Object-Oriented Programming

Session: Week 5 Session 1
Instructor: Eric Pogue

Agenda:

1. Review this week's programming Assignment
2. Introduce the week's Learning Objectives
3. Topics & Examples

FastPrime… plus Questions

Write a performance optimized command line Java application that will programmatically find prime numbers [link] and store those numbers sorted in an output file.

In FastPrime we will create a command line Java application that will:

1. Use multiple threads to find the prime numbers between two numbers
2. Sort those results and store them to a file
3. Perform some timings
4. … And do this all very fast

See the details in this week's assignment

Application performance is ALWAYS a challenge. Learn how to optimize, test, and enhance the performance regularly. It can't be built in at the end of a project!

Learning Objectives – Week 5

1. Understand how Java uses **files** for input and output (IO)
2. Design and implement a controller class to **serialize** (reads & writes) data to a text file
3. Understand **threads** and how to develop and optimize multi-treaded Java applications
4. Understand Java **packages** and compile a class so that it belongs to a particular package
5. Import a class you write that belongs to a particular package.
6. Identify reasons for using JAR (Java ARchive) files to group together related java classes
7. Create a JAR file that stores the contents of a particular package
8. Explain software **testing** terms including unit, integration, user acceptance, performance testing, manual, automated, verification, validation, etc.
9. Understand the importance of testing and the criticality of finding/fixing defects early
10. Explain the purpose, syntax, and annotations of the various assert statements **JUnit** supports
11. Install JUnit onto your machine and execute a JUnit test on your application

Serialization and Writing/Reading Text Files (IO)

Serialization is an object-oriented programming term that means converting an object to a byte steam usually to be written to or read from a text or binary file.

To write to a text file:

- Create a File object, feeding the file's path to the File class constructor
- Create a FileWriter to access the File
- Create a BufferedWriter to write data to the FileWriter efficiently
- Use BufferedWriter's write and newLine functions to commit the data to the file.

To read from a text file:

- Create a File object, feeding the file's path to the File class constructor
- Attach a Scanner object to it
- Use Scanner's readLine and hasNextLine functions to read the file

A buffered writer or buffered stream efficiently organization reads and writes for optimal disk performance. The danger is that if there is a power failure (or someone accidentally kicks your power strip) you could lose data. It is good to "flush the buffer" at critical times when using a buffered writer on mission critical projects. There is little or no danger in using a buffered reader.

Binary files versus text files.

XML – Text file format for structured data… HTML for data.
JSON – "Simplified" text file format for structured data.
Binary

XML Example

In computing, XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is an open standard that:

- Supports nearly all development languages and platforms
- Allows us to cross between many applications
- Can result in large files
- Supports schema to validate data

```xml
<?xml version="1.0" encoding="UTF-8"?>

<shiporder orderid="889923"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>
  </item>
</shiporder>
```

JSON Example

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is an open standard that:

- Supports nearly all development languages and platforms
- Allows us to cross between many applications
- Can result in large files

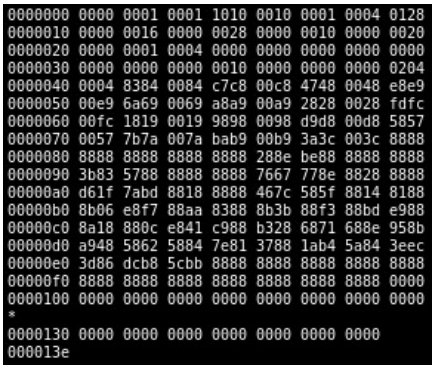
```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

JSON is an open-standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is a very common data format used for asynchronous browser/server communication, including as a replacement for XML in some web service style systems.

Binary Files

A binary file is a computer file that is not a text file. The term "binary file" is often used as a term meaning "non-text file". The can be open or closed formats that are generally:

- Fast, small, and efficient*
- Often not very portable across applications and platforms
- Difficult to maintain backward compatibility



Some people would say that binary files include all files, and that text files are just binary files that are being interpreted in a specific way.

Serialization Example

We are going to demonstrate Writing and Reading OvalDraw data to a proprietary text file by:

- Starting with the OvalDraw application
- Hooking up the “Open” and “Save” menu events
- Teaching our OvalDraw class to Write and Read itself
- Saving our session
- Loading our session

A buffered writer or buffered stream efficiently organization reads and writes for optimal disk performance. The danger is that if there is a power failure (or someone accidentally kicks your power strip) you could lose data. It is good to “flush the buffer” at critical times when using a buffered writer on mission critical projects. There is little or no danger in using a buffered reader.

Binary files versus text files.

XML – Text file format for structured data… HTML for data.
JSON – “Simplified” text file format for structured data.
Binary

End of Session

Course Number: CPSC-24500

Week: 5

Session: 1

Instructor: Eric Pogue

Threads & Multithreaded Applications

Multithreading: A technique by which a single set of code can be used by several processors or cores at different stages of execution.

Mul

I am using the multithreading and multiprocessing terms synonymously for our purposes.

Testing

We test to find defects and/or to validate that we have not introduced new defects.
Defects are exponentially more expensive to fix the longer the exist.
Performance issues are often the most difficult and expensive defects to fix. They are often not found until the application if running under production load… which is often only when it is in production.

Unit - $200
Integration - $600
User Acceptance - $6,000
Production - $100,000+

The permutations of modern software features, data, tools, environments, etc. quickly becomes unmanageable. Testability needs to be goal of nearly all non-trivial applications.

Integration (or Functional) Test: Another kind of test. These test whether the functionality of the entire application works as it should, or that the entire application behaves as expected.

Performance Test: Still another kind of test. These benchmark the performance of an application, especially under high load.

Behavior vs. State Testing: Behavior testing focuses on verifying that the correct functions were called with the correct parameters. In other words, are the various pieces interacting correctly? State testing focuses on the results of those calls. We will focus on state testing.

Manual Testing	Automated Testing
Executing a test cases manually without any tool support is known as manual testing.	Taking tool support and executing the test cases by using an automation tool is known as automation testing.
Time-consuming and tedious − Since test cases are executed by human resources, it is very slow and tedious.	**Fast** − Automation runs test cases significantly faster than human resources.
Huge investment in human resources − As test cases need to be executed manually, more testers are required in manual testing.	**Less investment in human resources** − Test cases are executed using automation tools, so less number of testers are required in automation testing.
Less reliable − Manual testing is less reliable, as it has to account for human errors.	**More reliable** − Automation tests are precise and reliable.
Non-programmable − No programming can be done to write sophisticated tests to fetch hidden information.	**Programmable** − Testers can program sophisticated tests to bring out hidden information.

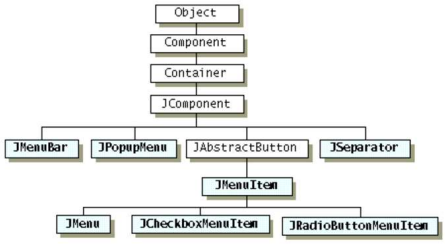
False positives versus valid defects found.

Manual Testing and Automated Testing can be supportive of each other.

Menu Bars, Menus, and Menu Items

Java Menus are implemented with three related classes:

- **JMenuBar**: the horizontal component across the top of the frame that contains Menus
- **JMenu**: the user interface element that implements "File" in the image below
- **JMenuItem**: the sub-items that can be selected from a Menu like "New" and "Exit" in the example below
- The inheritance hierarchy for menu-related classes:



Overview:

You create the JMenuBar. You create JMenu items that have captions (like File, Edit, etc.). You create JMenuItem objects that have captions (like Open, Close, Save, etc.) and add them to the JMenu's. You associate ActionListener objects with each JMenuItem to describe what should happen when the JMenuItem is clicked. You add each JMenuItem to the JMenu. You add the JMenu to the JMenuBar. You then tell the frame to set its JMenuBar using setJMenuBar.

Mouse Clicking, Mouse Dragging, and Keystrokes

Nearly all graphical applications respond to Mouse Clicking and Keystrokes. Many applications also implement special behavior for Mouse Dragging. With Java in order to respond to these events we implement ActionListeners including:

- **MouseListener:** Interface to implement to respond to Mouse Clicking
- **MouseMotionListener**: Interface to implement to respond to Mouse Dragging
- **KeyListener:** Interface to implement to respond to Keystrokes

MouseListener Interface

The MouseListener interface requires five methods to be implemented:

- mouseClicked(MouseEvent e): Invoked when the mouse button has been clicked on a component
- mouseEntered(MouseEvent e): Invoked when the mouse enters a component
- mouseExited(MouseEvent e): Invoked when the mouse exits a component
- mousePressed(MouseEvent e): Invoked when a mouse button has been pressed on a component
- mouseReleased(MouseEvent e): Invoked when a mouse button has been released on a component

Adapter Classes are an alternative to implementing all five methods. For our examples, I will be implementing all five, even if some of them seem unnecessary. You ware welcome to use Adapters in your assignments if you prefer.

MouseMotionListener Interface

The MouseMotionListener interface includes the following methods:

- mouseDragged(MouseEvent e): Invoked when a mouse button is pressed on a component and then dragged
- mouseMoved(MouseEvent e): Invoked when the mouse cursor has been moved onto a component but no buttons have been pushed

.

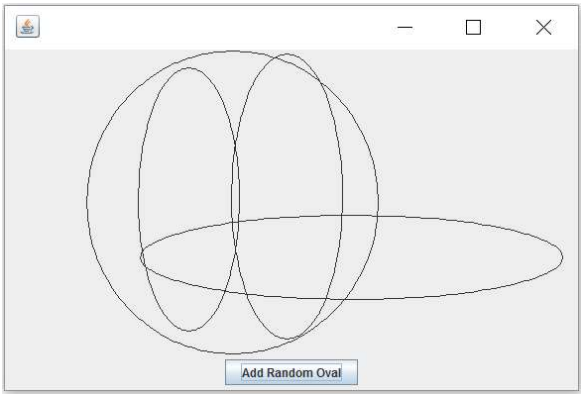
KeyListener Interface

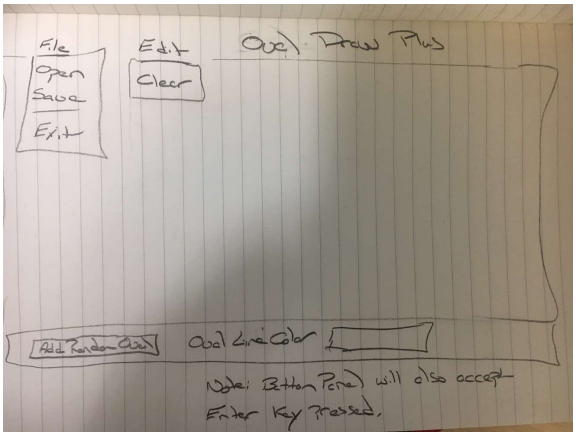
The KeyListener interface includes the following methods:

- keyPressed(KeyEvent e): Invoked when a key has been pressed
- keyReleased(KeyEvent e): Invoked when a key has been released
- keyTyped(KeyEvent e): Invoked when a key has been typed

.

OvalDraw Plus User Interface Design





Yes, this is often what a UI design looks like. In our first coding example this week, we are going to enhance the OvalDraw application to include menu elements to OvalDraw File|Exit and Edit|Clear. In addition, we will implement adding new random ovals with either a line color or a line weight. Finally, we will add file saving and opening and add the related Open and Save menu items to the application… more on that later.

I anticipate that we will also implement something with mouse clicking, mouse dragging, and timer also, but am uncertain what form that may take.

Timers

Timers are pretty straight forward to implement:

- Implement the ActionListener Interface
- Create a new timer passing in the object that implemented the ActionListener

```
//*************************************************************
// Timer

import java.util.Timer

class DrawPanel extends JPanel implements ActionListener {
    private Timer myTimer;

    public DrawPanel {
        myTimer = new Timer(1000,this);
    }

    public void actionPerformed(ActionEvent e) {
        // Do timer stuff.
    }
}
```

We will likely have a final programming example that will show some animation using timers. I'm not sure what form that example will take.

Model-View-Controller

Throughout our design and development activates we are working to create applications that segregate our View from our their Model & Data.

- The Java environment makes this more challenging than it should be at times
- Design tradeoffs need to be considered
- The most import aspect of Model-View-Controller is that the Model has no visibility to the View

Learning Objectives – Week 4

1. Implement a menu system that enables the user to trigger a variety of actions in a familiar way
2. Implement code that responds to of events including: clicking the mouse, moving or dragging the mouse, and typing a key on the keyboard
3. Design multiple intuitive ways for a user to perform a particular task
4. Implement animation using a timer and a corresponding event handler
5. Create interactive applications that adhere to the Model-View-Controller pattern
6. Understand how Java achieves speedier input and output (IO) through a hierarchy file IO classes
7. Design and implement a controller class that outputs data to a text file
8. Design and implement a controller class that inputs data from a text file and builds a collection of objects from the data read data

Any of these items would be a good candidate for your “interesting and unique” feature to add to your Mosaic assignment.

Serialization and Reading/Writing Text Files

Serialization is an object-oriented programming term that means converting an object to a byte steam usually to be written to or read from a file.

To write to a text file:

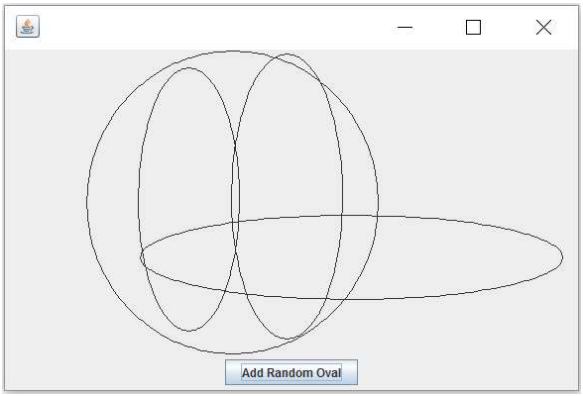
- Create a File object, feeding the file's path to the File class constructor
- Create a FileWriter to access the File
- Create a BufferedWriter to write data to the FileWriter efficiently
- Use BufferedWriter's write and newLine functions to commit the data to the file.

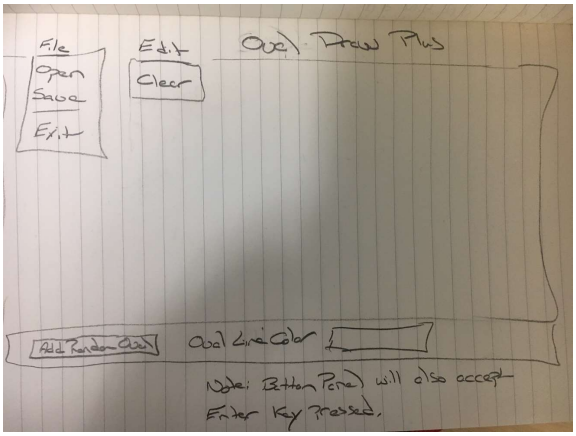
To read from a text file:

- Create a File object, feeding the file's path to the File class constructor
- Attach a Scanner object to it
- Use Scanner's readLine and hasNextLine functions to read the file

A buffered writer or buffered stream efficiently organization reads and writes for optimal disk performance. The danger is that if there is a power failure (or someone accidentally kicks your power strip) you could lose data. It is good to "flush the buffer" at critical times when using a buffered writer on mission critical projects. There is little or no danger in using a buffered reader.

OvalDraw Plus User Interface Design





Our second programming example this week will be implementing Save & Open in our OvalDraw application.

Closing Comments & Next Steps

- As promised, less theory and more development
- Look for three follow-up development examples this week:
  - OvalDraw with User Experience Enhancements
  - OvalDraw reading and writing files
  - Something with Timers and Animation

End of Session

Course Number: CPSC-24500

Week: 4

Session: 1

Instructor: Eric Pogue

Object-Oriented Programming

Session: Week 4 Session 2
Subject: More Interactive User Interfaces
Instructor: Eric Pogue

LEWIS UNIVERSITY
A Catholic and Lasallian University

Agenda:

1. Quick Review Learning Objectives we covered in Session 1
2. Overview of Session 2 Live Coding example and related Learning Objectives
3. Live User Interface Coding example
4. Review what we learned
5. Closing comments and next steps

Quick note: I have changed the resolution of our videos starting with this session. I was using 1366x768 which made for more manageable video downloads, etc. As we head toward using Visual Studio and C#, I would like to move toward 1920x1080 to better allow us to see the integrated development environment (IDE). Let me know if the increased resolution causes difficulty in accessing or viewing the videos.

Learning Objectives – Week 4

1. Implement a menu system that enables the user to trigger a variety of actions in a familiar way
2. Implement code that responds to of events including: clicking the mouse, moving or dragging the mouse, and typing a key on the keyboard
3. Design multiple intuitive ways for a user to perform a particular task
4. Implement animation using a timer and a corresponding event handler
5. Create interactive applications that adhere to the Model-View-Controller pattern
6. Understand how Java achieves speedier input and output (IO) through a hierarchy file IO classes
7. Design and implement a controller class that outputs data to a text file
8. Design and implement a controller class that inputs data from a text file and builds a collection of objects from the data read data

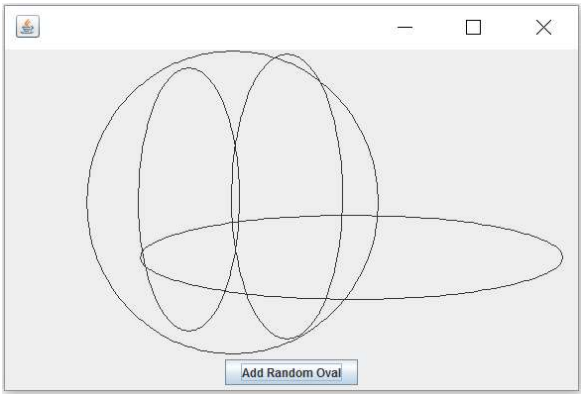
Any of these items would be a good candidate for your “interesting and unique” feature to add to your Mosaic assignment.

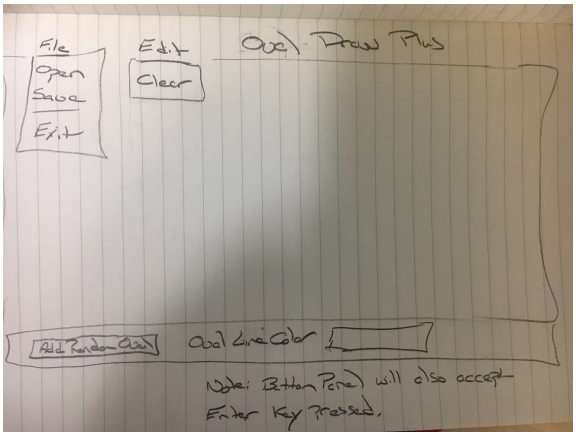
Learning Objectives – Week 4

1. Implement a menu system that enables the user to trigger a variety of actions in a familiar way
2. Implement code that responds to of events including: clicking the mouse, moving or dragging the mouse, and typing a key on the keyboard
3. Design multiple intuitive ways for a user to perform a particular task
4. Implement animation using a timer and a corresponding event handler
5. Create interactive applications that adhere to the Model-View-Controller pattern
6. Understand how Java achieves speedier input and output (IO) through a hierarchy file IO classes
7. Design and implement a controller class that outputs data to a text file
8. Design and implement a controller class that inputs data from a text file and builds a collection of objects from the data read data

Any of these items would be a good candidate for your “interesting and unique” feature to add to your Mosaic assignment.

OvalDraw Plus User Interface Design





Yes, this is often what a UI design looks like. In our first coding example this week, we are going to enhance the OvalDraw application to include menu elements to OvalDraw File|Exit and Edit|Clear. In addition, we will implement adding new random ovals with either a line color or a line weight. Finally, we will add file saving and opening and add the related Open and Save menu items to the application… more on that later.

I anticipate that we will also implement something with mouse clicking, mouse dragging, and timer also, but am uncertain what form that may take.

Learning Objectives – Week 4

1. Implement a menu system that enables the user to trigger a variety of actions in a familiar way
2. Implement code that responds to of events including: clicking the mouse, moving or dragging the mouse, and typing a key on the keyboard
3. Design multiple intuitive ways for a user to perform a particular task
4. Implement animation using a timer and a corresponding event handler
5. Create interactive applications that adhere to the Model-View-Controller pattern
6. Understand how Java achieves speedier input and output (IO) through a hierarchy file IO classes
7. Design and implement a controller class that outputs data to a text file
8. Design and implement a controller class that inputs data from a text file and builds a collection of objects from the data read data

Any of these items would be a good candidate for your “interesting and unique” feature to add to your Mosaic assignment.

Closing Comments & Next Steps

- I'm trying to get ups in a learning pattern where we:
  1. See the concepts and patterns (theory)
  2. Watch them being used in a "real-life" example
  3. Implement a solution using what we have learned
- Let me know how this is working, and how we can continue to improve

End of Session

Course Number: CPSC-24500

Week: 4

Session: 2

Instructor: Eric Pogue

Object-Oriented Programming

Session: Week 4 Session 3
Subject: Clicks, Drags, Timers, and Animations
Instructor: Eric Pogue

LEWIS UNIVERSITY
A Catholic and Lasallian University

Agenda:

1. Quick Review Learning Objectives we covered in Session 1
2. Overview of Session 3 Live Coding example and related Learning Objectives
3. Clicks and drags… with FaceDraw-ish solution
4. Short break
5. Timers and animation
6. Review what we learned and week 4 assignment as time allows
7. Final comment & questions

Start Recording!

Learning Objectives – Week 4

1. Implement a menu system that enables the user to trigger a variety of actions in a familiar way
2. Implement code that responds to of events including: clicking the mouse, moving or dragging the mouse, and typing a key on the keyboard
3. Design multiple intuitive ways for a user to perform a particular task
4. Implement animation using a timer and a corresponding event handler
5. Create interactive applications that adhere to the Model-View-Controller pattern
6. Understand how Java achieves speedier input and output (IO) through a hierarchy file IO classes
7. Design and implement a controller class that outputs data to a text file
8. Design and implement a controller class that inputs data from a text file and builds a collection of objects from the data read data

Any of these items would be a good candidate for your “interesting and unique” feature to add to your Mosaic assignment.

Learning Objectives – Week 4

1. Implement a menu system that enables the user to trigger a variety of actions in a familiar way
2. Implement code that responds to of events including: clicking the mouse, moving or dragging the mouse, and typing a key on the keyboard
3. Design multiple intuitive ways for a user to perform a particular task
4. Implement animation using a timer and a corresponding event handler
5. Create interactive applications that adhere to the Model-View-Controller pattern
6. Understand how Java achieves speedier input and output (IO) through a hierarchy file IO classes
7. Design and implement a controller class that outputs data to a text file
8. Design and implement a controller class that inputs data from a text file and builds a collection of objects from the data read data
9. Review week 4 assignment as time allows

Mouse Clicking, Mouse Dragging, and Keystrokes

Nearly all graphical applications respond to Mouse Clicking and Keystrokes. Many applications also implement special behavior for Mouse Dragging. With Java in order to respond to these events we implement ActionListeners including:

- **MouseListener:** Interface to implement to respond to Mouse Clicking
- **MouseMotionListener**: Interface to implement to respond to Mouse Dragging
- **KeyListener:** Interface to implement to respond to Keystrokes

MouseListener Interface

The MouseListener interface requires five methods to be implemented:

- mouseClicked(MouseEvent e): Invoked when the mouse button has been clicked on a component
- mouseEntered(MouseEvent e): Invoked when the mouse enters a component
- mouseExited(MouseEvent e): Invoked when the mouse exits a component
- mousePressed(MouseEvent e): Invoked when a mouse button has been pressed on a component
- mouseReleased(MouseEvent e): Invoked when a mouse button has been released on a component

Adapter Classes are an alternative to implementing all five methods. For our examples, I will be implementing all five, even if some of them seem unnecessary. You ware welcome to use Adapters in your assignments if you prefer.

The mouseClicked method is the most often overridden. The mousePressed and mouseReleased are most often used with dragging.

```
import java.awt.event.MouseListener;
import java.awt.event.MouseEvent;

public void mousePressed(MouseEvent e) {}
public void mouseReleased(MouseEvent e) {}
public void mouseEntered(MouseEvent e) {}
public void mouseExited(MouseEvent e) {}
public void mouseClicked(MouseEvent e) {}

addMouseListener(this);

MouseEvent:
int getX();
int getY();
```

MouseMotionListener Interface

The MouseMotionListener interface includes the following methods:

- mouseDragged(MouseEvent e): Invoked when a mouse button is pressed on a component and then dragged
- mouseMoved(MouseEvent e): Invoked when the mouse cursor has been moved onto a component but no buttons have been pushed

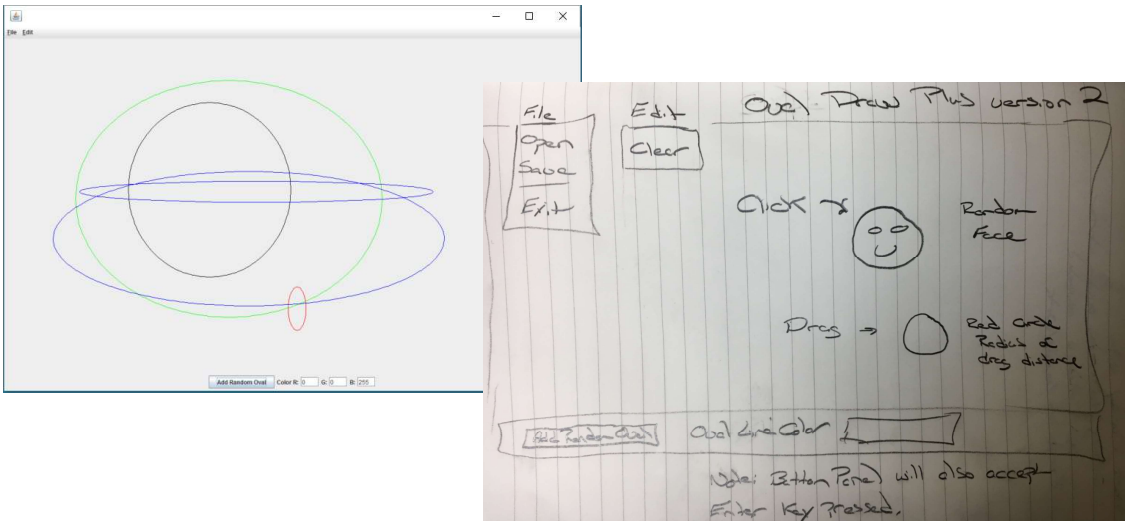
Note: Dragging is quite challenging to implement… and not that often utilized.

```
import java.awt.event.MouseMotionListener;
import java.awt.event.MouseEvent;

MouseMotionListener
public void mouseDragged(MouseEvent e) {}
public void mouseMoved(MouseEvent e) {}

addMouseMotionListener(this);
```

OvalDraw Plus With Clicking and Dragging



Enhance OvalDraw to include drawing a random face (similar to FaceDraw) with clicked and to draw a red circle with a radius of the drag distance when the mouse is dragged.

OvalDraw Plus Enhancements Steps

1. Clone our current Git repository
2. Copy original files over to OvalDrawPlus folder
3. Compile library and ShapeDraw code
4. Quickly test application
5. Review ShapeDraw code… focus on ShapeDrawPanel
6. Minor code cleanup… and “compress” classes that we will not be looking at today
7. Implement MouseListener Interface
8. Implement MouseMotionListener Interface
9. Add random oval on click
10. Implement red circle on mouse drag… this is challenging
11. Implement FaceDraw on click… and remove step nine’s random oval draw
12. Review, final question, and commit/push

Quick Break

Course Number: CPSC-24500

Week: 4

Session: 3

Instructor: Eric Pogue

Timers

Timers are pretty straight forward to implement:

- Implement the ActionListener Interface
- Create a new timer passing in the object that implemented the ActionListener

```
//*************************************************************
// Timer

import java.util.Timer

class DrawPanel extends JPanel implements ActionListener {
    private Timer myTimer;

    public DrawPanel {
        myTimer = new Timer(1000,this);
    }

    public void actionPerformed(ActionEvent e) {
        // Do timer stuff.
    }
}
```

We will likely have a final programming example that will show some animation using timers. I'm not sure what form that example will take.

```
import java.util.Timer;

private Timer animationTimer;

animationTimer = new Timer(1000,this);

public void actionPerformed(ActionEvent e) {
   System.out.println("actionPerformed(ActionEvent e)");
}
```

OvalDraw Plus Timer and Animated Faces

1. Implement ActionListener on ShapeDrawPanel
2. Implement Timer
3. Implent updateMouth in FaceDraw
4. Implement ArrayList for FaceDraw objects
5. Update FaceDraw mouths and redraw

Learning Objectives – Week 4

1. Implement a menu system that enables the user to trigger a variety of actions in a familiar way
2. Implement code that responds to of events including: clicking the mouse, moving or dragging the mouse, and typing a key on the keyboard
3. Design multiple intuitive ways for a user to perform a particular task
4. Implement animation using a timer and a corresponding event handler
5. Create interactive applications that adhere to the Model-View-Controller pattern
6. Understand how Java achieves speedier input and output (IO) through a hierarchy file IO classes
7. Design and implement a controller class that outputs data to a text file
8. Design and implement a controller class that inputs data from a text file and builds a collection of objects from the data read data
9. Review week 4 assignment as time allows

End of Session

Course Number: CPSC-24500

Week: 4

Session: 3

Instructor: Eric Pogue

Object-Oriented Programming

Session: Week 4 Session 4
Subject: Thursday Lunch Discussion & Lecture
Instructor: Eric Pogue

LEWIS UNIVERSITY
A Catholic and Lasallian University

Agenda – Thursday, April 13 from noon to 1pm:

1. Good natured chatting and banter ~10min
2. Quick introductions and how are you feeling about the course… start, stop, continue ~10min
3. Discussion, questions & answers
4. Week 4 Assignment Helpful Hints with MosaicLite

Start Recording!

Note: I am looking forward to the discussion mostly and will continue to complete the MosaicLite activities after 1pm as needed. Don't hesitate to drop off as your schedule requires. I will send out the link to the session after it is complete.

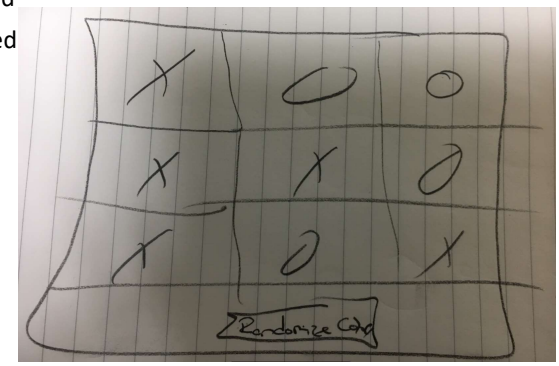
Introductions, Feedback, and Q&A

1. Name and one thing you are planning for the Easter weekend
2. Start, stop, continue:
   - What new thing should we start doing for the class?
   - What is one thing that we should stop doing?
   - What is one thing that we are doing that we should definitely keep doing?
3. Questions & Answers

This is your time… how can I help.

MosaicLite

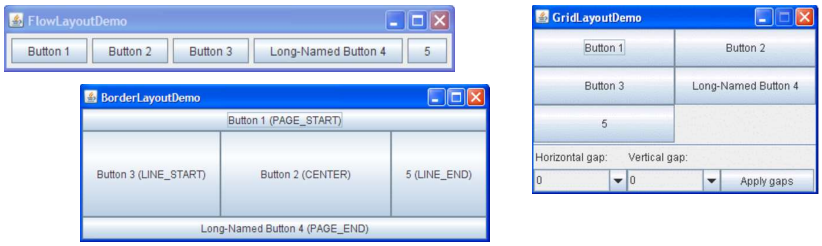
1. Implement a 3x3 grid of colored tiles
2. Implement tiles with a random background color
3. Paint an X or an O on the tile
4. Create a Randomize button
5. Randomize the colors when the button is pushed
6. Randomize the X or O when the button is pushed



Layout Managers (continued)

Layout Managers arrange controls on the screen so they are visually appealing. We will be focusing on three very common Layout Managers:

- FlowLayout: Arranges components left to right, top to bottom. When a FlowLayout runs out of room horizontally on a row, it places the next component as far left as it can go on the row below.
- BorderLayout: Arranges components in NORTH, SOUTH, EAST, WEST, and CENTER sections.
- GridLayout: Arranges components in an Excel-like table of rows and columns.



This is different from environments where developers (or designers) layout screens pixel by pixel (like Visual Basic for example). With modern UIs that need to fit in all types of sizes and orientations, layout managers are a necessity.

Several AWT and Swing classes provide layout managers for general use:

- BorderLayout
- BoxLayout
- CardLayout
- FlowLayout
- GridBagLayout
- GridLayout
- GroupLayout
- SpringLayout

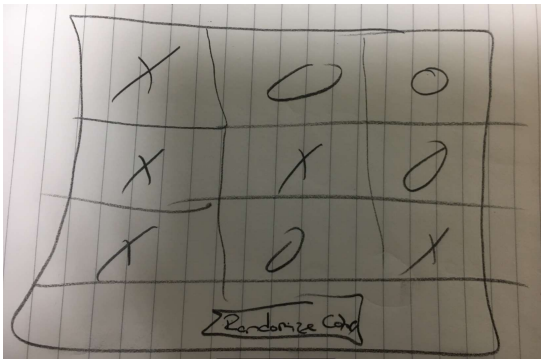
MosaicLite

1. JFrame
2. JPanel… grid & tile
3. JButton

4. GridLayout
5. BorderLayout
6. FlowLayout

7. ActionListener & ActionEvent

8. Random
9. Graphics
10. Color
11. Font



See application source code!

Closing Comments

- I hope the example helps…
- Give the holiday weekend, I am going to delay our file writing & reading activities until week 5
- Have a great weekend!

End of Session

Course Number: CPSC-24500

Week: 4

Session: 4

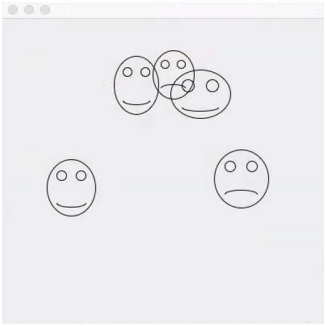
Instructor: Eric Pogue

Briefly Discuss FaceDraw Assignment

FaceDraw and Learning Objectives: FaceDraw will be a challenging assignment this week. Our Learning Objects, Discussion, Lecture, and Examples will all be focused on helping you complete the assignment and learn the key concepts through delivering a concrete example.

In FaceDraw we will:

1. Create an application in Java that draws random faces on a window
2. Generate a random number between 3 & 10 and draw that many faces
3. Randomly generate reasonable and visually appealing face
4. Position itself to a reasonable size and location
5. Draw all faces so they are entirely within the window.
6. Draw each face with two eyes and a mouth.
7. The mouth should be randomly smiling, frowning, or in-between



I'm looking for your feedback!

Briefly Discuss Mosaic Assignment

Mosaic and Learning Objectives: Mosaic will likely not be as challenging for you as FaceDraw was last week. Most of the elements that we will need for Mosaic we already learned last week. Therefore, this week's objectives will be more forward looking.

In Mosaic we will:

1. Draw a set of random tiles
2. Draw random shapes, letters, and colors
3. Respond to a "Randomize" button
4. Display a new set of random tiles when the button is pushed
5. Implement a interesting and unique feature of our own choosing

Less theory and more practical development…