61FIT3JSD
Fall 2022

Lecture 9

GUI programming (4)
Advanced issues

Lecture outline

- Tabular display: JTable
- GUI tool kit: Font, Color
- Custom GUI using drawing

Exer
cise

Lecture 8 review exercise

- Extend `MyApp` application to:
  - validate data entered by user
  - display an info. message for successful data entry
  - display an error message for erroneous data entry
  - support the functionality for two domain entities Customer and Order

Tabular display: `JTable`

- Swing provides `JTable` to display data in a tabular form
- A table contains a header row and one or more rows of data
- The header row is an array of column names
- A data row is an array of values (possibly of different types)
- A column is an array of values of the same type
- Objects of different types can be displayed in a table

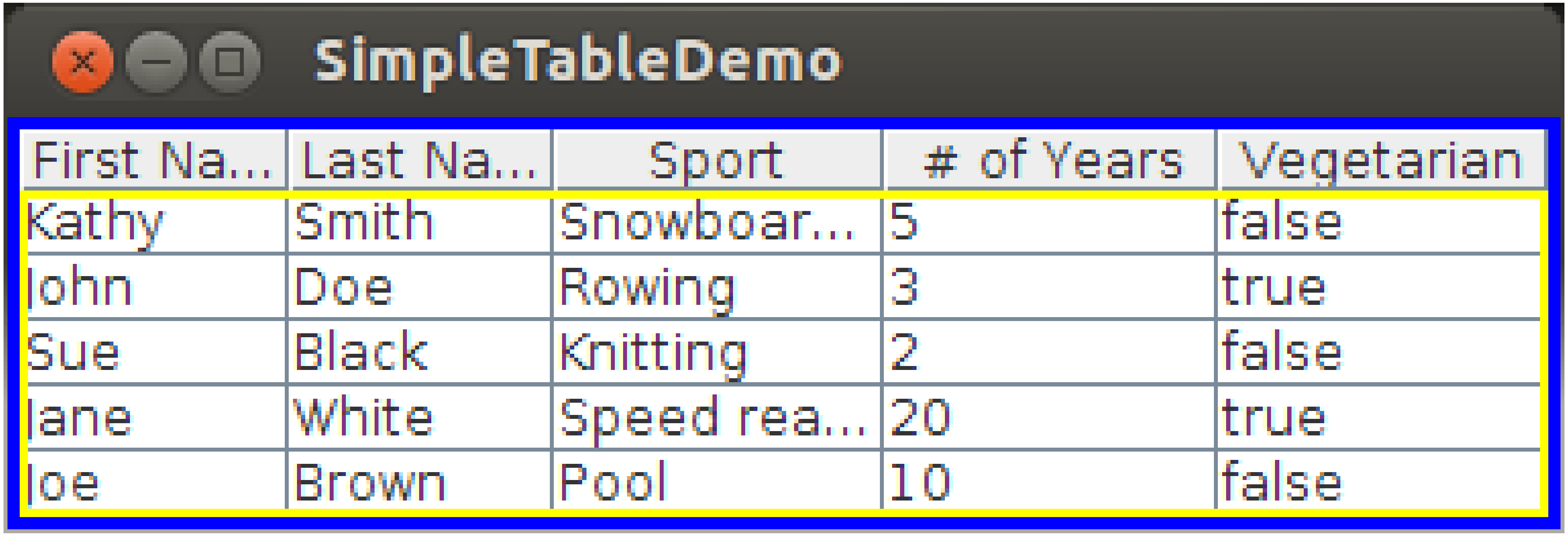
A simple `JTable`

- Create headers
- Create data rows
- Create a `JTable` object
- Put table object into a scroll bar object
- Add scroll bar object to window

DEMO

A simple JTable

```
lect09.tables.SimpleTableDemo
```



First Na...	Last Na...	Sport	# of Years	Vegetarian
Kathy	Smith	Snowboar...	5	false
John	Doe	Rowing	3	true
Sue	Black	Knitting	2	false
Jane	White	Speed rea...	20	true
Joe	Brown	Pool	10	false

Create headers

```
Object[] head = {
        "First Name",
        "Last Name",
        "Sport",
        "# of Years",
        "Vegetarian" };
```

Create data rows

```
Object[][] data = {
   {"Kathy","Smith","Snowboarding",5,false},
   {"John","Doe","Rowing",3,true },
   {"Sue","Black","Knitting",2,false },
   {"Jane","White","Speed reading",20,true},
   {"Joe","Brown","Pool",10,false}
};
```

Create a JTable object

```
JTable table = new JTable(data, head);
```

Put table object into a scroll bar object

```
// put table in a scroll bar
JScrollPane scroll = new JScrollPane(table);
```

Add scroll bar object to window

```
// add scroll bar to a window
w.add(scroll);
```

Table model

- Class: `DefaultTableModel`,

  `AbstractTableModel`,

  `TableModel`

- Manages the table data
- To get the table model:

```
getModel(): TableModel
```

- To change the table model:

```
setModel(TableModel)
```

Column model

- Class: `DefaultTableColumn`,
  `TableColumnModel`
- Manages all the table columns
- To get the column model:

```
getColumnModel(): TableColumnModel
```

- To change the column model:

```
setColumnModel(TableColumnModel)
```

Table header

- Class: `JTableHeader`
- Manages the table header
- To get the table header:

```
getTableHeader(): JTableHeader
```

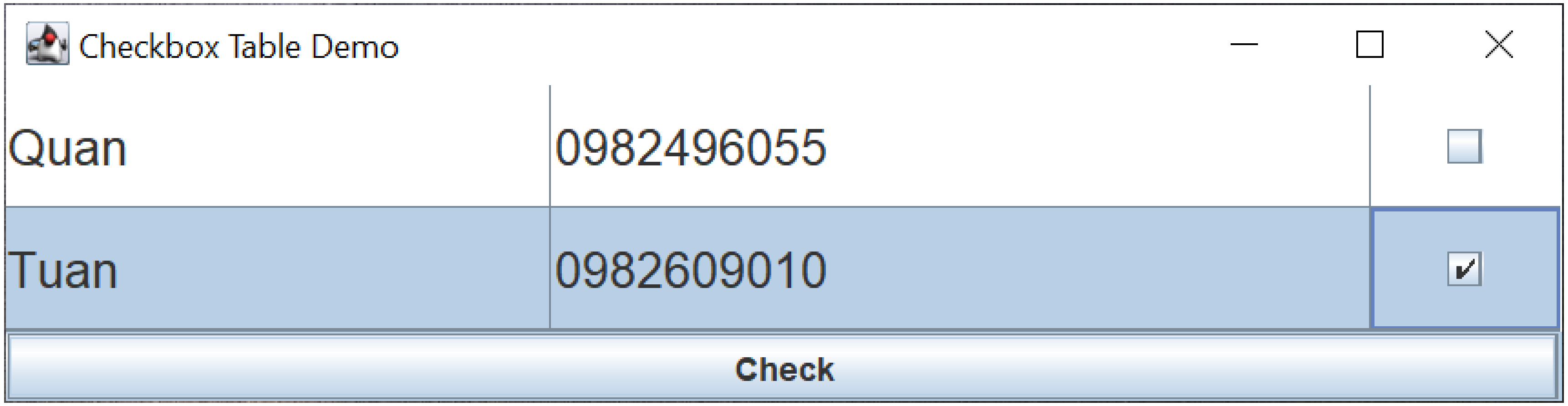
- To change the table header:

```
setTableHeader(JTableHeader)
```

DEMO

JTable with Checkboxes

```
lect09.tables.CheckBoxTableDemo
```



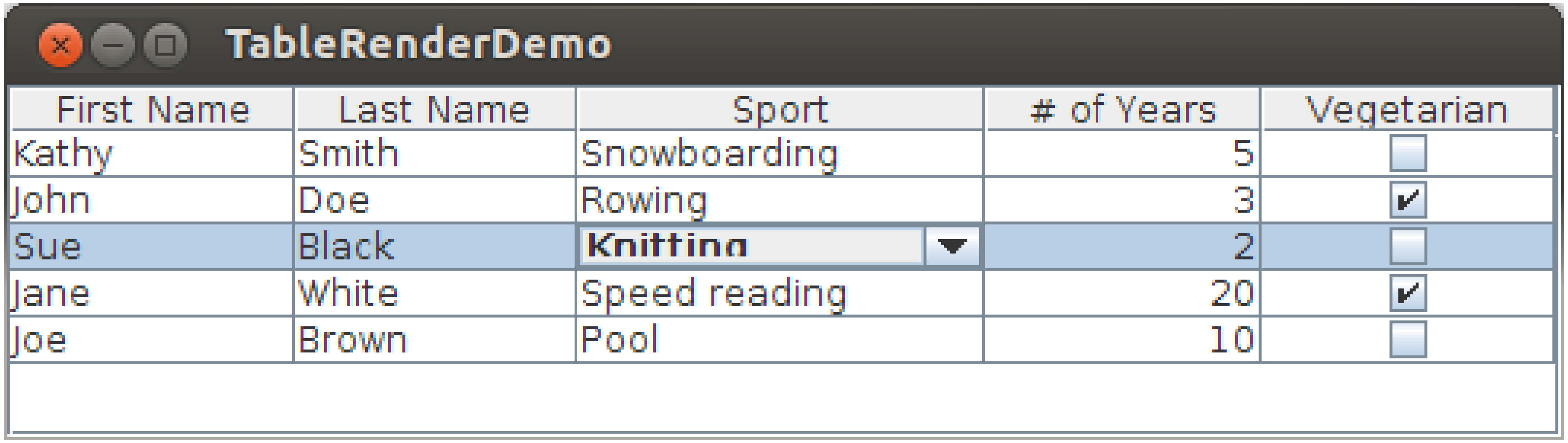
Checkbox Table Demo

Quan	0982496055	☐
Tuan	0982609010	✓

DEMO

Another JTable example

```
lect09.tables.TableRenderDemo
```



First Name	Last Name	Sport	# of Years	Vegetarian
Kathy	Smith	Snowboarding	5	☐
John	Doe	Rowing	3	✓
Sue	Black	Knitting	2	☐
Jane	White	Speed reading	20	✓
Joe	Brown	Pool	10	☐

GUI tool kit

- Font
- Custom color
- Display tool kit

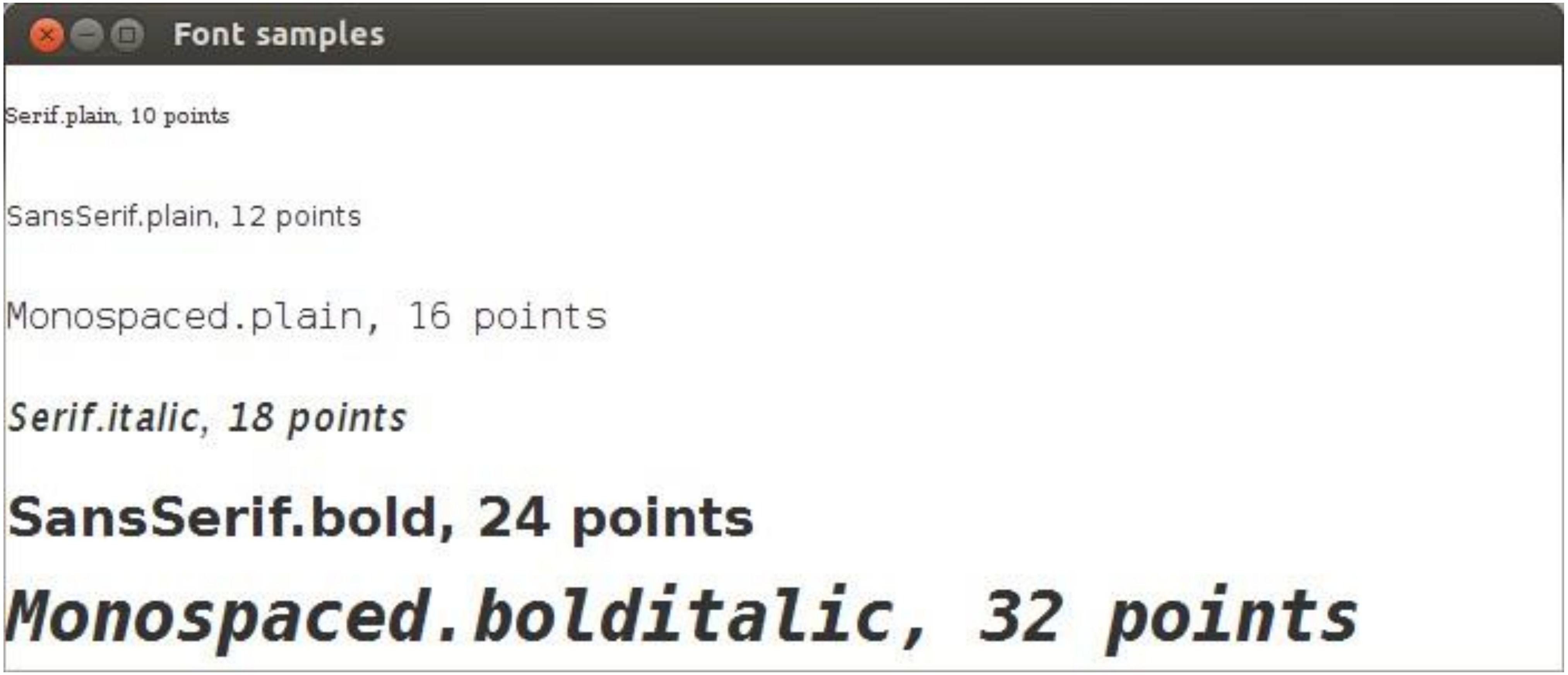
Font

- Class: `java.awt.Font`
- Constructor arguments:
  - family name: e.g. Times, SansSerif, Monospace,
  - style: Font.PLAIN, Font.BOLD, Font.ITALIC
  - size: number of points (point = 1/72 inch)
- A new Font can be derived from an existing one.

DEMO

Font

`lect09.font.FontDemo`



```
Serif.plain, 10 points
SansSerif.plain, 12 points
Monospaced.plain, 16 points
Serif.italic, 18 points
SansSerif.bold, 24 points
Monospaced.bolditalic, 32 points
```

Custom colour

- Class: `java.awt.Color`
- Create a new Color object with arguments red, green, and blue
- R,G,B values are either in [0,255] or [0,1]:

```
// using integral
Color brown = new Color(200,150,0);
// using float: (float) 200/255, ...
```

Useful Color methods

- `getRed(): int`
  - returns the red value in the range [0,255]
- `getGreen(): int`
  - returns the green value
- `getBlue(): int`
  - returns the blue value
- `brighter(): Color`
  - returns a brighter color of the current one
- `darker(): Color`
  - returns a darker color of the current one

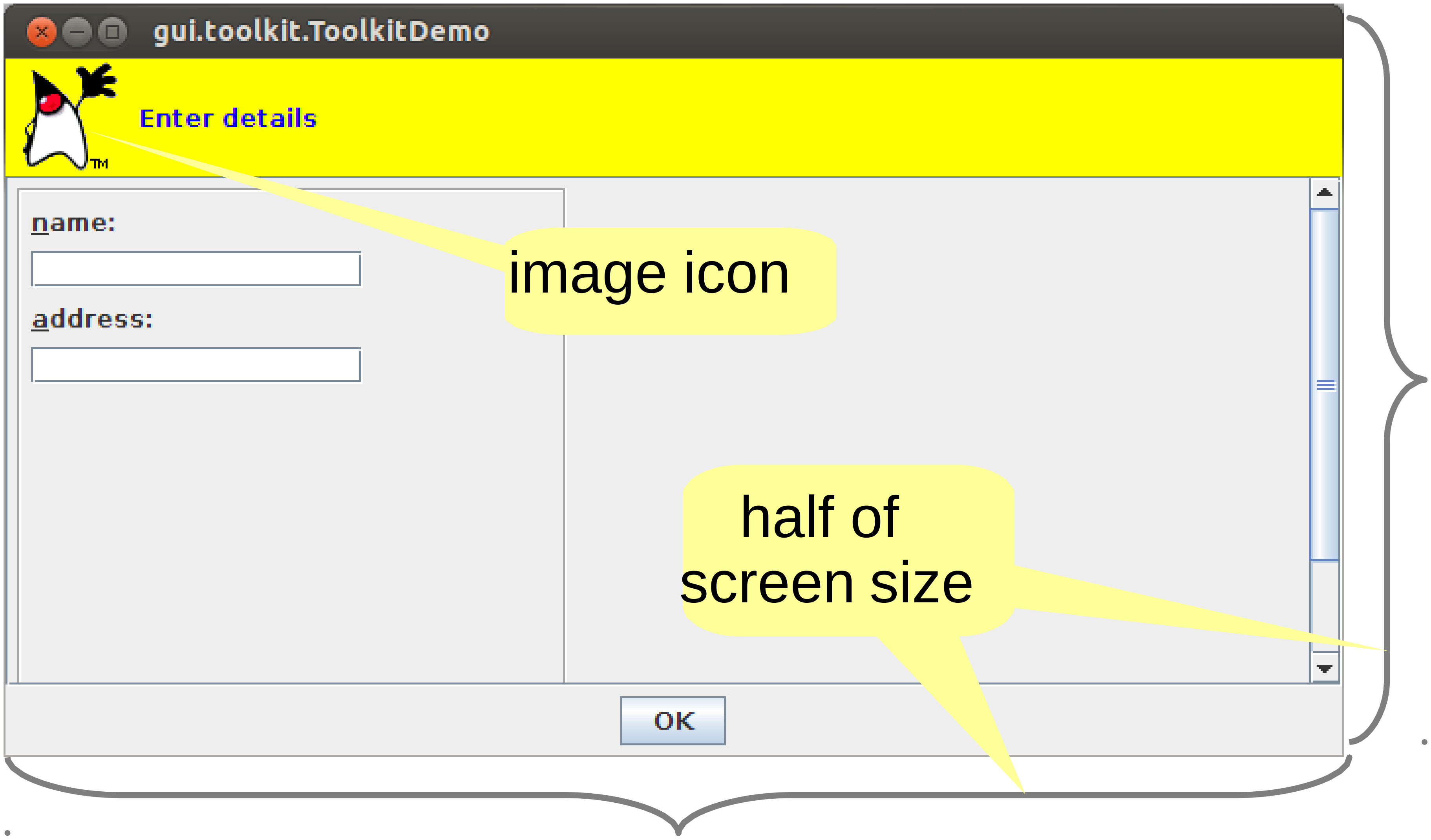
Display tool kit

- Class: `java.awt.Toolkit`
- Provides utility methods for:
  - getting screen size
  - creating an image from a file

DEMO

Tool kit

```
lect09.toolkit.ToolkitDemo
```

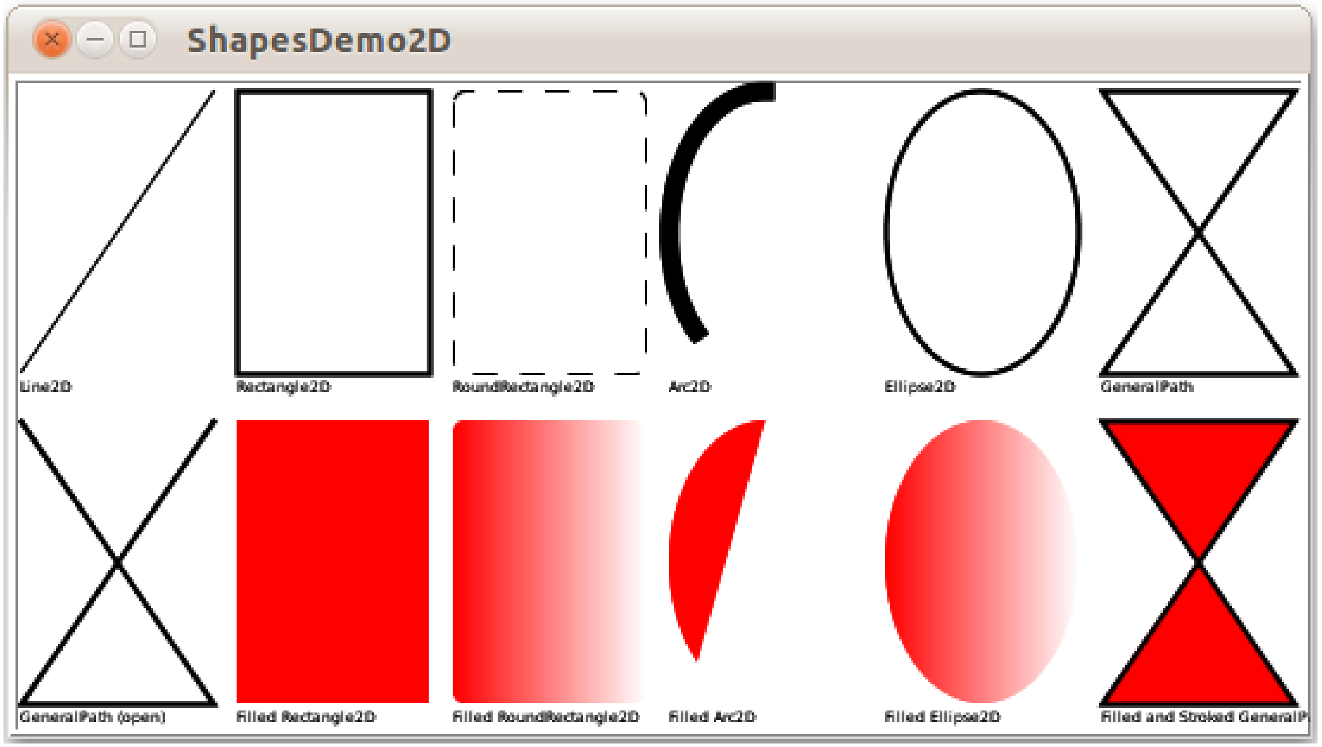


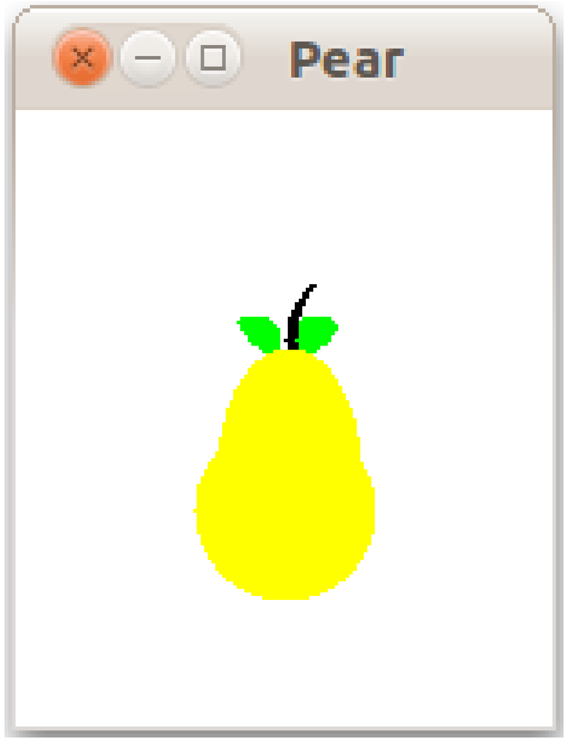
Custom GUI using drawing

- GUI drawing is used for special graphical requirements
- Every Swing component has an associated graphics object
- Class: `java.awt.Graphics`,

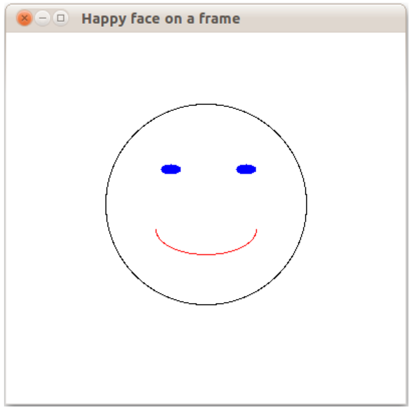
  `java.awt.Graphics2D`

Basic examples





basic shapes



a happy face

a pear!

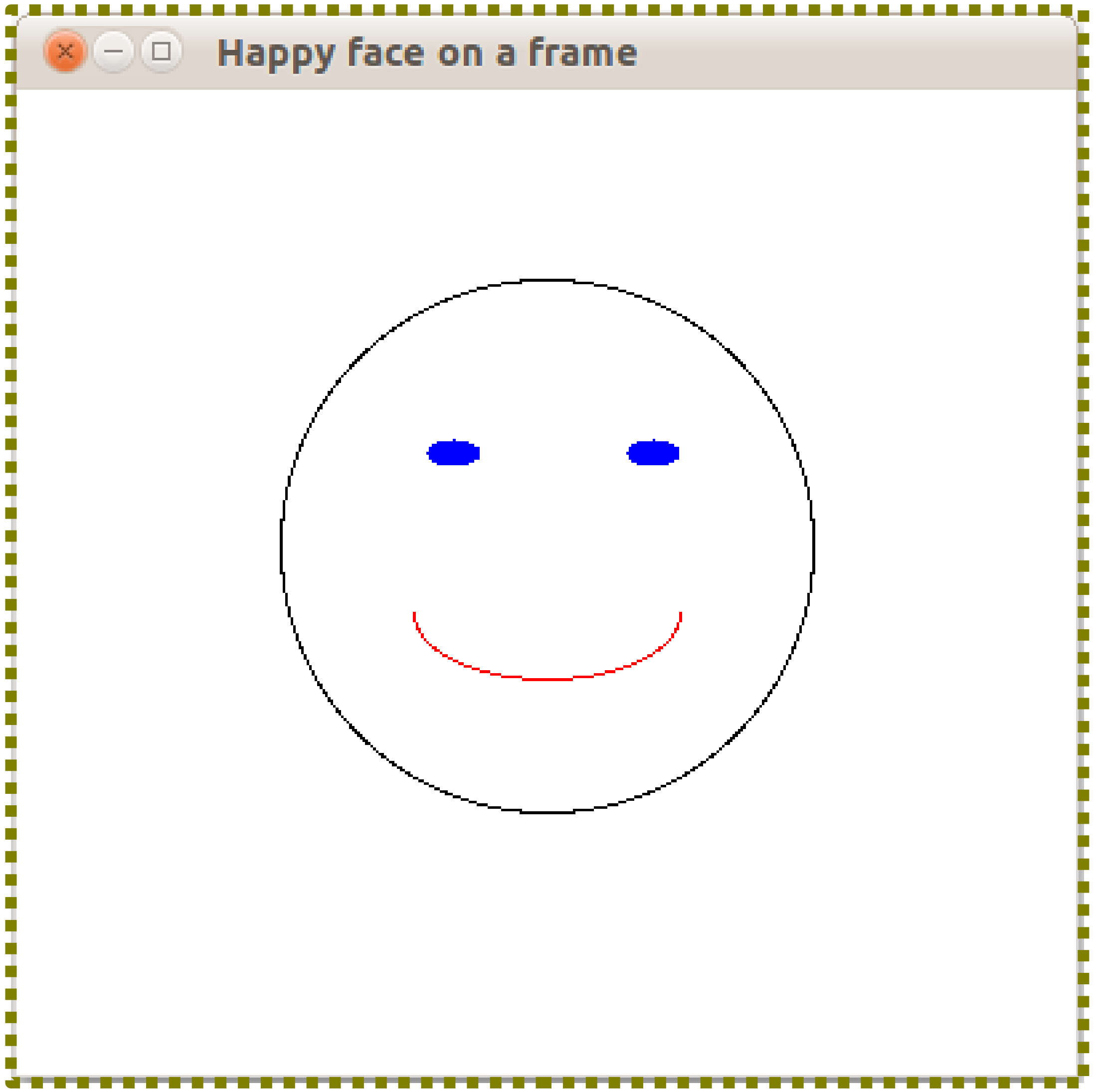
Class `Graphics`

- Encapsulates the GUI-related state of a display component
- Specifies the display area of the component:
  - all drawings on the component will appear within this area
- Provides methods for drawing primitive shapes (lines, circles, rectangles, etc.):
  - `drawX()`: draws an outline of shape X
  - `fillX`(): fills the area defined by `drawX()`
- Font and color of each drawing can be changed

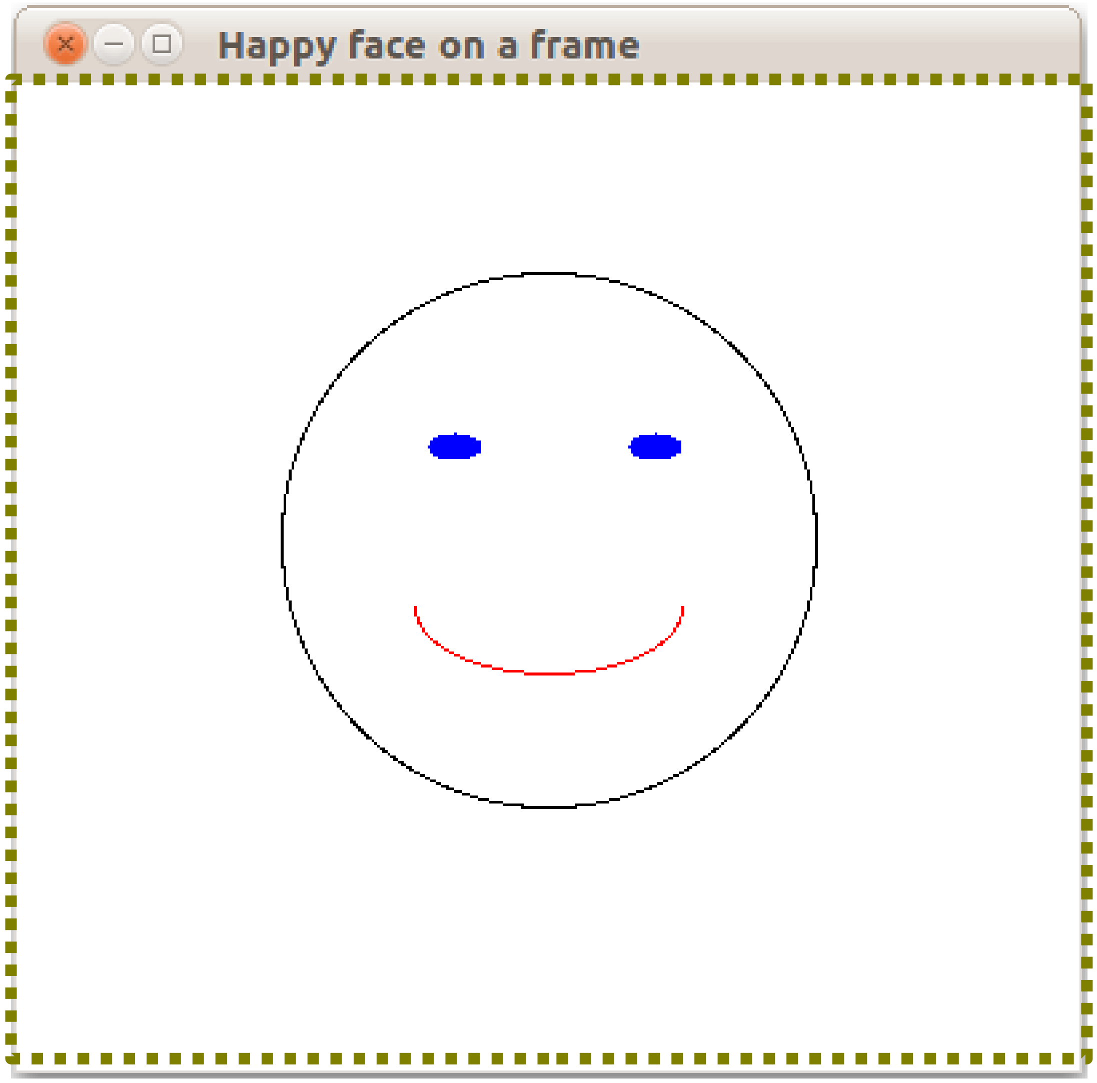
Display area

- Top-level container (e.g. JFrame):
  - the entire window
- Nontop-level container:
  - display area of the container of the component

Top-level display area



Component display area



GUI drawing basics

- Sub-class a `JComponent` object:
  - *commonly* `JFrame` or `JPanel`
- Override the `paint` or `paintComponent` method:
  - must invoke the super-class method first!
  - use `Graphics` object to draw the desired basic shapes
- Display the object:
  - use a `JFrame` if object is not a top-level container

Using JFrame

```
public class HappyFaceColor extends JFrame {
   public HappyFaceColor() {
      // set up this frame
   }
   @Override
   public void paint(Graphics g) {
      // invoke super class method first!
      super.paint(g);
      // custom drawing using g
   }
}
```

Using JPanel (1)

```
class Drawing extends JComponent {
  @Override
  public void paintComponent(Graphics g) {
    // invokes super class method first!
    super.paintComponent(g);
    // custom drawing using g
  }
}
```

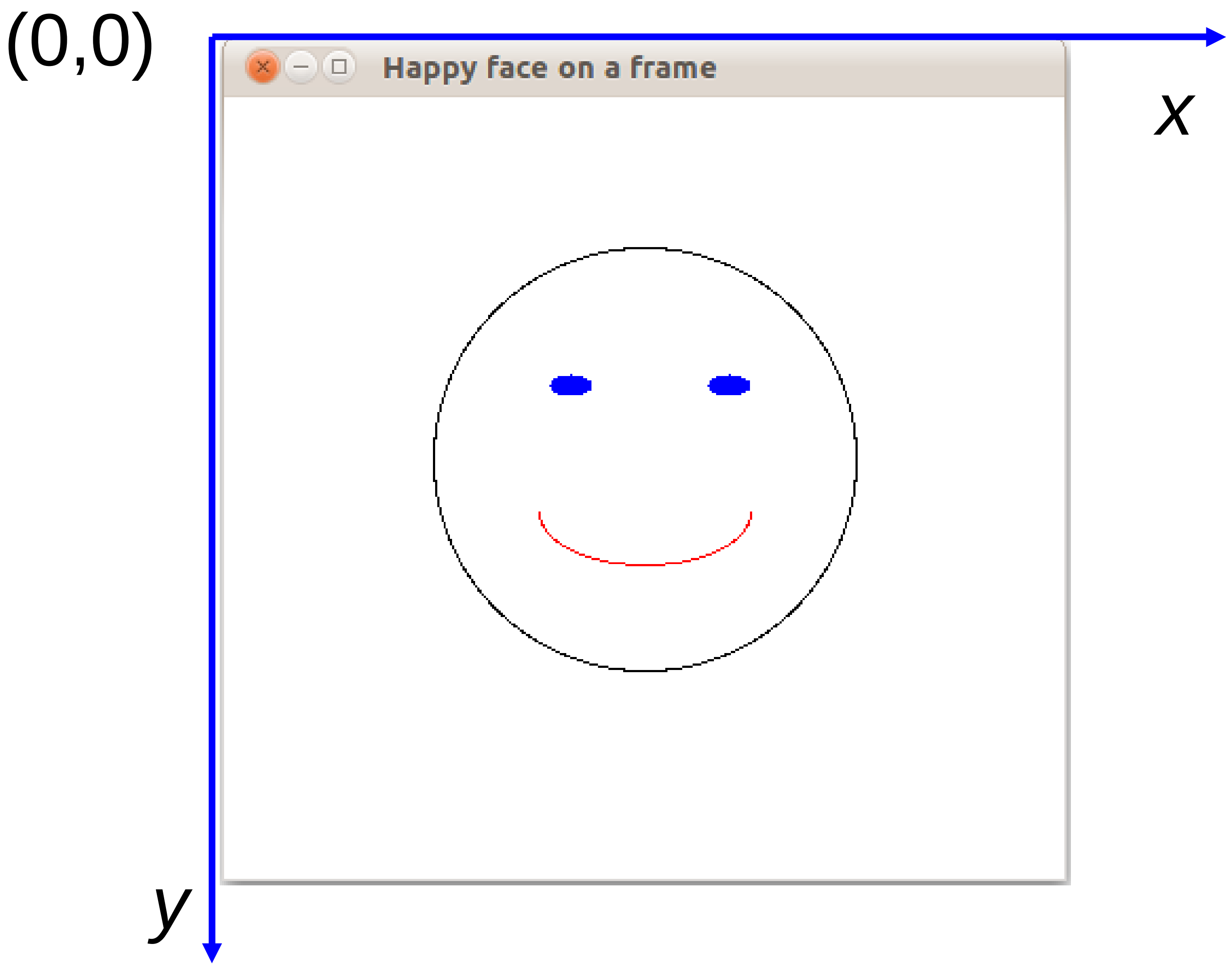
Using JPanel (2)

```
public class DrawingApp {
   private JFrame frame;
   public void createAndShowGUI() {
      // ...
      // setup drawing
      Drawing draw = new Drawing();
      // setup drawing panel
      JPanel drawPanel = new JPanel();
      drawPanel.add(draw);
      ...
      frame.add(drawPanel);
      // ...
   }
   ...
}
```

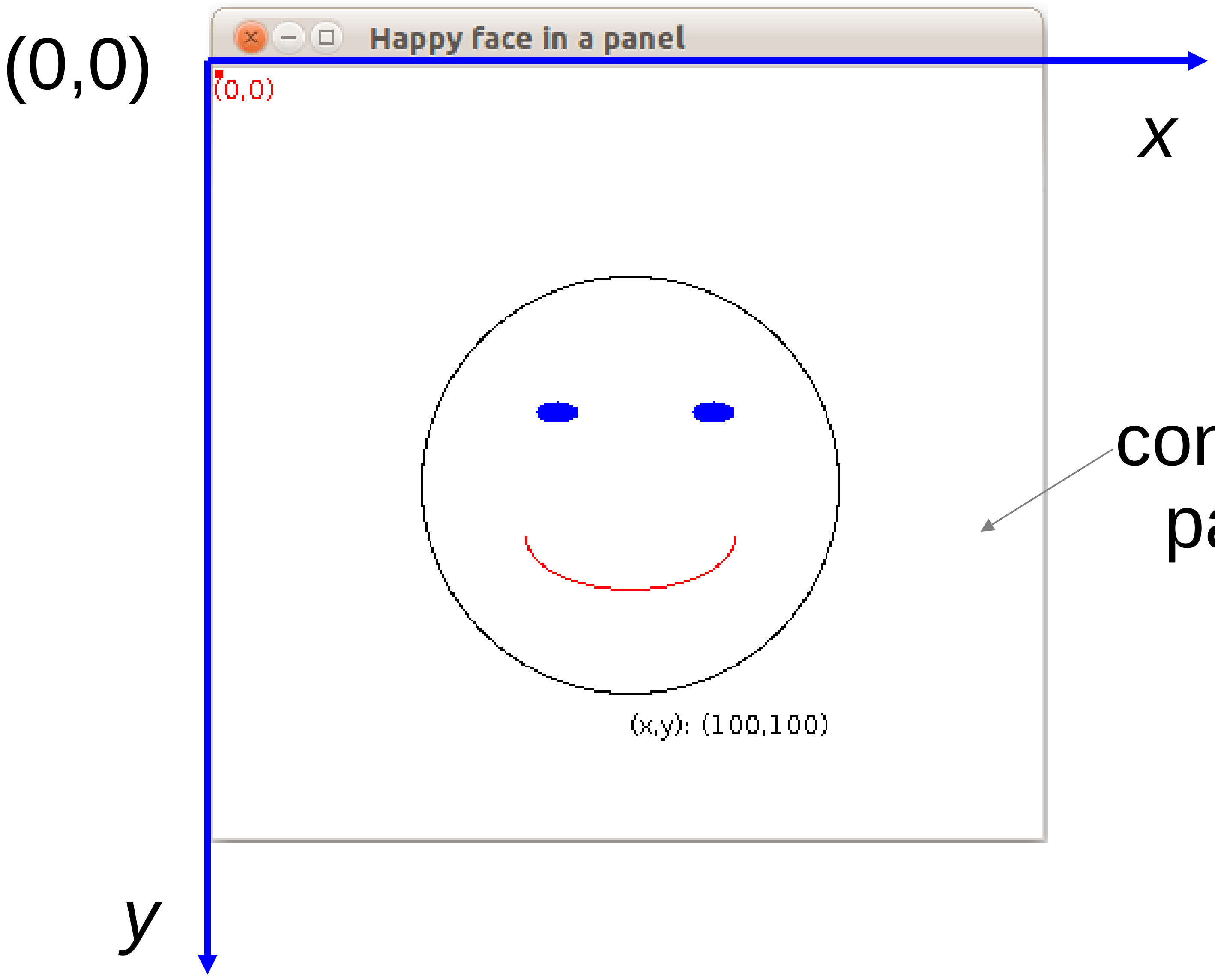
Drawing coordinate space

- The coordinate space of the component display area:
  - origin (0,0) is the top-left corner
  - x-axis extends rightward
  - y-axis extends downward

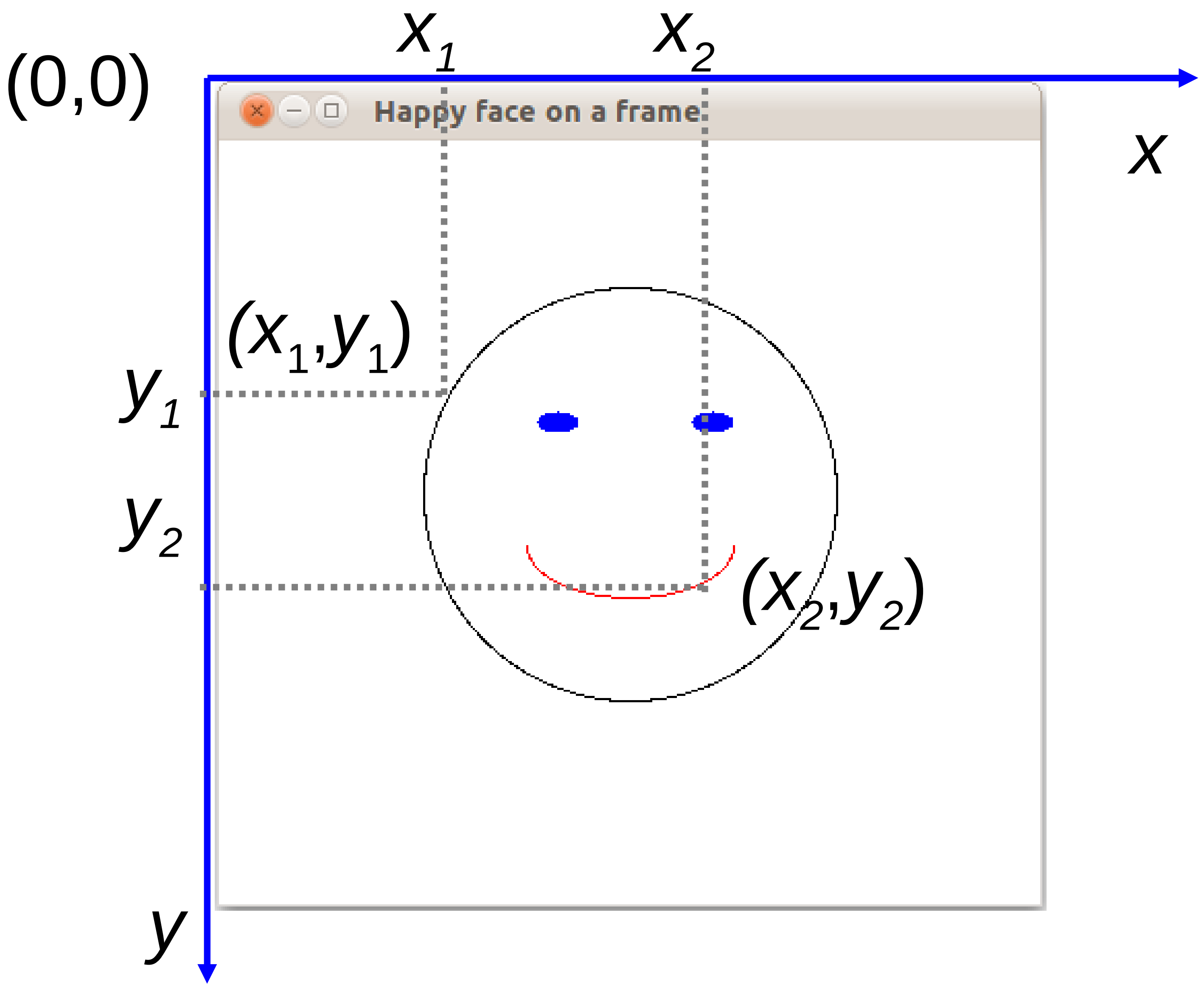
Coordinate space: JFrame



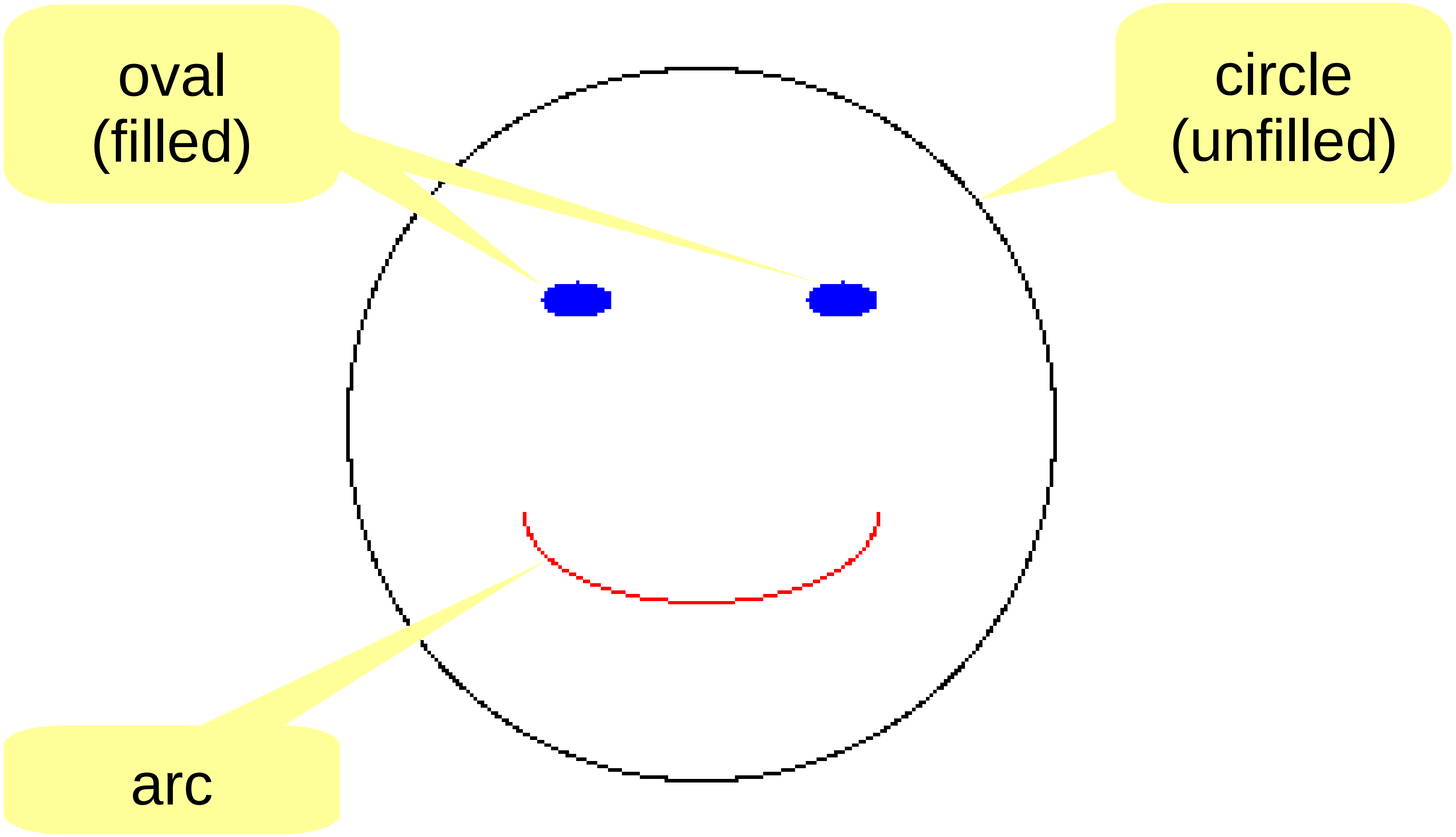
Coordinate space: content pane



Drawing points



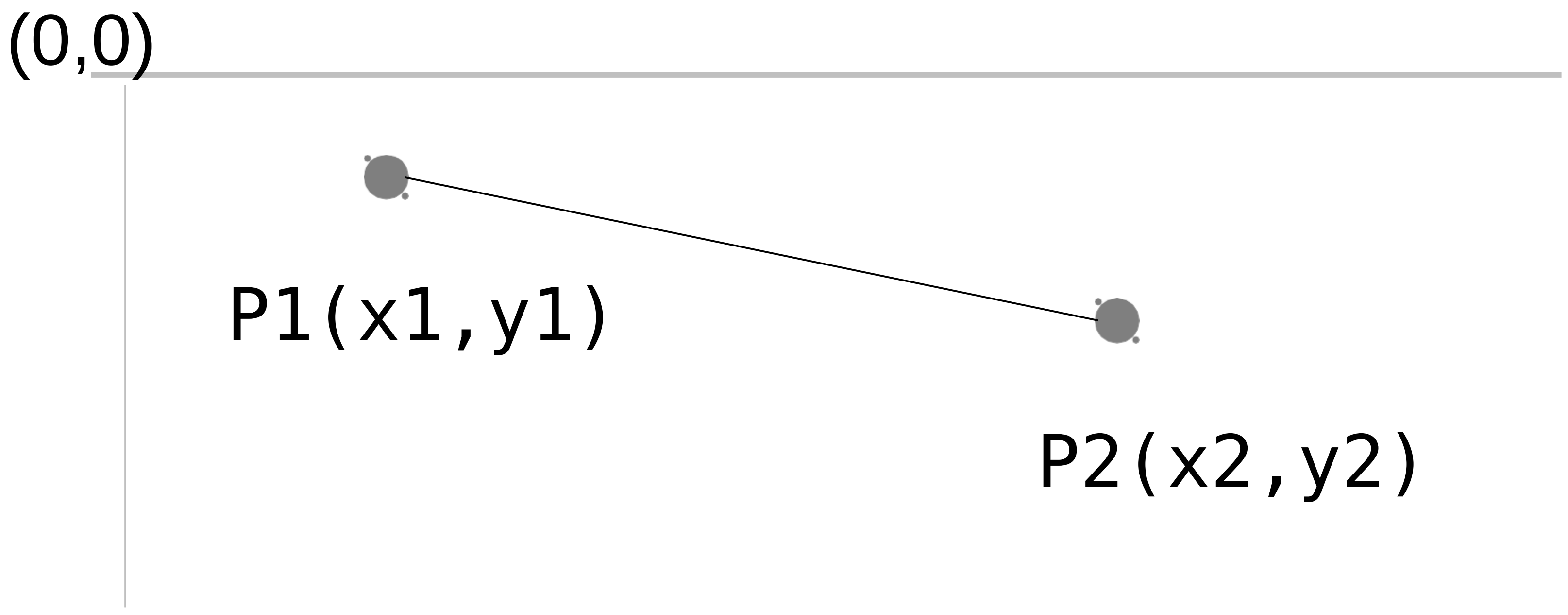
Basic drawing shapes



drawLine()

```
public void drawLine(int x1, int y1,
                                  int x2, int y2)
```

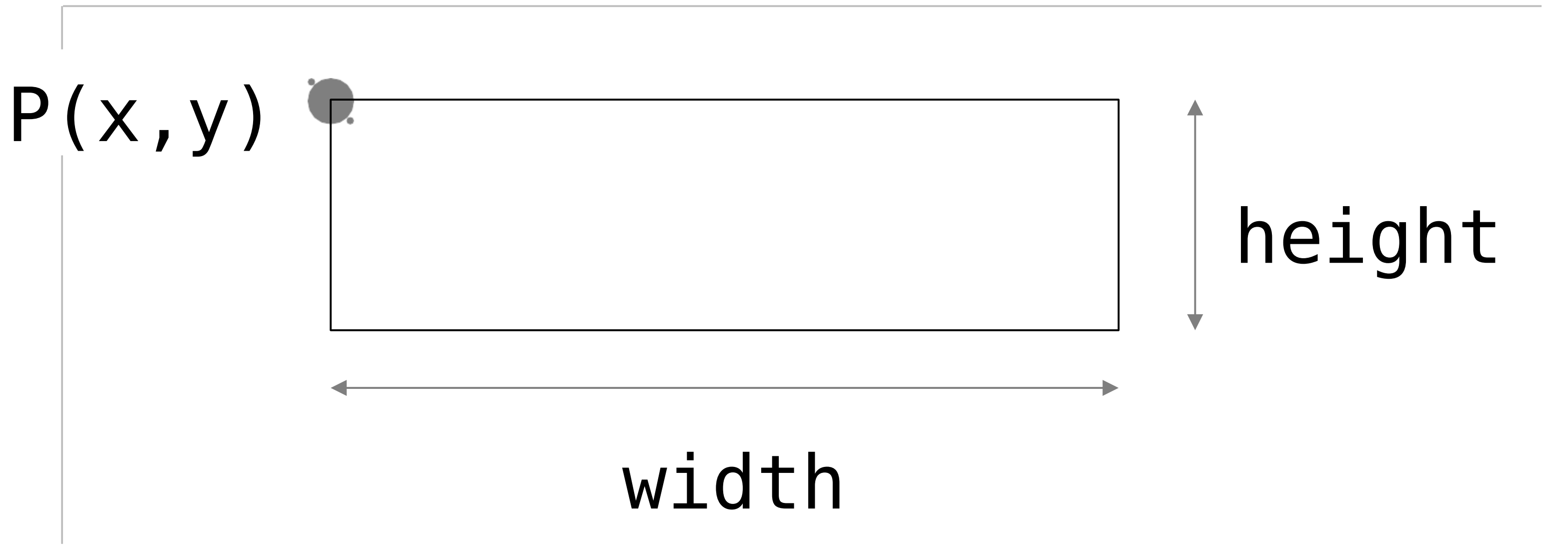
- Draw a line *segment* between two points `P1(x1,y1)`, `P2(x2,y2)`



drawRect()

```
public void drawRect(int x, int y,
                            int width, int height)
```

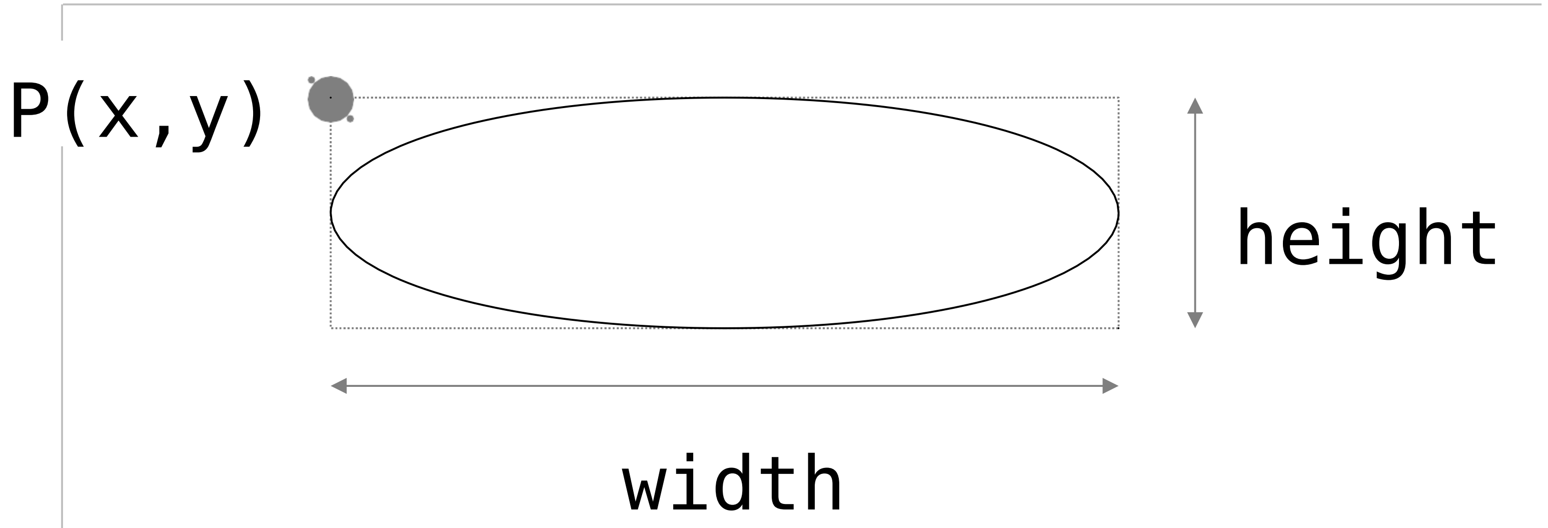
- Draw an (unfilled) rectangle whose top-left corner is `P(x,y)` and whose dimension is `width`, `height`



drawOval()

```
public void drawOval(int x, int y,
                                   int width, int height)
```

- Draw an (unfilled) oval bounded by an (invisible) rectangle: top-left corner = `P(x,y)` & dimension =(`width`, `height`)



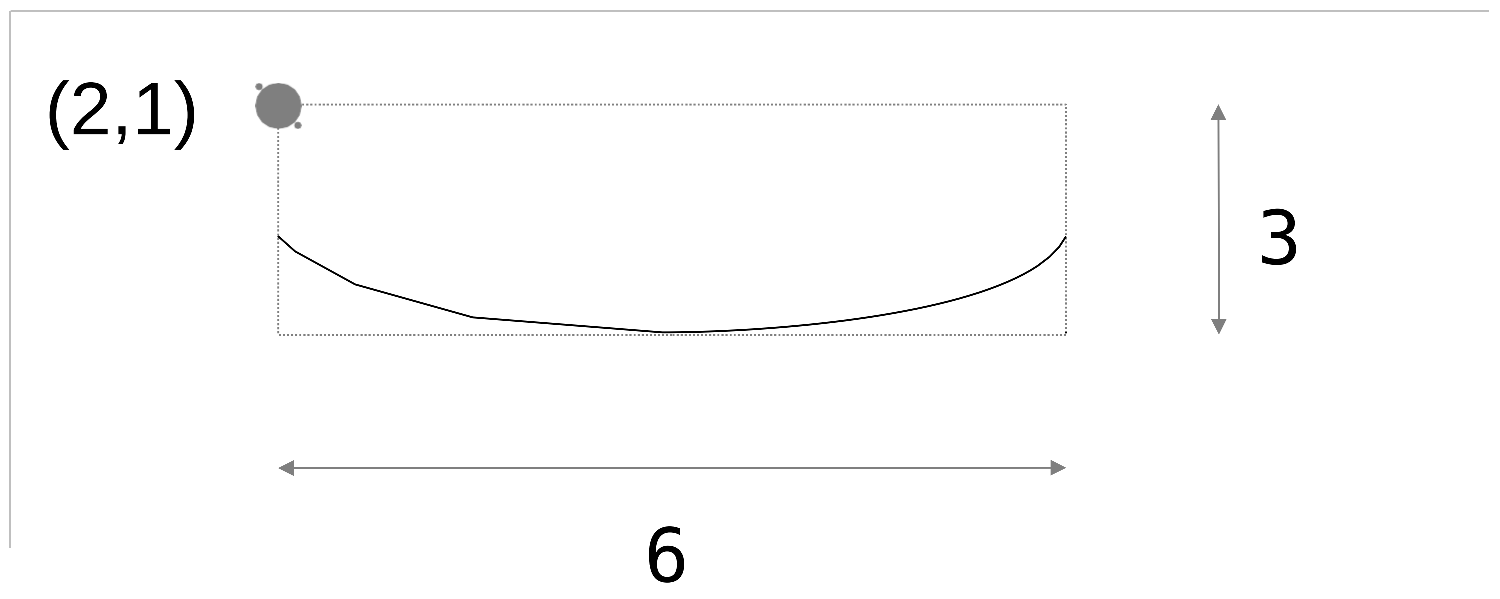
drawArc()

```
public void drawArc(int x, int y,
                    int width, int height,
                    int startAngle, int sweepAngle)
```

- Draw an (unfilled) arc of an oval
    - bounded by an (invisible) rectangle `<P(x,y), width, height>`,
    - start position is at `startAngle` and
    - end position is at the angle `startAngle + sweepAngle`

drawArc() method example

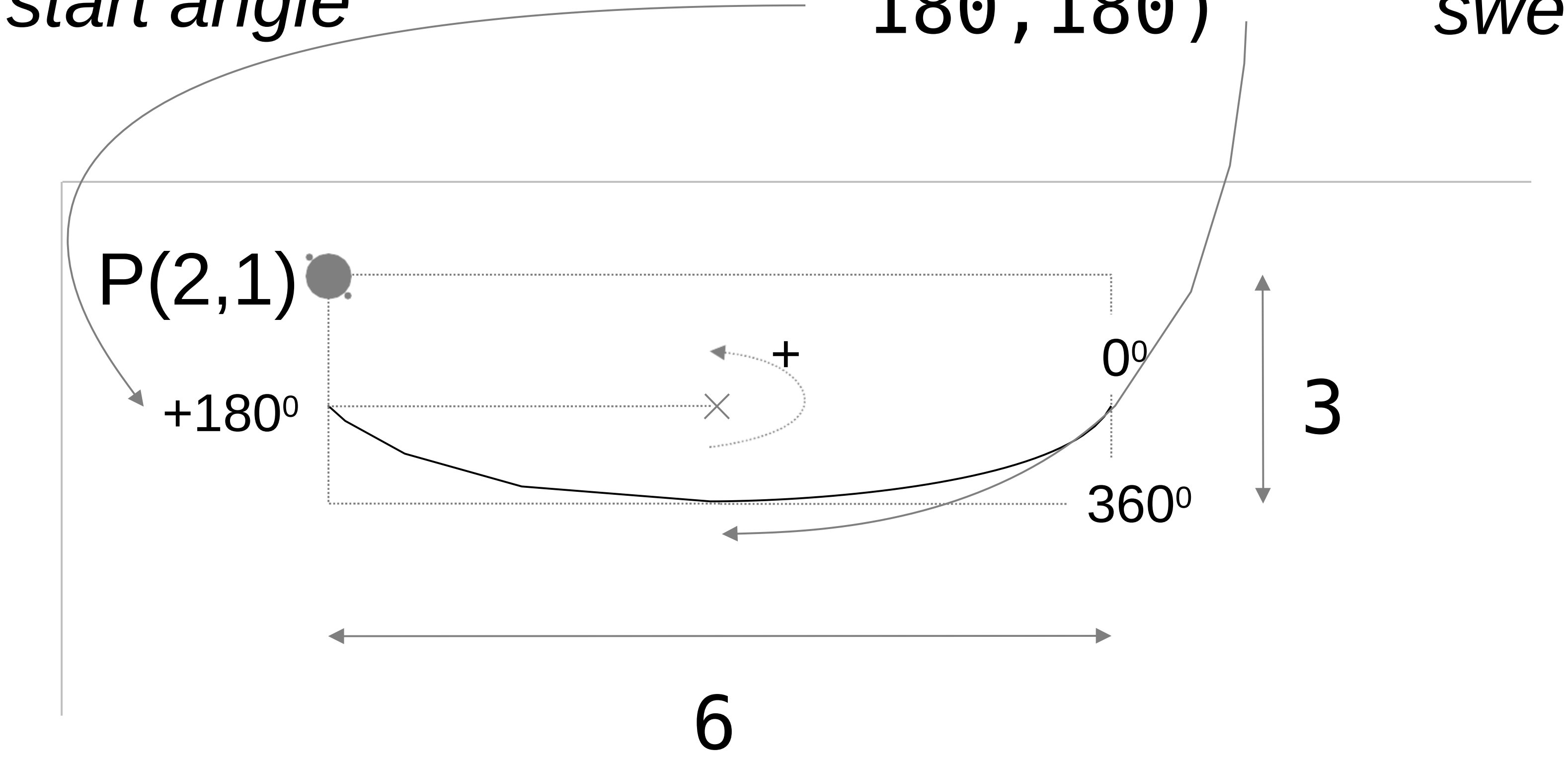
```
public void drawArc(2, 1,
                    6, 3
                    180,180)
```



drawArc() method example (2)

```
public void drawArc(2, 1,
                          6, 3
                          180,180)
```

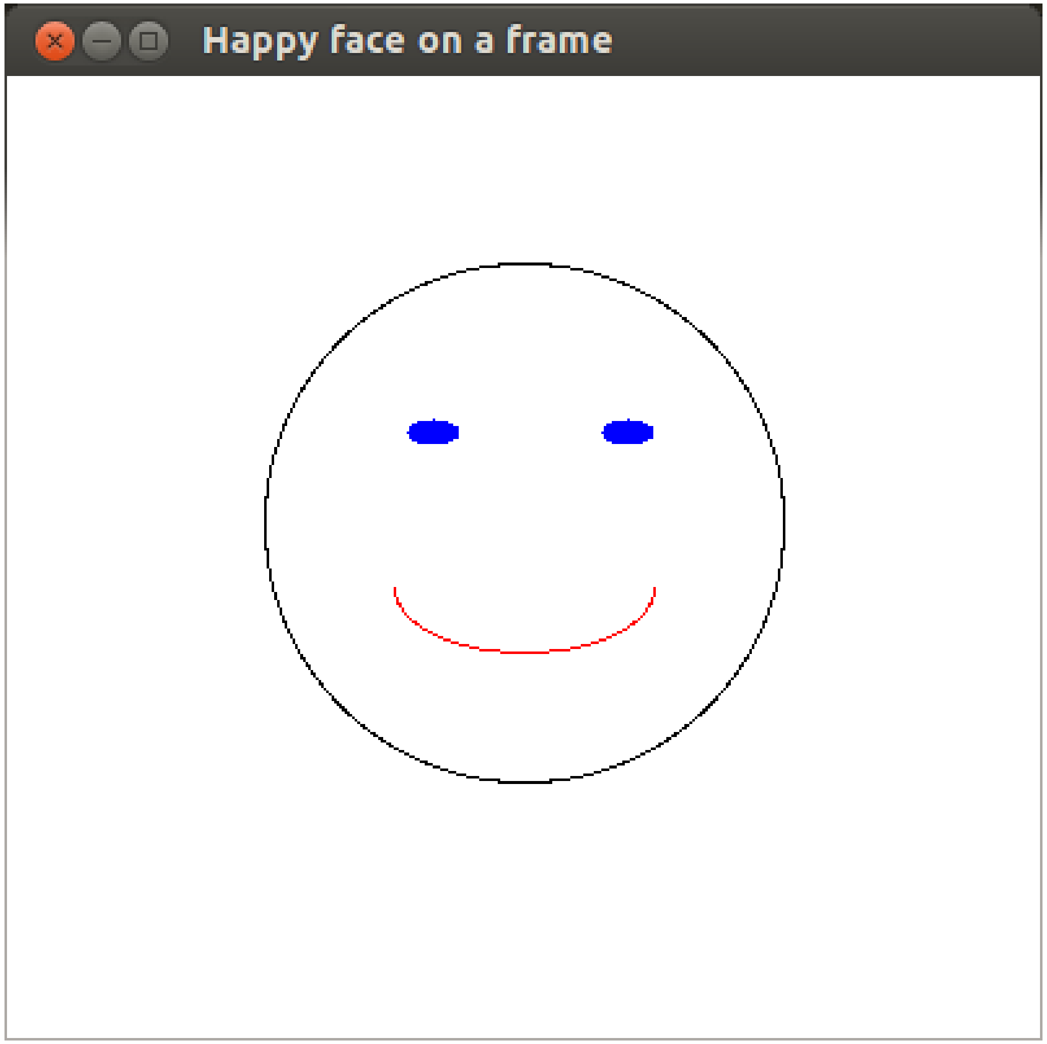
start angle *sweep angle*



DEMO

Happy face

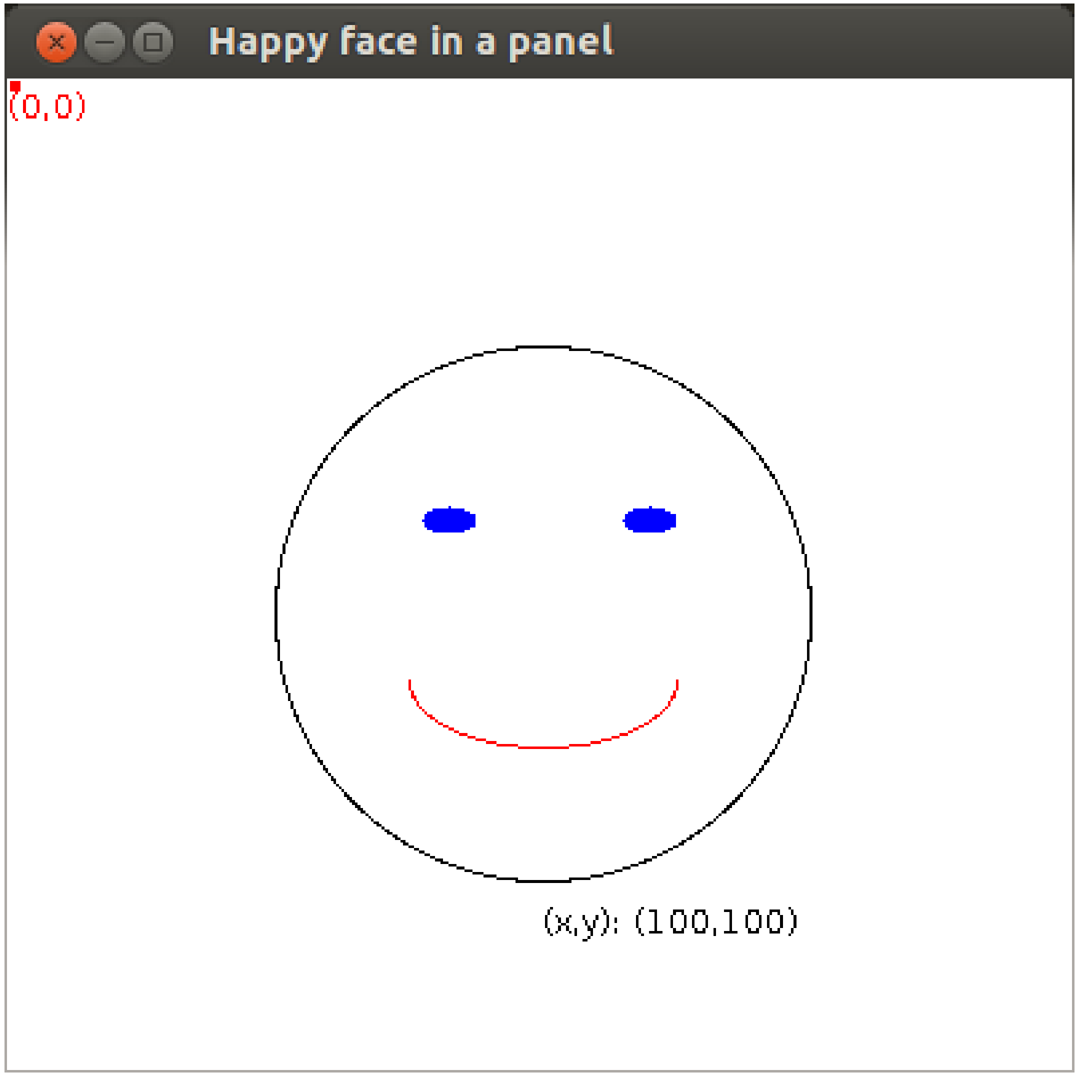
`lect09.drawing.HappyFaceColorFrame`



DEMO

Happy face panel

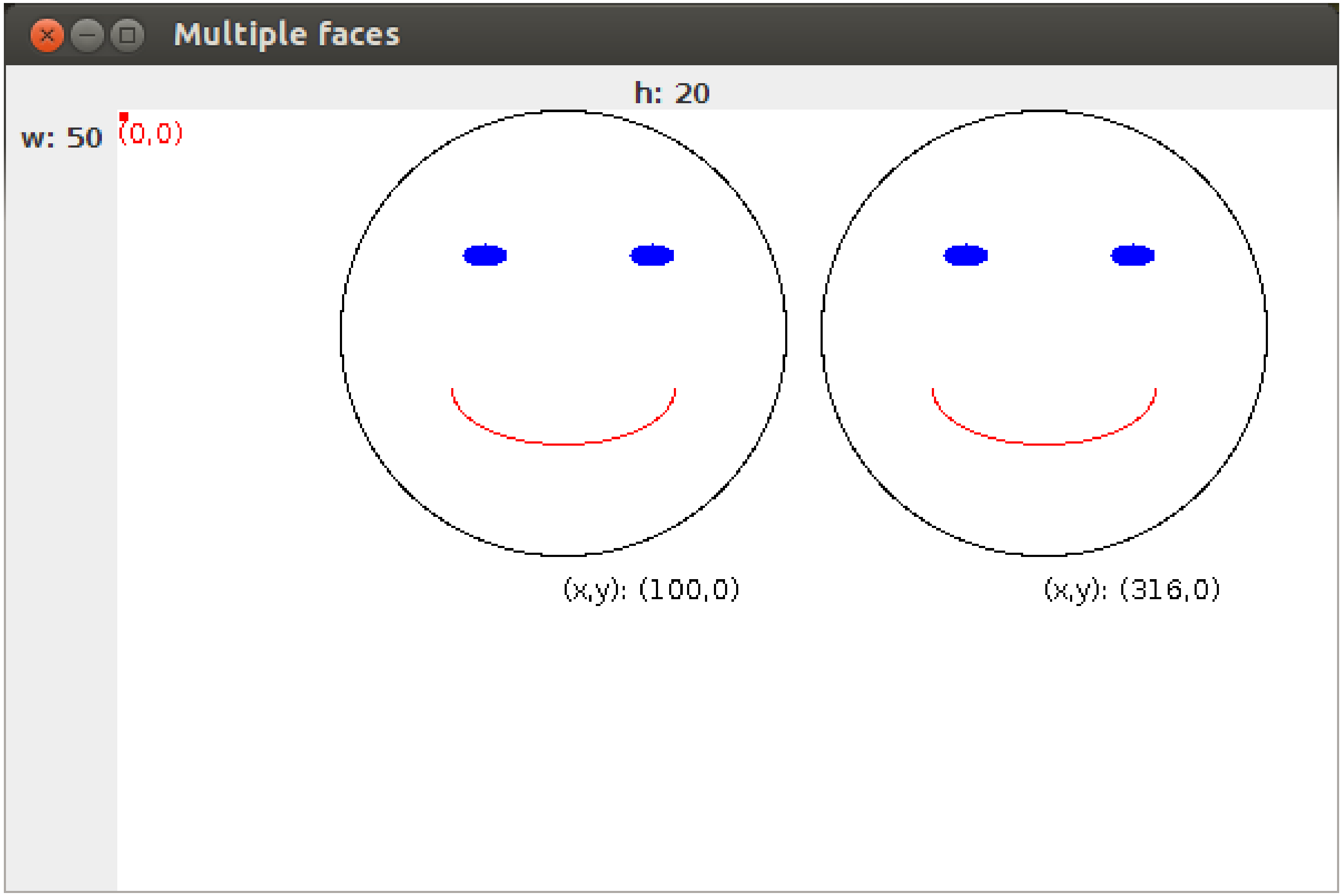
`lect09.drawing.HappyFaceColorPanel`



DEMO

Multiple happy faces

`lect09.drawing.MultipleHappyFaces`



References

Savitch W., Absolute Java, 6th, Pearson, 2015

- Chapter 17,18

Oracle, The Java Tutorial, Oracle, 2011, http://docs.oracle.com/javase/tutorial

-Lesson: Creating a GUI With JFC/Swing, Using Swing Components

- Trail: 2D Graphics