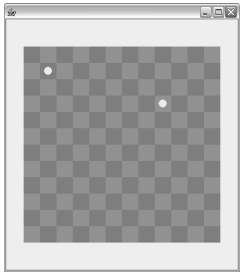
CS133: Developing Programming Principles

Lecture 20

Mouse Events, `Board` (2)

Last Day: A Simple `Board`

- No text, no line.
- Only yellow pegs



Today: Mouse clicks

Event-Driven Programming

- Programming so far: programmer decides what happens next ("flow of control")
- GUIs: user decides what happens next
  - load a file, click a peg (which one?), draw a rectangle, quit the game...
- Idea: GUI generates events. The programmer writes code to react to the events.

Listening for Mouse Events

In order to receive mouse events we must declare the **`BoardPanel`** class as ***implementing*** the **`MouseListener`** ***interface:***

```
public class BoardPanel extends JPanel
             implements MouseListener { …
```

Interfaces

In Java, an interface defines a set of methods that a class must implement. Just like a class, an interface is a type.

(In the remainder of the course, we'll see several interfaces, but we won't study them in detail.)

Listening for Mouse Events

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class BoardPanel extends JPanel
                    implements MouseListener
{
   // etc.
}
```

`MouseListener` methods

```
public void mouseClicked(MouseEvent e)
public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
public void mousePressed(MouseEvent e)
public void mouseReleased(MouseEvent e)
```


`mouseClicked`

```
public void mouseClicked(MouseEvent e)
{
    int x = (int)e.getPoint().getX();
    int y = (int)e.getPoint().getY();

    boardClicked(x, y);
    repaint();
}
```

Class **MouseEvent**

When a mouse action occurs in a GUI component, a **MouseEvent** is generated by Java and delivered to the **MouseListener** for that component.

public Point getPoint()

Returns the x, y position of the event relative to the source component.

repaint

When **repaint** is called it makes a request to Java that eventually results in a call to **paintComponent**.

public void repaint()

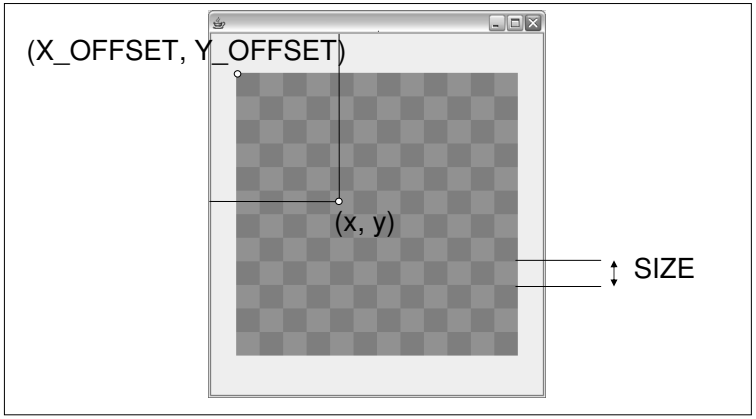
Remaining `MouseListener` Methods

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


`boardClicked`

```
private void boardClicked(int x, int y)
{
    int row, col;
    if (x < X_OFFSET || y < Y_OFFSET) {
      return;
    }
    row = (y - Y_OFFSET)/SIZE;
    col = (x - X_OFFSET)/SIZE;
    flipPeg(row, col);
}
```

(x, y) to (row, col)



Flipping Pegs On and Off

```
public void flipPeg(int row, int col)
{
  if (row >= 0 && row < rows &&
      cols >= 0 && col < cols)
   {
     grid[row][col] ^= true;
   }
}
```

Listening for Mouse Events

Finally, in the constructor we must tell Java that the **`BoardPanel`** class will be responsible for handling its own mouse events.

```
addMouseListener(this);
```


Revised Constructor

```
public BoardPanel(int rows, int cols) {
    int xSize = 2*X_OFFSET + cols*SIZE;
    int ySize = 2*Y_OFFSET + rows*SIZE;

    this.rows = rows;
    this.cols = cols;
    emptyGrid();
    setPreferredSize(
         new Dimension(xSize, ySize));
    addMouseListener(this);
}
```

Listening for Mouse Events

1. Class must **`implements MouseListener`**
2. Define all **`MouseListener`** methods
3. **`addMouseListener(this)`**

Additional Features of **`Board`**

- Window characteristics (title, resizing)
- Implementing **`displayMessage`**
- Implementing **`getClick`**

Window Characteristics

Setting the title of a JFrame:

```
this.setTitle("Board game");
```

Preventing a resize of the JFrame:

```
this.setResizable(false);
```


Revised Constructor

```
public Board(int rows, int cols) {
   Container content = getContentPane();
   board = new BoardPanel(rows, cols);
   content.add(board);
   setDefaultCloseOperation(EXIT_ON_CLOSE);
   pack();
   setVisible(true);

   this.setTitle("Board game");
   this.setResizable(false);
}
```

Displaying Text

```
void drawString(String s, int x, int y)
```

Draws the text given by the specified **`String`**, using a graphics context's current font and color.

Displaying Text

```
public class TextExample extends JPanel
{
  public TextExample()
  {
    setPreferredSize(new Dimension(100, 100));
  }

  public void paintComponent(Graphics canvas)
  {
    canvas.setColor(Color.BLACK);
    canvas.drawString("Hello world!", 20, 50);
  }
}
```

Displaying Text

```
public static void main(String[] args)
{
  JFrame frame = new JFrame();
  JPanel eg = new TextExample();
  Container content = frame.getContentPane();
  content.add(eg);
  frame.setDefaultCloseOperation(
              JFrame.EXIT_ON_CLOSE);
  frame.pack();
  frame.setVisible(true);
}
```

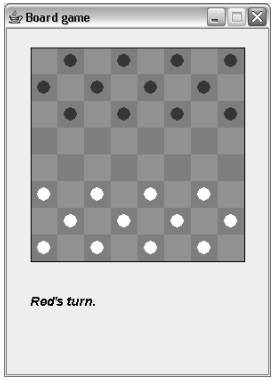

Displaying Text

Implementing `getClick`

`getClick` is tricky to implement. The calling ***thread of execution*** must explicitly wait for the mouse event.

(NOT CS133!)

Checkers



ERROR: undefined
OFFENDING COMMAND:

STACK: