MSc/ICY Software Workshop
Graphics

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

Tuesday 19 November 2019

Overview

1. Pocket calculator computations, base types, simple strings, variables, static methods, JavaDoc
   Wed/Thu/Fri: 1st Lab Lecture (login, editor, javac, javadoc)
2. Classes, objects, methods, JUnit tests
   Wed/Thu/Fri: 2nd Lab Lecture (Eclipse)
3. Conditionals, ‘for’ Loops, arrays, ArrayList
4. Exceptions, I/O (Input/Output)
5. Functions, interfaces
6. Sub-classes, inheritance, abstract classes
7. Inheritance (Cont’d), packages
8. **Graphics**
9. Revision
10. Graphical User Interfaces
11. Graphical User Interfaces (Cont’d)

Changes possible

JavaFX

In the following we will introduce JavaFX for the graphical display (JavaFX replaces Swing the previous graphic package). In order to display objects graphically we generate a subclass of `Application`,
`public class DrawLine extends Application`.
We also have to import classes, here by

```
import javafx.application.Application;
```

Furthermore, Eclipse has to be set up properly to find the modules.
On the command line we can compile files by adding the modules:

```
javac --module-path PathToJavaFXLibrary
      --add-modules=javafx.controls
      --add-modules=javafx.swing MyClass.java
```

Likewise for the commands `java` and `javadoc` need to be extended. You may want to use aliases.

JavaFX (Cont'd)

The class `public class DrawLine extends Application` will contain the window, called `stage`, which contains all the objects displayed. It is an argument of the `start` method.
The `stage` contains a `scene` and a `scene` a scene graph of type `Group`.
We can set the size and the title of the scene by

```
Group root = new Group();
Scene scene = new Scene(root, 600, 300);
```

A Minimal Example

```
public class Minimal extends Application{
    //A red empty window of 600x300 pixels with title.
    @Override
    public void start(Stage stage) throws Exception {
        Group root = new Group();
        Scene scene = new Scene(root, 600, 300);
        stage.setTitle("Minimal");
        stage.setScene(scene);
        scene.setFill(Color.RED);
        stage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Adding a Line

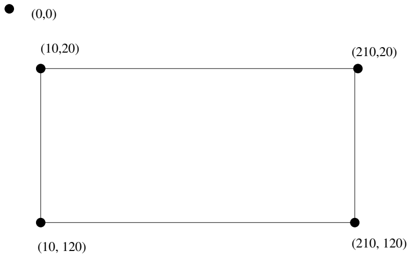
A straight line with the two end points (x1,y1) and (x2,y2) is created with the constructor `Line(x1,y1, x2,y2)` and can be added to the group.

```
@Override
public void start(Stage stage) throws Exception {
    /* Create a line object with end points (100,150) and (500,180) in
     * a coordinate system where the values are measured in pixels.
     * The 1st value (x-value) is how many pixels to the right, the
     * 2nd value (y-value) how many pixels to go down from the left
     * upper corner.
     */
    Line line = new Line(100,150, 500,180);
    //Create a Group (scene graph) with the line as member
    Group root = new Group(line);
    // The scene consists of just one group.
    Scene scene = new Scene(root, 600, 300);
    stage.setTitle("Line");
    stage.setScene(scene);
    stage.show();
}
```

Adding a Rectangle

- Create a Rectangle object `Rectangle rectangle = new Rectangle(x, y, width, height)`
- `rectangle.setFill(Color.BLUE);`
  (Colour is BLACK if not otherwise specified.)

Note that the `x` and `y` give the coordinate of the left upper point of the rectangle. E.g., `Rectangle(10, 20, 200, 100)`



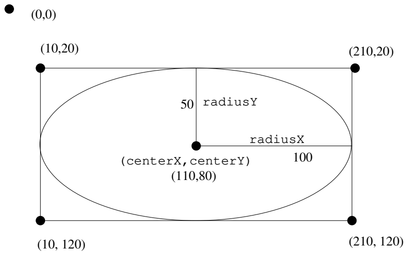
Adding a Circle and an Ellipse

- Create a Circle and Ellipse object:

```
Circle circle = new Circle(centerX,centerY,radius)
Ellipse oval =
        new Ellipse(centerX,centerY,radiusX,radiusY);
```

`centerX` and `centerY` give the coordinates of centres of the circle and the ellipse.

E.g., `Ellipse(110, 80, 100, 50)`



Adding a Polyline and a Polygon

- Create a Polyline object:

```
Polyline polyline =
     new Polyline(210,10, 10,210, 410,210);
```

Likewise

- Create a Polygon object:

```
Polygon polygon =
     new Polygon(210,10, 10,210, 410,210);
```

In a Polygon there is a line from last point to the first.

```
Polygon polygon = new Polygon(210,10, 10,210, 410,210);
// do not fill polygon by:
polygon.setFill(null);
// make borderlines visible
polygon.setStroke(Color.BLACK);
// Create a Group (scene graph) with the polygon
Group root = new Group(polygon);
```

Adding Text

```
Text text = new Text(100.0,150.0, "Hello World");

//Changing the font to "verdana" at a size of 70 pt
text.setFont(Font.font("verdana", 70));

/* FontWeight accepts nine values: BLACK,BOLD,EXTRA_BOLD,
 * EXTRA_LIGHT,MEDIUM, NORMAL, SEMI_BOLD, and THIN.
 * FontPosture two values: REGULAR and ITALIC.
 */
text.setFont(Font.font("verdana", FontWeight.BOLD,
                       FontPosture.ITALIC, 100));
//The text gets a horizontal line in the middle through it
text.setStrikethrough(true);

//The text is underlined.
text.setUnderline(true);
```

Using Colour

Some colours are predefined by constants such such as Color.BLACK, Color.RED and so on. They can also be defined by Color.rgb(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: rgb(0,0,0)
RED: rgb(255,0,0)
GREEN: rgb(0,255,0)
BLUE: rgb(0,0,255)
ORANGE: rgb(255,200,0)
PINK: rgb(255,175,175)
CYAN: rgb(0,255,255)
```

```
MAGENTA: rgb(255,0,255)
YELLOW: rgb(255,255,0)
WHITE: rgb(255,255,255)
LIGHT_GRAY: rgb(192,192,192)
GRAY: rgb(128,128,128)
DARK_GRAY: rgb(64,64,64)
SOME_COLOUR: rgb(164,255,64)
```

Adding an Image

Create an Image and add it as an ImageView to a Group.

```
private static Image image;
public void start(Stage stage) throws Exception {
    //Setting the image view
    ImageView imageView = new ImageView(image);
    imageView.setX(150);
    imageView.setY(100);
    Group root = new Group(imageView);
    ...
}
public static void main(String[] args) {
    //Initializing the image
    image = new Image("images/firstCar.jpg");
    //image = new Image("http://www.cs.bham.ac.uk/...");
    launch(args);
}
```

Animation

We show an example Animation with two regular polygons, one rotating, one shrinking and expanding.

```
public void start(Stage stage) throws Exception {
    RotateTransition rotateTr = new RotateTransition();
    rotateTr.setDuration(Duration.millis(10000));
    rotateTr.setByAngle(360);
    rotateTr.setCycleCount(5);
    rotateTr.setAutoReverse(false);
    rotateTr.setNode(polygons[0]);
    rotateTr.play();

    ScaleTransition scaleTr = new ScaleTransition();
    scaleTr.setDuration(Duration.millis(1000));
    scaleTr.setNode(polygons[1]);
    scaleTr.setByY(-0.5);
    scaleTr.setByX(-0.5);
    scaleTr.setCycleCount(50);
    scaleTr.setAutoReverse(true);
    scaleTr.play();
    ...
}
```

Much More

There is a lot of information available online, e.g., by Oracle:
https://docs.oracle.com/javafx/2/get_started/hello_world.htm

There are also online tutorials:
https://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm

https://www.tutorialspoint.com/javafx
The latter was used heavily in the preparation of the slides and the examples to this lecture.