CSE 8B Today

Don’t forget: Fill out your CAPES, TA evaluations, and your survey

PSA8 due WEDNESDAY week 10 (+1 slip day if you have any left)

PSA8 + Threads + Exceptions

Simulations? For what?

Cool-Awesome Scientific Computer Simulations
1. Which one of the following statements about Multi-threading is correct?

A. Multithreading enables multiple programs to run the same methods.

B. Multithreading enables multiple tasks in a program to be executed concurrently.

C. Multithreading means multiple methods can run in the main method.

D. Multithreading means that we can achieve a sequential execution of all our methods.

B. Multithreading enables multiple tasks in a program to be executed concurrently.
2. Tasks in Java are objects. They actually need to be an object of a class that implements a particular interface. Which interface?

A. Runnable

B. JTask

C. TaskListener

D. Thread
3. Consider the following declaration of a variable of the class PrintChar (this class is exactly as defined in the book, so it consists of a task class that prints several times a given char):
PrintChar printA = new PrintChar('a', 100);

Which line or lines of code do you need to start a Thread that runs that task?

A. printA.run();

B. Thread t1 = new Thread(printA);
t1.start();

C. Thread t2 = new Thread();
t2.run(printA);

D. Task t3 = new Task(printA);
4. The Timer class can be used to control animations. What does the Timer class fire at a predefined rate?

A. An Exception
B. A System.out.println message
C. An ActionEvent
D. A new Thread
CSE 8B Today

PSA 8: WHAT NEEDS TO BE DONE?
Listeners (Buttons and Panel)
Display Critters (paint)
Critters “behaviors” (Interactor and reactTo)

PSA8 due WEDNESDAY week 10 (+1 slip day if you have any left)
CSE 8B Today

Trying to make an animation...
(Steps in this class example are very similar to PSA8 steps... SOME are already given in the starter files)

Threads vs. SwingTimers
// 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 );

THIS IS NOT A CLASS!!!
But we are allowed to use the "interface" name as the type.
It means "any" class that Implements that interface can be assigned to that variable.

1. ADD START/STOP animation button listener
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!!");
        }
    }
}

2. Actually implement the START/STOP button listener class

What will happen when the user clicks the Start button?
C. The GUI will show only the last frame of 20 frames of animation, but it will take a lot longer!!

This Puts the MAIN GUI thread to sleep 500 ms …

WE NEED “ANOTHER” THREAD … ➔ DEMO!
public class ConnectFourAnimDemo 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) { }
  }
}

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 as it happened before ...
public class CritterController extends JApplet implements Runnable

// Instance variables that maintain various parts of the simulation
private ArrayList<Critter> critterList;  /** The Critters in the world */
private Interactor actor; /** … object that controls how Critters interact */
private boolean running;  /** Whether or not the simulation is running */
private CritterPanel world; /** The custom JPanel that displays the Critters */
private JLabel stateLabel;  /** The label at the top of the panel */
private JLabel critLabel; /** The label next to the Critter buttons */

public init() { // ALL DONE, including how to start the thread...
    // rest of the init code . . .
    new Thread( this ).start();
}

public void run(){ // ALL DONE! But method
    try {
        while( true ) {
            if ( running ) {
                Rectangle r = new Rectangle( . . .);
                for ( int i = 0; i<critterList.size(); i++) {
                    Critter crit = (Critter) critterList.get(i); // PICK A CRITTER!
                    actor.interact( crit, critterList, r ); // MAKE it INTERACT with others
                }
                repaint(); // UPDATE how things look like after ALL THE INTERACTIONS!
            }
        }
    }
    catch (InterruptedException ex) { //want to do something if this crashes?   }
}

DON’T PANIC! Many of these steps are given in PSA8 starter files …
DON’T PANIC! Many of these steps are given in PSA8 starter files...

The code you need to use about Threads and Exceptions is given in the starter files! → NO NEED TO WAIT TO START PSA8! (in a minute... you’ll know already all you need to finish it!)

TODAY we’re learning to understand what the thread-related Code means

TUESDAY a few more PSA8 final tips and understand better what the Exception-handling related code means
So... Run() is “doing” the board variable updates IF simulation is running; and the main is handling the repaint() request. DO I need to put something in the button listener then?

```java
class StartStopListener implements ActionListener
{
    public void actionPerformed( ActionEvent e )
    {
        JButton src = (JButton)e.getSource();
        if (src.getText().equals("Start")) {
            // _______________ ?? ________________
        } else if (src.getText().equals("Stop")) {
            // _______________ ?? ________________
        }
    }
}
```

*Fill in the missing to correctly control the animation.*
2. Actually implement the START/STOP button listener class

So... Run() is “doing” the board variable updates IF simulation is running; and the main is handling the repaint() request. DO I need to put something in the button listener then?

```java
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;
        }
    }
}
```

Fill in the missing to correctly control the animation.
ConnectFourAnimationDemo

running false

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) {
    }
}

Main GUI thread

Animation thread

new Thread(this).start();

Start button clicked, running set to true

Wake up and do loop

Change contents of board

Animation thread sleeps between checks. Main GUI thread waits (and yields) when not active.
Trying to WRAP up . . . our ConnectFourAnimationDemo object IS our task object

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

In the ConnectFourAnimationDemo constructor, what line of code will execute the ConnectFourAnimationDemo’s run method in a new Thread?

A. Thread.start(this);

B. new Thread(this).start();

C. this.run();

D. ConnectFourAnimationDemo.run();
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

Animation runs continuously in a separate thread

The animation is controlled by an infinite loop that wakes up and checks to see if something needs to be done.
PanelAnimator is an inner class of ConnectFourAnimationDemo class PanelAnimator implements Runnable {
    int panelNum;
    int delay;
    public PanelAnimator( int num, delayIn ) {
        panelNum = num;
        delay = delayIn;
    }
    public void run() {
        try {
            while( true ) {
                if ( running ) {
                    if ( board[panelNum] == ' ' )
                        board[panelNum] = turn;
                    else
                        board[panelNum] = ' ';
                    repaint();
                } else
                    board[panelNum] = ' ';
                repaint();
            }
            Thread.sleep( delay );
        }
        catch (InterruptedException ex) { } }
    }
}

// In the ConnectFourAnimationDemoConstructor...
PanelAnimator p0 = new PanelAnimator( 0, 500 );

// What line of code will start this PanelAnimator running in a new Thread?
} // End of C4AD constructor

// PanelAnimator is an inner class of ConnectFourAnimationDemo
class PanelAnimator implements Runnable {
    In this case, what line of code will start this Panel animation in a new Thread?
}

WHAT IF I WANT task object a separate class????

A. p0.run();
B. p0.start();
C. Thread.run( p0 );
D. new Thread(p0).start();
E. new Thread(p0).run();
PanelAnimator is an inner class of ConnectFourAnimationDemo class PanelAnimator implements Runnable {
    int panelNum;
    int delay;
    public PanelAnimator( int num, delayIn ) {
        panelNum = num;
        delay = delayIn;
    }
    public void run() {
        try {
            while( true ) {
                if ( running ) {
                    if ( board[panelNum] == ' ' )
                        board[panelNum] = turn;
                    else
                        board[panelNum] = ' ';
                    repaint();
                }
                Thread.sleep( delay );
            }
        }
        catch (InterruptedException ex) { }
    }
}

// In the ConnectFourAnimationDemoConstructor...
PanelAnimator p0 = new PanelAnimator( 0, 500 );
new Thread(p0).start();
} // End of C4AD constructor

WHAT IF I WANT task object a separate class????

What will the GUI do when the user clicks “Start”?
A. All three circles will change colors automatically, like they did before.
B. Only the first circle will change colors automatically
C. Nothing will happen
class PanelAnimator implements Runnable {
    int panelNum;
    int delay;
    public PanelAnimator( int num, delayIn ) {
        panelNum = num;
        delay = delayIn;
    }
    public void run() {
        try {
            while( true ) {
                if ( running ) {
                    if ( board[panelNum] == ' ' )
                        board[panelNum] = turn;
                    else
                        board[panelNum] = ' ';
                    repaint();
                }
                Thread.sleep( delay );
            }
        } catch (InterruptedException ex) { }
    }
}

// In the ConnectFourAnimationDemoConstructor...
PanelAnimator p0 = new PanelAnimator( 0, 500 );
new Thread(p0).start();

class PanelAnimator implements Runnable {
    int panelNum;
    int delay;
    public PanelAnimator( int num, delayIn ) {
        panelNum = num;
        delay = delayIn;
    }
    public void run() {
        try {
            while( true ) {
                if ( running ) {
                    if ( board[panelNum] == ' ' )
                        board[panelNum] = turn;
                    else
                        board[panelNum] = ' ';
                    repaint();
                }
                Thread.sleep( delay );
            }
        } catch (InterruptedException ex) { }
    }
}

DO IT AT HOME: Add code to make the three circles change color at different speeds
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) { }
    }

    catch (InterruptedException ex) { } }
}
public class AnimationDemoSwingTimer extends JFrame implements Runnable{
    private boolean running; // Should the animation run?
    private Timer animationTimer; // Timer to control the animation

    public ConnectFourAnimationDemo(){
        public AnimationDemoSwingTimer() {

            new Thread( this ).start();
            this.animationTimer = new Timer( 500, new AnimationListener());
            this.animationTimer.stop();
        }

        public void run() {
            class AnimationListener implements ActionListener {
                public void actionPerformed( ActionEvent e ) {
                    if (board[panelNum] == ' ' ) {
                        board[panelNum] = turn;
                    }
                    else {
                        board[panelNum] = ' ';
                    }
                    panelNum = (panelNum + 1) % 3;
                    repaint();
                }
            }
    

    Where did the loop go?
    TIMER FIRES ACTION_EVENTS! Every 500 ms
public class AnimationDemoSwingTimer extends JFrame implements Runnable {
    private boolean running;  // Should the animation run?
    private Timer animationTimer;  // Timer to control the animation

    class StartStopListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            JButton src = (JButton)e.getSource();
            if (src.getText().equals( "Start" )) {
                running = true;
                __________________(1)___________________
            }
            else if (src.getText().equals( "Stop" )) {
                running = false;
                __________________(2)___________________
            }
        }
    }

    private void this.run() {
        __________________(3)___________________
    }

    Recall that the StartStopListener is attached to the Start and Stop buttons. What should go in blank 1?
    A. while (true ){
    B. animationTimer.start()
    C. animationTimer.stop()
    D. this.run()
    E. Something else
public class AnimationDemoSwingTimer extends JFrame implements Runnable {
    private boolean running; // Should the animation run?
    private Timer animationTimer; // Timer to control the animation

class StartStopListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        JButton src = (JButton)e.getSource();
        if (src.getText().equals("Start")) {
            animationTimer.start();
        } else if (src.getText().equals("Stop")) {
            animationTimer.stop();
        }
    }
}
class AnimationListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        if (board[panelNum] == ' ') {
            board[panelNum] = turn;
        } else {
            board[panelNum] = ' ';
        }
        panelNum = (panelNum + 1) % 3;
        repaint();
    }
}
CSE 8B Today

Trying to make an animation...
(steps in this class example are very similar to PSA8 steps... SOME are already given in the starter files)

Threads vs. SwingTimers

Main GUI thread
Animation thread

new Thread(this).start();
run()
repaint()
repaint()
repaint()

Wake up and do loop
Change contents of board

Main GUI thread

Timer actionEvents fired
NEXT: What is an IOError?

... And what can we do with them
What is an IOException?

// In class StringPlay
public String readIt( String filename ) throws IOException
{
    File sourceFile = new File( filename );
    Scanner input = new Scanner( sourceFile );
    String allText = "";
    while ( input.hasNextLine() )
    {
        String s1 = input.nextLine();
        allText = allText.concat( s1 );
    }
    System.out.println( allText );
    return allText;
}

Essentially, an exception is just an error that can be passed around (thrown)
If sourceFile does not exist, the Scanner’s constructor will throw an exception
```
// In class StringPlay
public String readIt( String filename )
{
    File sourceFile = new File( filename );
    Scanner input = null;
    try {
        input = new Scanner( sourceFile );
    } catch ( IOException e ) {
        System.out.println( e.getMessage() );
        return "";
    }
    String allText = "";
    while ( input.hasNextLine() )
    {
        String s1 = input.nextLine();
        allText = allText.concat( s1 );
    }
    System.out.println( allText );
    return allText;
}
```
Handling exceptions

// In class StringPlay
public String readIt( String filename )
{
    File sourceFile = new File( filename );
    Scanner input = null;
    try {
        input = new Scanner( sourceFile );
        }
    catch ( IOException e ) {
        System.out.println( e.getMessage() );
        return ""
    }
    String allText = "";
    while ( input.hasNextLine() )
    {
        String s1 = input.nextLine();
        allText = allText.concat( s1 );
    }
    System.out.println( allText );
    return allText;
}