1 **JavaFX Events and Animation**

CST141

2 **Event Handling**

- GUI components generate *events* when users interact with controls
- Typical events include:
  - Clicking the mouse
  - Moving the mouse
  - Typing in a text box (TextField)

3 **Event Listeners**

- To process an event, the programmer must:
  - Register (declare) an event listener
  - Implement one or more event handler methods
- When an event occurs, GUI component notifies the listener by *calling* the event’s handling method(s)

4 **The EventHandler Interface (Page 1)**

- A class file that implements an interface must include all methods “defined” in the interface
  - From Java API programmer or defined
- EventHandler is an interface used to manage *event listening* and *event handling* for Button’s
  - And other JavaFX GUI components
- Objects instantiated from a class that implements the EventHandler interface “are” event handlers, e.g. "*Is an* EventHandler"

5 **The EventHandler Interface (Page 2)**

- The method handle() is declared inside the EventHandler interface and *must be defined* in any class that implements it
  - E.g. if a user clicks a Button object, (and “event listening” is activated) the handle() “event handler” method is called automatically
- Imported from javafx.event package:

  import javafx.event.EventHandler;

6 **The EventHandler Interface (Page 3)**

- Format:

  private class *EventHandlerClassName* implements EventHandler<ActionEvent>
  { …
  - *implements* instead of extends
- Example:

  private class ButtonEventHandler implements EventHandler<ActionEvent>
  { …

7 **The ActionEvent Class (Page 1)**

- Class that represents the variable *type* of parameter e in the header of the actionPerformed() method
- ActionEvent is the <subtype> for the interface EventHandler
- The variable e is a reference that stores the a reference of the *event* information about the specific GUI component that *triggered the event*
- Imported from javafx.event package:
  import javafx.event.ActionEvent;

8 **The ActionEvent Class (Page 2)**

- Example:

```
private class ButtonEventHandler implements EventHandler<ActionEvent>
{
      public void handle(ActionEvent e)
   { …
```

9 **The handle Method (Page 1)**

- The method handle() is a member of interface EventHandler and must be included in every class that implements the interface
- If user clicks a Button object and event listening is activated, the method handle() is called automatically
- A variable "e" of type ActionEvent is included in the method signature and provides access to ActionEvent methods and properties

10 **The handle Method (Page 2)**

- Example:

```
private class ButtonEventHandler implements EventHandler<ActionEvent>
{
      public void handle(ActionEvent e)
   { …
```

11 **Instantiating an EventHandler Object**

- An ActionListener *class* must have been defined previously
- Format:
  *EventHandlerClass eventHandlerObject* = new *EventHandlerConstructor*();
- Example:
  ButtonEventHandler eventHandler = new ButtonEventHandler();

12 **The setOnAction Method (Page 1)**

- A method of a Button (and other "action listener" GUI components) that assign an EventHandler object to the component
- The EventHandler object is the *argument* to the method
- This method effectively *activates* event listening
- Must be executed for every GUI component that will be an *event listener*

13 **The setOnAction Method (Page 2)**

- Format:
  *GuiComponentObject*.setOnAction( *eventHandlerObject* );
- Example:
  button.setOnAction(eventHandler);
  – The GUI component 'button' is a Button

14 **Steps to Create Event Handler**

- The event handler method:
  1. Create a "nested" class the implements the interface EventHandler (within the JavaFX Application class)
  2. Create a method handle() in that class
- Register event listening:
  3. Instantiate an object from the class that implements the interface EventHandler
  4. For each Button call method setOnAction()

16 **The e.getSource Method**

- A method of type ActionEvent that returns a reference to the object *that activated the event*
- Points to the *address* of the object
- Format:
  e.getSource()
- Example:
  if (e.getSource() == buttonOK) ...
- A parameter of type ActionEvent is declared in the handle() method:
  public void handle( ActionEvent e)

18 **The getText Method**

- Returns the String value currently stored in a TextField (or another GUI component that has a text property) object
- For a TextField, the text property is the value currently displayed in the text box
- Format:
  *TextFieldObject*.getText()
- Example:
  String sFirst = firstNumber.getText();

19 **The setText Method**

- Sets the contents of a TextField object (or some other GUI component that has a text property) to a *new value*
- Format:
  *TextFieldObject*.setText(*string*)
- Example:
  resultField.setText(resultString);

20 **The setEditable Method**

- Sets a boolean value that determines if TextField object may be edited by a user
- Frequently is set to false if the object will be used for *output* only
- Format:
  *TextFieldObject*.setEditable(true/false)
- Example:
  resultField.setEditable(false);

22 **The selectAll Method**

- A method of class TextField (inherited from class TextComponent ← TextField) that selects all the text in the object
  – As if it had been selected with a mouse
- Format:
  *jComponentObject*.selectAll();
- Example:
  inputAge.selectAll();

33 **Lambda Expression Event Handling (Page 1)**

- Lambda expression event handling is a new feature in Java 8 which *replaces* the anonymous inner class with a *more consise syntax*
- Also defined within the setOnAction() method combining creation of the object (Button or other object) with a single *method* that replaces the class

34 **Lambda Expression Event Handling (Page 2)**

- Format:

```
ClassName object = new ConstructorName(…);
object.setOnAction( (e) ->
{
   statements
}
);
```

  – The parameter variable e (or other programmer-defined variables) may be explicitly declared by type or the type inferred by the compiler)

35 **Lambda Expression Event Handling (Page 3)**

- Example:

```
Button buttonOK = new Button("OK");
buttonOK.setOnAction( (e) ->
{
   System.out.println("OK clicked");
}
);
```

38 **Lambda Expression Event Handling (Page 4)**

- The Lambda expression may point directly to a *method call*

 Also the parameter variable e does not have to be wrapped inside (parentheses)

39 **Lambda Expression Event Handling (Page 5)**

 Format:
*ClassName object* = new *ConstructorName*(…);
*object*.setOnAction( e -> *methodCall*() );

 Example:
Button buttonOK = new Button("OK");
buttonOK.setOnAction( e -> System.out.println("OK clicked") );

41 **The PathTransition Class (Page 1)**

 Used to create a “path” which is the “border” of one shape node along which another node travels, e.g.:
– A Rectangle node object traverses along the outer border of a Circle node object
– An ImageView node object displaying an image traverses along a Line node object

 Imported from javafx.animation package:
import javafx.animation.PathTransition;

42 **The PathTransition Class (Page 2)**

 Format to instantiate a PathTransition object:
PathTransition *object* = new PathTransition();

 Example:
PathTransition path = new PathTransition();

43 **The setDuration Method (Page 1)**

 For a PathTransition object, sets the amount of time that it takes the node object to traverse the “path” one time

 Amount of time is measure in milliseconds (1000 milliseconds is one second)
– Default is 400 milliseconds (0.4 seconds)

44 **The setDuration Method (Page 2)**

 The setDuration() method takes an argument of type Duration from the class by the same name, e.g. Duration.millis(*double*)

 Imported from javafx.util package:
import javafx.util.Duration;

45 **The setDuration Method (Page 3)**

 Format:
*pathTransitionObject*.<u>setDuration</u>( Duration.millis(*double*) );

 Example:
path.setDuration( Duration.millis(5000) );
– 5000 milliseconds is five seconds

46 **The setPath Method**

 For a PathTransition object, sets (names) the node (e.g. Circle, Rectangle, Line, etc.)

object that is the "path" for another node object to follow

- Format:

  *pathTransitionObject*.setPath(*nodeObject*);

  – *nodeObject* becomes the "path"
- Example:

  path.setPath(circle);

47 **The setNode Method**

- For a PathTransition object, sets (names) the animated node (e.g. Circle, Rectangle, etc.) that follows the "path"
- Format:

  *pathTransitionObject*.setNode(*nodeObject*);

  – *nodeObject* is the node that follows the "path"
- Example:

  path.setNode(rectangle);

48 **The setOrientation Method (Page 1)**

- For a PathTransition object, sets the "upright orientation" of the node object along path
- The method takes one of two enum constants from PathTransition.Orientation:

  PathTransition.OrientationType.NONE

  • Which means that the node stays upright (default)

  PathTransition.OrientationType.ORTHOGONAL_TO_TANGENT

  • Which means that the node rotates to keep perpendicular with the path

49 **The setOrientation Method (Page 2)**

- Format:

  *pathTransitionObject*.setOrientation( *orientationType* );
- Examples:

  path.setOrientation( PathTransition.OrientationType.ORTHOGONAL_TO_TANGENT );

  path.setOrientation( PathTransition.OrientationType.NONE );

50 **The setCycleCount Method (Page 1)**

- For a PathTransition object, sets the number of times the traversal of the "path" will be repeated

  – Default is 1.0
- Inherited from Animation class
- Format:

  *pathTransitionObject*.setCycleCount(*int*);

  – *int* is the number of repetitions
- Examples:

  path.setCycleCount(5);

51 **The setCycleCount Method (Page 2)**

- The INDEFINITE constant from class Timeline specifies that an animation repeats

indefinitely
- Class is imported from javafx.animation package:
  import javafx.animation.Timeline;
- Format:
  *pathTransitionObject*.setCycleCount( Timeline.INDEFINITE );
- Example:
  path.setCycleCount(Timeline.INDEFINITE);

52 **The setAutoReverse Method**

- For a PathTransition object, sets boolean property which determines whether the animation reverses direction on each alternating cycle
  – Default is false (in which case the animation loops)
- Inherited from Animation class
- Format:
  *pathTransitionObject*.setAutoReverse( true / false );
- Examples:
  path.setAutoReverse(true);

53 **The play Method**

- For a PathTransition object, starts an animation running (has no effect if already running)
- Inherited from Animation class
- Format:
  *pathTransitionObject*.play();
- Examples:
  path.play();

54 **The pause Method**

- For a PathTransition object, pauses a running animation (has no effect if not currently running)
- Continues from same point when it runs again
- Inherited from Animation class
- Format:
  *pathTransitionObject*.pause();
- Examples:
  path.pause();

55 **The stop Method**

- For a PathTransition object, stops a running animation and resets play to back initial position (has no effect if not currently running)
- Inherited from Animation class
- Format:
  *pathTransitionObject*.pause();
- Examples:

path.pause();

57 **Classes that extend Pane (Page 1)**

- Objects instantiated from a class that extends class Pane contain JavaFX node objects and can be placed directly into a Scene
- Format:
  public class *ClassName* extends Pane { … }
- Example:
  public class StickMan extends Pane { … }

58 **Classes that extend Pane (Page 2)**

- Example to instantiate the object:
  StickMan stickman = new StickMan();
- Example to place object directly into Scene:
  Scene scene = new Scene(stickMan, 300, 300);

59 **The KeyEvent Class (Page 1)**

- The KeyEvent class is a *subtype* for the interface EventHandler that provides functionality for JavaFX applications to respond to keyboard events
  – An alternative to the ActionEvent class
- Imported from javafx.scene.input package:
  import javafx.scene.input.KeyEvent;

60 **The KeyEvent Class (Page 2)**

- Format:
  public class *ClassName* implements EventHandler<KeyEvent>
- Example:
  public class KeyEventHandler implements EventHandler<KeyEvent>

61 **The KeyEvent Class (Page 3)**

- For keyboard events, the class KeyEvent is the object variable type for the "event" parameter in the handle() method
- Example:
  public void handle(KeyEvent e) { … }

62 **The getCode Method (Page 1)**

- For the ActionEvent parameter of method handle(), the getCode() method returns a code for non-displaying keyboard keys, e.g.:
  – DOWN, UP, ALT, CONTROL, etc.
- Format:
  e.getCode

63 **The getCode Method (Page 2)**

- Example:
  switch ( e.getCode() )

```
{
    case DOWN:
        y += 10;
        break;
    case UP:
        y -= 10;
        break;
    case LEFT:
        x -= 10;
        break;
    case RIGHT:
        x += 10;
        break;
}
```

65 **The switch Statement (Page 1)**

- A Java structure that can be used to implement *linear nested* function (if … else if … else if …)
- The value of a *single* variable or expression can be tested for multiple "equal to" values

66 **The switch Statement (Page 2)**

- The keyword break terminates execution of the switch structure when a true code block finishes executing
  - Otherwise program execution will crash into subsequent cases
- A final optional default case may be executed if all the previous cases are false

67 **Format of switch Structure**

```
switch (testExpression)
{
    case value:
        statement(s) to be executed when
          this case is true go here;
        break;
    case value:
        statement(s) to be executed when
          this case is true go here;
        break;
   [case … ]

   [default:
        statement(s) to be executed when
          no case is true go here;]
}
```

68 **Example of switch Structure**

```
switch (procedureCode)
{
   case 101:
      billing = "Teeth cleaning--$50";
      break;
   case 103:
      billing = "Rabies vaccine--$15";
      break;
   default:
      billing = "Invalid code entered";
}
```

69 **Equivalent of switch**

```
if (procedureCode == 101)
{
   billing = "Teeth cleaning--$50";
}
else if (procedureCode == 103)
{
   billing = "Rabies vaccine--$15";
}
else
{
   billing = "Invalid code entered";
}
```

70 **Testing for More than One true case in a switch**

- Two or more true cases may tested for as follows:

```
switch (procedureCode)
{
   case 101:
   case 222:
     billing = "Teeth cleaning--$50";
     break;
   ...
}
```

  – Evaluates true if procedureCode equals either 101 *or* 222