Withdraw deadline today
Check your grades
• Most of you are doing GREAT!
• If you have:
  • > 8 missed classes OR
  • < 60% average OR
  • < 50% exam average
You might consider withdrawing and trying again in the future.

What does the following code print?
```
int k = 4;
System.out.println( 1/k + ", " + (double)(1/k) );
```
A. 1/4, 0.25
B. 0, 0.25
C. 0.25, 0.25
D. 0, 0.0
1. What do you call the tasks that a program can run concurrently?

A. Programs

B. Threads

C. Events

D. Jobs
2. How does the JVM choose which thread to run first?

A. It picks the one with highest priority.

B. It picks the one with lowest priority.

C. It randomly chooses threads.

D. It chooses the one that will complete in the shortest amount of time.
3. Which interface do you have to implement in order to define a task class?

A. Thread
B. Executable
C. Runnable
D. Task

(C) Runnable
public void mouseClicked( ...) {
    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 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) {
                ________________________________;
                repaint();
            }
            // We will register this listener
            // with the MyPanel objects
        }
    }
}
// The start button
JButton jbtStart = new JButton( "Start" );
ActionListener ssListen = new StartStopListener();
jbtStart.addActionListener( ssListen );

// The stop button
JButton jbtStop = new JButton( "Stop" );
jbtStop.addActionListener( ssListen );
class StartStopListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        JButton src = (JButton)e.getSource();<br>
        if (src.getText().equals( "Start" )) {
            int panelNum = 0;
            for ( int i = 0; i < 20; i++ ) {
                if ( board[panelNum] == ' ' ) {
                    board[panelNum] = turn;
                } else {
                    board[panelNum] = ' ';
                }
            }
            panelNum = (panelNum + 1) % 3;
            repaint();
        }
        else if (src.getText().equals( "Stop" ))
            System.out.println( "STOP!!" );
    }
}
class StartStopListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        JButton src = (JButton)e.getSource();
        if (src.getText().equals( "Start" )) {
            int panelNum = 0;
            for ( int i = 0; i < 20; i++ ) {
                if ( board[panelNum] == ' ' ) {
                    board[panelNum] = turn;
                } else {
                    board[panelNum] = ' ';  
                }
                panelNum = (panelNum + 1) % 3;
                repaint();
            }
        } else if (src.getText().equals( "Stop" ))
            System.out.println( "STOP!!" );
    }
}
class StartStopListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {

        JButton src = (JButton)e.getSource();
        if (src.getText().equals( "Start" )) {
            int panelNum = 0;
            for ( int i = 0; i < 20; i++ ) {
                if ( board[panelNum] == ' ' ) {
                    board[panelNum] = turn;
                } else {
                    board[panelNum] = ' ';
                }
                panelNum = (panelNum + 1) % 3;
            }
            repaint();
        } else if (src.getText().equals( "Stop" ))
            System.out.println( "STOP!!" );
    }
}

repaint() simply schedules a call to refresh the component, *when the main GUI thread gets around to it*. Because the GUI thread is busy executing your code, nothing happens until your code finishes.
class StartStopListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        JButton src = (JButton)e.getSource();
        if (src.getText().equals("Start")) {
            int panelNum = 0;
            for (int i = 0; i < 20; i++) {
                if (board[panelNum] == ' ')
                    board[panelNum] = turn;
                else
                    board[panelNum] = ' ';
                panelNum = (panelNum + 1) % 3;
            }
            repaint();
            try {
                Thread.sleep(500);
            }
            catch (InterruptedException e) {} 
        } else if (src.getText().equals("Stop"))
            System.out.println("STOP!!");
    }
}
public class ConnectFourAnimationDemo extends JFrame implements Runnable {
    private boolean running; // Should the animation run?
    public ConnectFourAnimationDemo() {  
        ...
        new Thread( this ).start();
    }

    public void run() {
        try {
            int panelNum = 0;
            while( true ) {
                if ( running ) {
                    if ( board[panelNum] == ' ' )
                        board[panelNum] = turn;
                    else
                        board[panelNum] = ' ';
                    panelNum = (panelNum + 1) % 3;
                    repaint();
                }
                Thread.sleep( 500 );
            }
        } catch (InterruptedException ex) { }
    }
}

Can be run in a separate thread
A class variable sets the “state” of the animation
Launch the run method in a separate thread.

repaint asks the main GUI thread to repaint.
But sleep happens in a separate thread
so is doesn’t affect the main thread (in fact it allows it to run again!)
This is not the best way to do animation in Swing. You should use Swing Timers instead (see Section 16.12 in the book). I am doing it this way to illustrate Threads.
Fill in the missing to correctly control the animation.

class StartStopListener implements ActionListener
{
    public void actionPerformed( ActionEvent e )
    {
        JButton src = (JButton)e.getSource();
        if (src.getText().equals( "Start" )) {
            running = true;
        }
        else if (src.getText().equals( "Stop" )) {
            running = false;
        }
    }
}