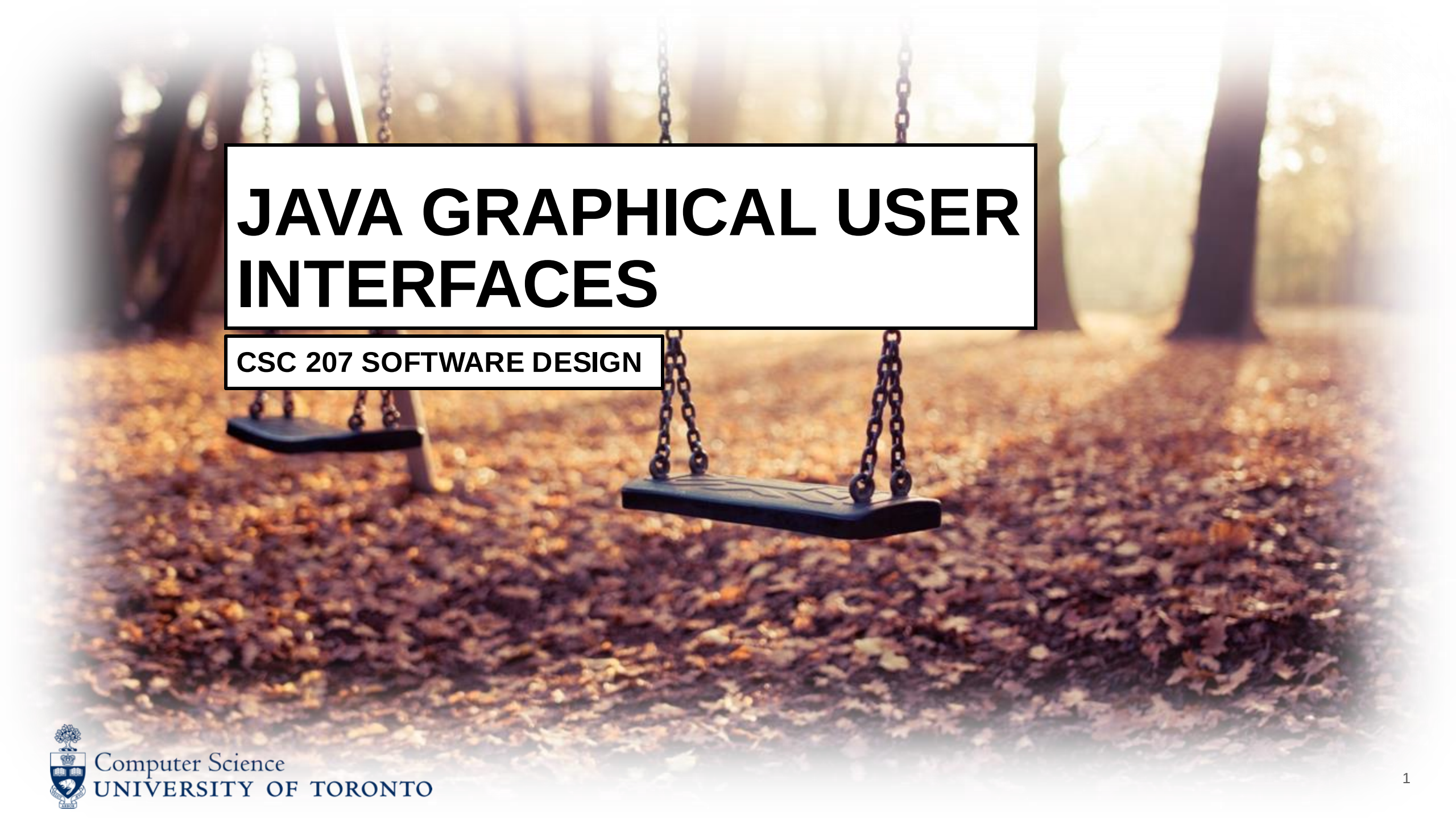


JAVA GRAPHICAL USER INTERFACES

CSC 207 SOFTWARE DESIGN

Computer Science
UNIVERSITY OF TORONTO

Course Overview

Tools (Weeks 1-4)

- Java
- Version Control
- Software Tools

Design (Weeks 5-8)

- Clean Architecture
- SOLID
- Design Patterns

Professional Topics (Weeks 9-12)

- Ethics
- Internships
- GenAI

- Last week we talked about interfaces, Generics, and the Java Collections Framework
- This week, we will discuss
  - testing with Junit
  - making GUIs with Java Swing

Questions to be answered this week...

Java Swing (GUIs)

- What are components in Swing?
- What are the differences between Flow Layout and a Box Layout?
- Describe a login screen and explain how you would create it by putting textboxes and buttons on a JPanel.
- What is a listener?

JUnit (testing)

- What is the goal of unit testing?
- What are the three steps to running tests in Junit?

LEARNING OUTCOMES

- Understand that the Java Swing framework has classes for building user interfaces: buttons, text fields, labels, and lots more, called *components*
- Be able to use nested panels and layout managers to organize a UI in a visually structured way
- Be able to implement an event listener to respond to a button click

Note: understanding how to make a *pretty* GUI is **not** a learning objective

JAVA SWING

- Swing is a Graphical User Interface (GUI) toolkit for creating user interfaces
- Most class names begin with “J”: JButton, JLabel, JTextField, and lots more
- Each graphical user interface class is called a *component*
- Event-driven: a method gets called automatically when an event happens

A SET OF EXAMPLES

- https://github.com/paulgries/JavaGUIExamples

A JAVA SWING EXAMPLE

```java
import javax.swing.*;

public class LabelAndButtonExample {
    public static void main(String[] args) {
        JLabel label = new JLabel("Hello, Swing!");
        JButton button = new JButton("Click Me");

        JPanel panel = new JPanel();
        panel.add(label);
        panel.add(button);

        JFrame frame = new JFrame("Simple Swing App");
        frame.setSize(300, 200);
        frame.setContentPane(panel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

JPANELS AND LAYOUT MANAGERS

- `JPanel` is a *container*: you can add other components to it, including other `JPanels`
- Each `JPanel` has a *layout manager* that determines how the components are arranged: Left to right? Vertical? In a grid?
- `FlowLayout` is the default — components flow left to right as you add them
- `BoxLayout` is like an advanced `FlowLayout`, supporting both horizontal and vertical flow
- Using nested `JPanels`, you can make some ugly but functional user interfaces
- There are other layout managers, some quite complicated https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

JPANEL USES FLOWLAYOUT BY DEFAULT

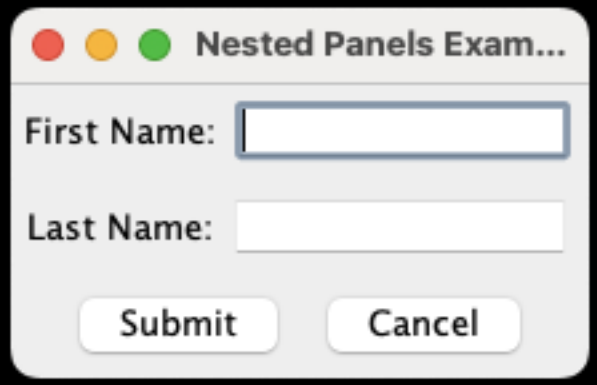
```
JPanel firstNamePanel = new JPanel();
firstNamePanel.add(new JLabel("First Name:"));
firstNamePanel.add(new JTextField(10));

JPanel lastNamePanel = new JPanel();
lastNamePanel.add(new JLabel("Last Name:"));
lastNamePanel.add(new JTextField(10));

JPanel buttonPanel = new JPanel();
buttonPanel.add(new JButton("Submit"));
buttonPanel.add(new JButton("Cancel"));

JPanel mainPanel = new JPanel();
mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.Y_AXIS));
mainPanel.add(firstNamePanel);
mainPanel.add(lastNamePanel);
mainPanel.add(buttonPanel);

JFrame frame = new JFrame("Nested Panels Example");
frame.setContentPane(mainPanel);
```



LISTENING FOR A BUTTON CLICK

- A *listener* is an object
- When you *instantiate* the listener, you *inject* it into a particular button object
- Each listener has a method, `actionPerformed`, that gets called when that button is clicked — this is magic that the JVM does for us!
- Inside each `actionPerformed` method, you may want to look at the values in text fields and other components, perhaps checkboxes and option buttons, so you will need to name them using variables in an *enclosing scope.*

LISTENING FOR A BUTTON CLICK

```
JButton submit = new JButton("Submit");
submit.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String firstName = firstNameField.getText();
        String lastName = lastNameField.getText();
        JOptionPane.showMessageDialog(null, "Hello " + firstName + " " + lastName);
    }
});
```

`ActionListener` is an interface

`new ActionListener` creates an *anonymous class*: we don't name it

Java lets us declare implementing classes for interfaces using this syntax

`firstNameField` and `lastNameField` must be declared and initialized somewhere in scope!

IN CONTEXT IN THE METHOD

```
JPanel lastNamePanel = new JPanel();
JTextField lastNameField = new JTextField(10);
lastNamePanel.add(new JLabel("Last Name:"));
lastNamePanel.add(lastNameField);

JPanel buttonPanel = new JPanel();
JButton submit = new JButton("Submit");
buttonPanel.add(submit);
buttonPanel.add(new JButton("Cancel"));

submit.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String firstName = firstNameField.getText();
        String lastName = lastNameField.getText();
        JOptionPane.showMessageDialog(null, "Hello " + firstName + " " + lastName);
    }
});
```