with:
> appletviewer index2.html
DO NOT DEBUG IN THE BROWSER!!
Figure 16.14 The **KeyEvent** class encapsulates information about key events.

**KeyEvent**

(java.awt.event.KeyEvent)

- `getKeyChar() : char`
- `getKeyCode() : int`

Returns the character associated with the key in this event.

Returns the integer key code associated with the key in this event.
class MessageListener implements KeyListener
{
    public void keyPressed( KeyEvent e ) { }  
    public void keyReleased( KeyEvent e ) { }  
    public void keyTyped( KeyEvent e )
    {
        _______________________________
    }
}

class AppletDemo
{
    // in the AppletDemo class:
    // There is a JLabel named message here.
    // We associate the listener to the textField (where the user is going to type)
    textField.addKeyListener( new MessageListener() );

    Which line of code goes in the blank to complete the Applet? (MessageListener is an inner class of AppletDemo)

    A. message.setText( "You typed a " + e.getKeyChar() );
    B. System.out.println("You typed a " + e.getKeyChar() );
    C. textField.keyTyped();
    D. textField.setText("You typed a " + e.getKeyChar() );
Other than the fact that this is an Applet (no big deal, just a “different canvas”) and we’re using KeyListeners you’ve done all this before!