

MSc/ICY Software Workshop Graphics

Manfred Kerber www.cs.bham.ac.uk/~mmk

23 November 2016

JFrame

In the following we will look at packages called AWT Graphics and Swing for the graphical display. In order to display objects graphically in a subclass of JPanel,

```
public class NewClass extends JPanel,  
we always first create a JFrame of a particular size by  
JFrame frame = new JFrame()
```

We can set the size and the title of the frame by

```
final int FRAME_WIDTH = 600;    600 pixels  
final int FRAME_HEIGHT = 400;   400 pixels  
frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);  
frame.setTitle("Example frame");
```

Usually we want the application to terminate when the frame is closed and want it to be visible:

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
frame.setVisible(true);
```

We add to a frame a so-called JPanel.

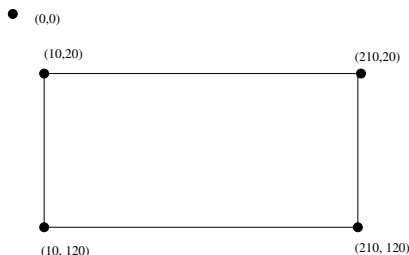
```
JPanel panel = new JPanel();
```

On the panel we draw objects by overriding the method

```
public void paintComponent(Graphics g) e.g.
```

```
@Override
```

```
public void paintComponent(Graphics g) {  
    g.drawRectangle(10,20,200,100);  
}
```



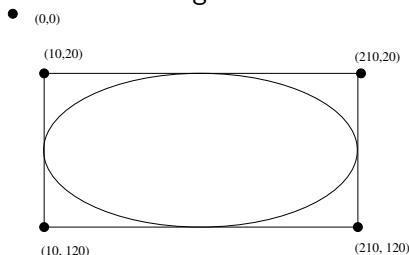
What to Add to a Panel?

Note that the dimensions are given in pixels from the left-right corner of the frame.

We can draw:

- outline of a Rectangle `drawRect(x, y, width, height)`
- filled Rectangle `fillRect(x, y, width, height)`
- outline of an Oval `drawOval(x, y, width, height)`
- filled Oval `fillOval(x, y, width, height)`

Note that the x and y in case of an oval (ellipse) give the left uppermost point of the bounding box of the oval (not the oval itself).



What to Add to a Panel? (Cont'd)

We can add a line from (x_0, y_0) to (x_1, y_1) by adding the line to the body of `paintComponent`, that is, by

```
@Override  
public void paintComponent(Graphics g) {  
    g.drawLine(x0, y0, x1, y1);  
}
```

What to Add to a Panel? (Cont'd)

By setting a font by something like

`setFont(new Font("Dialog",1,12))` we can add some text by:
`g.drawString("Some text added here",10,10)` at position (10,10).

We can draw arbitrary polygons by specifying the x- and y-values of the vertices by two arrays:

```
int[] xPoints = new int[vertices];  
int[] yPoints = new int[vertices];  
g.drawPolygon(xPoints, yPoints, vertices);
```

`vertices` is the number of vertices of the Polygon. We can also create a Polygon object by

```
Polygon pol = new Polygon(xPoints, yPoints, vertices)
```

Likewise, `drawPolyline` (does not draw line back to the start).

Adding an image

We can add an image (in `paintComponent(Graphics g)`) by `g.drawImage(loadImage(image), xPos, yPos, null)` with arguments: an image, the xPosition, the yPosition, and an `ImageObserver` not used in our context.

Some colours are predefined by constants such as BLACK, RED and so on. They can also be defined by `Color(r,g,b)` where `r,g,b` are values between 0 and 255. `r=red`, `g=green`, and `b=blue`. `0,0,0` stands for black, `255,0,0` for red, `0,255,0` for green, and `0,0,255` blue with other values in between.

BLACK: `Color(0,0,0)`

RED: `Color(255,0,0)`

GREEN: `Color(0,255,0)`

BLUE: `Color(0,0,255)`

ORANGE: `Color(255,200,0)`

PINK: `Color(255,175,175)`

CYAN: `Color(0,255,255)`

MAGENTA: `Color(255,0,255)`

YELLOW: `Color(255,255,0)`

WHITE: `Color(255,255,255)`

LIGHT_GRAY: `Color(192,192,192)`

GRAY: `Color(128,128,128)`

DARK_GRAY: `Color(64,64,64)`

`Color(164,255,64)`

Many more Methods

E.g., in `public void paintComponent(Graphics g)` use `g.copyArea(0,0,100,100,300,300)` [to copy the area in the rectangle from (0,0) to (100,100) to one starting at (300,300)]

For more methods, see e.g.

<http://docs.oracle.com/javase/8/docs/api/java/awt/Graphics.html>

See, also examples.