

Week 10, Lecture 1 Hunting the Treasure



1

Week 10

Lecture: **Hunting the treasure &
Building a calculator**

Java Genesis:

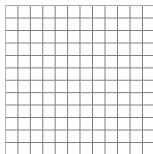
–Ch10: Graphical components

Lab Assessment 8

2

Describing the treasure hunt

Treasure is hidden at one of the squares in a 12 by 12 board of squares.



At the start of the hunt all the squares on the board are white. If we click with the mouse on a square it turns an appropriate colour to indicate its distance from the treasure.

3

Denoting distance by colour

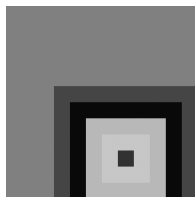
If the distance from the selected square to the treasure is

- 5** or more the square turns **gray**
- 4** the square turns **purple**
- 3** the square turns **blue**
- 2** the square turns **green**
- 1** the square turns **orange**
- 0** the square turns **red** (treasure found)

4

Measuring distance to treasure

distance to treasure: the smallest number of steps to reach the treasure, where a step is a move to an adjacent horizontal, vertical or diagonal square.



5

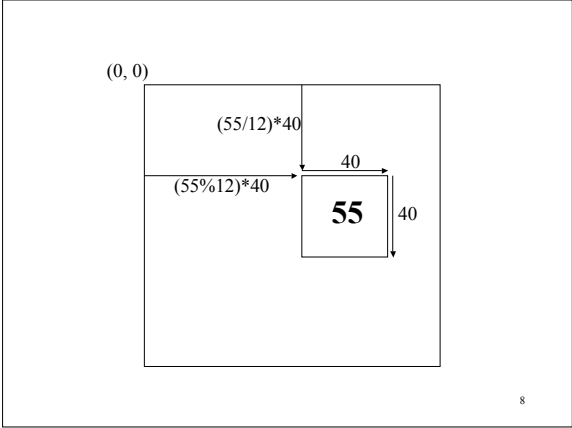
So squares on the board can be referenced, number them from 0 to 143 as shown here:

0	1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69	70	71
72	73	74	75	76	77	78	79	80	81	82	83
84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107
108	109	110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129	130	131
132	133	134	135	136	137	138	139	140	141	142	143

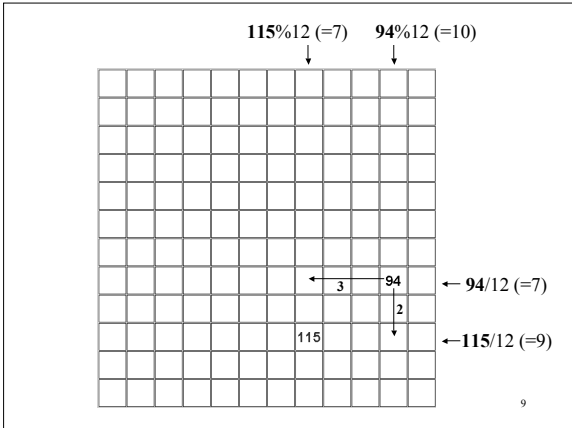
6

$n\%12 = 0$												
0	1	2	3	4	5	6	7	8	9	10	11	← $n/12 = 0$
12	13	14	15	16	17	18	19	20	21	22	23	
24	25	26	27	28	29	30	31	32	33	34	35	
36	37	38	39	40	41	42	43	44	45	46	47	
48	49	50	51	52	53	54	55	56	57	58	59	← $55/12 = 4$
60	61	62	63	64	65	66	67	68	69	70	71	
72	73	74	75	76	77	78	79	80	81	82	83	
84	85	86	87	88	89	90	91	92	93	94	95	
96	97	98	99	100	101	102	103	104	105	106	107	
108	109	110	111	112	113	114	115	116	117	118	119	
120	121	122	123	124	125	126	127	128	129	130	131	
132	133	134	135	136	137	138	139	140	141	142	143	← $n/12 = 11$

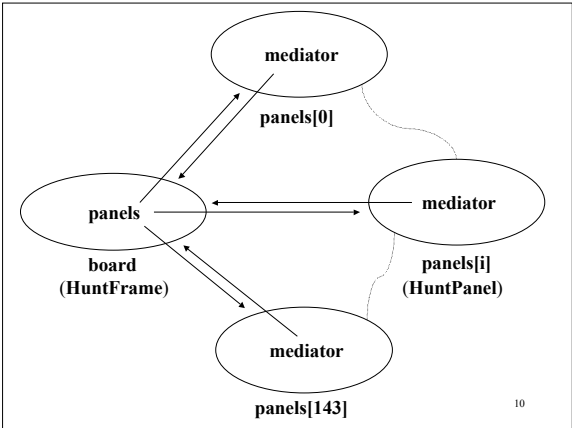
7



8



9



10

```

public class Main {
    public static void main (String [ ] args) {
        HuntFrame board = new HuntFrame ( );
        board.setVisible(true);
    }
}

```

11

```

import java.awt.*;
import genesis.*;

public class HuntFrame extends ExitFrame {
    // instance variables
    private HuntPanel [ ] panels =
        new HuntPanel [144];
    private int treasurePosn =
        (int)(144*Math.random( ));
    // constructor
    public HuntFrame ( ) {
        setTitle("Treasure Hunt");
        setBounds(300, 100, 12*40+10, 12*40+35);
    }
}

```

12

```

Container c = getContentPane( );
c.setLayout(null);
for (int i=0; i<144; i++) {
    panels[i] =
        new HuntPanel(i, treasurePosn, this);
    panels[i].setBounds
        ((i%12)*40, (i/12)*40, 40, 40);
    c.add(panels[i]);
}
public void showColourOfRest ( ) {
    for (int i=0; i<144; i++)
        if (i != treasurePosn)
            panels[i].showColour( );
}
}

```

13

```

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

public class HuntPanel extends JPanel {

    // instance variables
    private Color [ ] colours =
        {Color.red, Color.orange,
         Color.green, Color.blue,
         Color.magenta, Color.gray};

    private int colourNum;
    private int index;
    private int treasurePosn;
    private HuntFrame mediator;
}

```

14

```

// constructor
public HuntPanel
    (int in, int posn, HuntFrame med) {
    index = in;
    treasurePosn = posn;
    mediator = med;
    setBackground(Color.white);
    addMouseListener(new MouseAdapter( ) {
        public void mousePressed(MouseEvent e) {
            showColour( );
        }
    });
}
}

```

15

```

public int showColour ( ) {
    int horizDist =
        Math.abs(index%12-treasurePosn%12);
    int vertDist =
        Math.abs(index/12-treasurePosn/12);
    colourNum = Math.max(horizDist, vertDist);
    if (colourNum > 5) colourNum = 5;
    setBackground(colours[colourNum]);
    if (colourNum == 0)
        mediator.showColourOfRest( );
}
}

```

16

Building a Calculator

Specification:

We need to specify both the **design**
and **functionality** of the calculator.

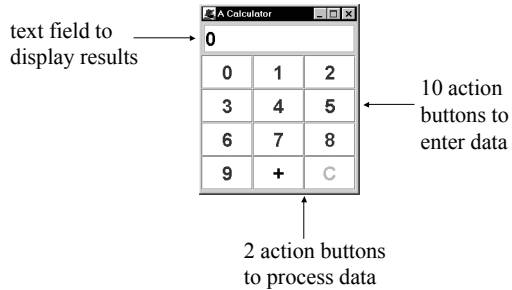
17

Calculator Design

We need to specify the number, type and layout of the GUI components, i.e. list all the action buttons and text fields and decide where to position them.

18

Specifying the Design



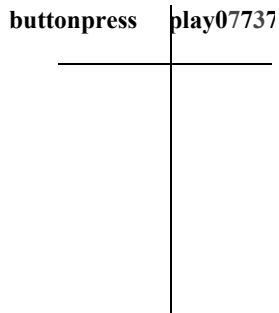
19

Calculator Functionality

We need to specify how the calculator works, i.e. describe exactly what happens when any button is pressed in any situation.

20

Example



21

Specifying Functionality

- When a number button is pressed the associated digit is appended to the end of the sequence of digits in the display, except in the following cases:
 - if the previously pressed button was + the associated digit alone is shown in the display;
 - if 0 is pressed when 0 is being displayed, the display continues to show just 0.

22

Specifying Functionality (cont)

- There is a (hidden) register that records the 'sum so far'. Initially the register is 0.

continued....

23

Specifying Functionality (cont)

- When + is pressed the integer in the register is added to the display. This sum is then
 - shown in the display, and
 - stored in the register.

continued....

24

Specifying Functionality (cont)

- When C is pressed
 - 0 is shown in the display, and
 - 0 is stored in the register.

25

```
public class Main {  
    public static void main (String [ ] args) {  
        Calculator calcWin = new Calculator();  
        calcWin.setVisible(true);  
    }  
}
```

26

```
import java.awt.*;  
import java.awt.event.*;  
import javax.swing.*;  
import genesis.*;  
  
public class Calculator extends JFrame {  
    // instance variables  
    private JTextField display =  
        new JTextField("0", 9);  
    private JButton [ ] numButs =  
        new JButton [10];  
    private JButton plus = new JButton("+");  
    private JButton clear = new JButton("C");  
    private int register = 0;  
    private boolean start = true;
```

27

```
// constructor  
public Calculator ( ) {  
    setTitle("A Calculator");  
    setBounds(400, 200, 250, 300);  
    Container c = getContentPane();  
    display.setEditable(false);  
    display.setFont  
        (new Font("SansSerif", Font.BOLD, 30));  
    display.setBackground(Color.white);  
    JPanel pTop = new JPanel( );  
    pTop.add(display);  
    c.add(pTop, "North");  
    JPanel pBot = new JPanel( );  
    pBot.setLayout(new GridLayout(4, 3));
```

28

```
for (int i=0; i<10; i++) {  
    numButs[i] = new JButton(""+i);  
    numButs[i].setFont  
        (new Font("SansSerif",Font.BOLD,30));  
    numButs[i].setBackground(Color.white);  
    numButs[i].setForeground(Color.red);  
    numButs[i].addActionListener(new ActionListener ( ) {  
        public void actionPerformed(ActionEvent e) {  
            String butVal = e.getActionCommand( );  
            if (start || display.getText( ).equals("0"))  
                display.setText(butVal);  
            else display.setText(display.getText( )+butVal);  
            start = false;  
        }  
    });  
    pBot.add(numButs[i]);  
}
```

29

```
plus.setFont (new Font("SansSerif", Font.BOLD, 30));  
plus.setBackground(Color.white);  
plus.setForeground(Color.blue);  
plus.addActionListener(new ActionListener ( ) {  
    public void actionPerformed(ActionEvent e) {  
        int displayVal=Integer.parseInt(display.getText());  
        register = register + displayVal;  
        display.setText(""+register);  
        start = true;  
    }  
});  
pBot.add(plus);
```

30

```
clear.setFont(new Font("SansSerif", Font.BOLD, 30));
clear.setBackground(Color.white);
clear.setForeground(Color.green);
clear.addActionListener(new ActionListener ( ) {
    public void actionPerformed(ActionEvent e) {
        register = 0;
        display.setText(""+0);
        start = true;
    }
});
pBot.add(clear);
c.add(pBot, "Center");
}
```

31