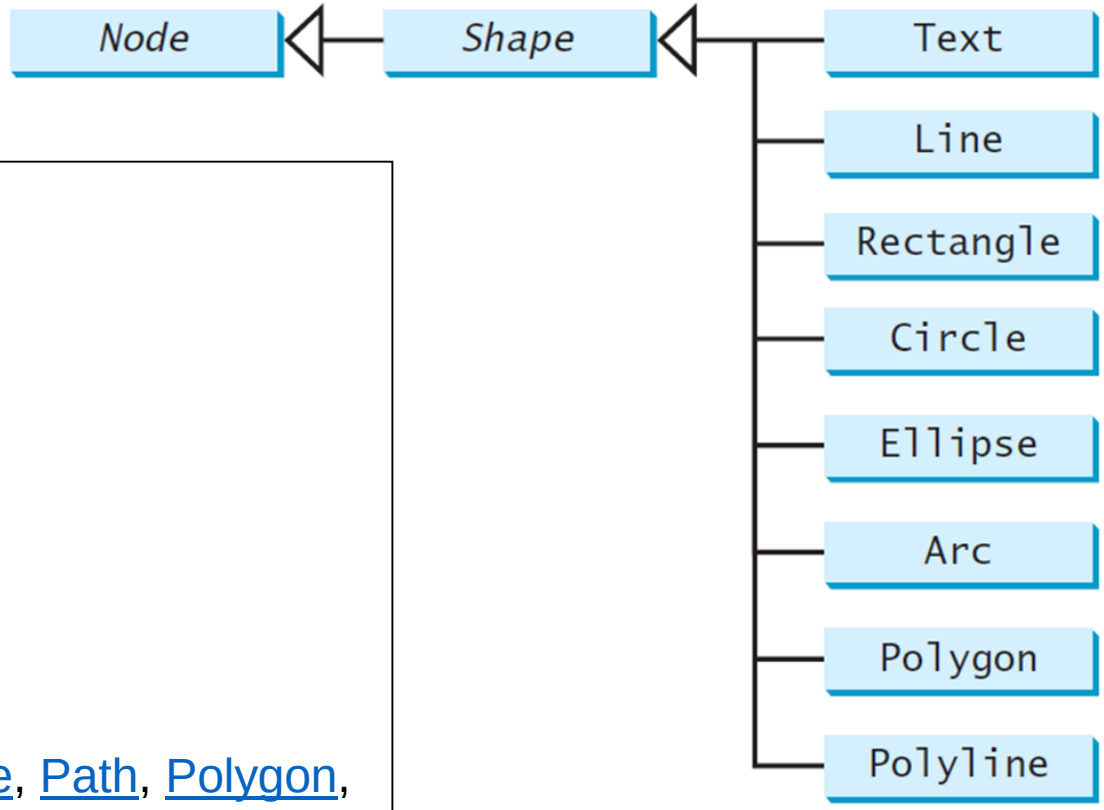
Shapes og Events

- Shapes: tekst, linjer og flater
- Case study: The ClockPane Class
- Events og Event Handlers
- Indre klasser, anonyme indre klasser, Lambda-uttrykk
- Case study: Lånekalkulator
- Muse-event
- Tastatur- (key-) event
- Lyttere

Shapes



Fra API'et:
Class Shape

java.lang.Object
    javafx.scene.Node
        javafx.scene.shape.Shape

All Implemented Interfaces:
    Styleable, EventTarget

Direct Known Subclasses:
    Arc, Circle, CubicCurve, Ellipse, Line, Path, Polygon, Polyline, QuadCurve, Rectangle, SVGPath, Text

Text

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.text.Text

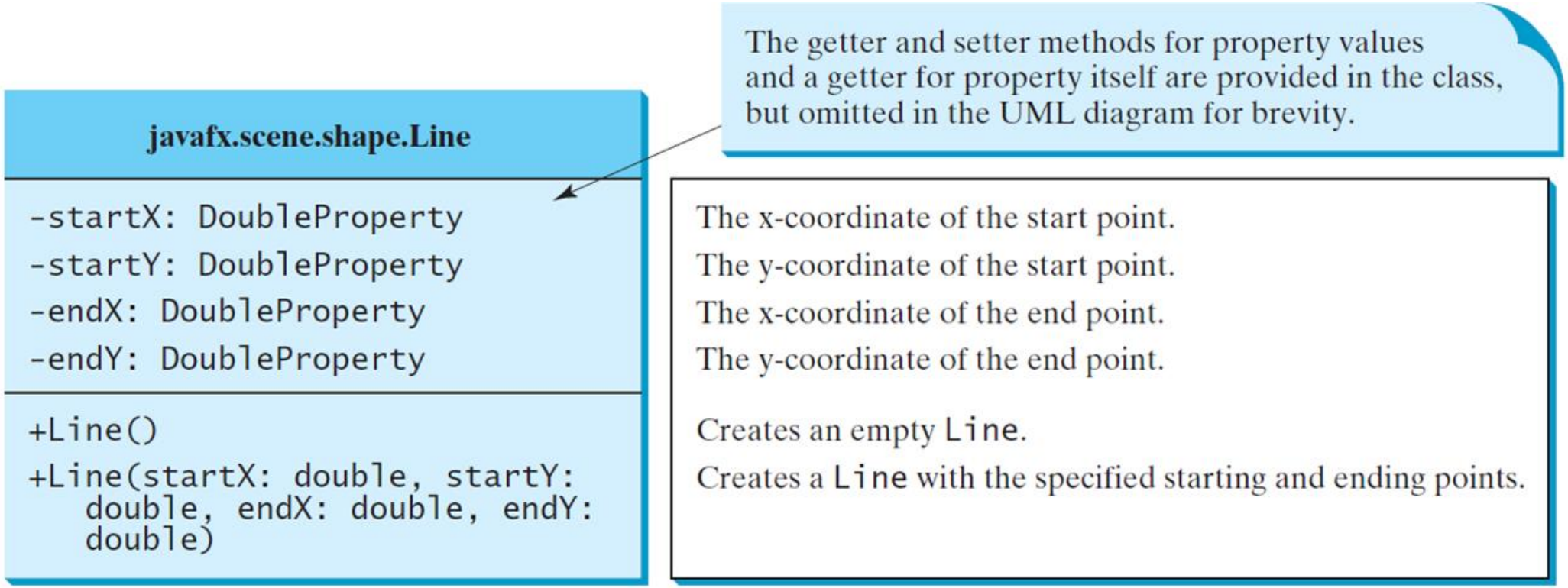
-text: StringProperty	Defines the text to be displayed.
-x: DoubleProperty	Defines the x-coordinate of text (default 0).
-y: DoubleProperty	Defines the y-coordinate of text (default 0).
-underline: BooleanProperty	Defines if each line has an underline below it (default `false`).
-strikethrough: BooleanProperty	Defines if each line has a line through it (default `false`).
-font: ObjectProperty<Font>	Defines the font for the text.
+Text()	Creates an empty Text.
+Text(text: String)	Creates a Text with the specified text.
+Text(x: double, y: double, text: String)	Creates a Text with the specified x-, y-coordinates and text.

ShowText

Husk koordinatsystemet:
(0,0) oppe til venstre
X mot høyre
Y nedover

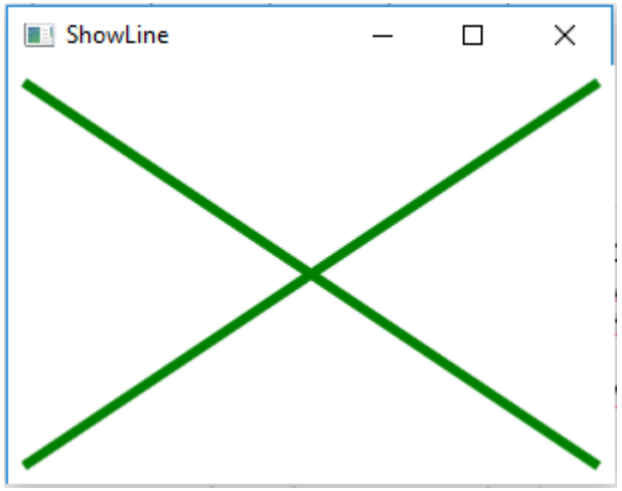
```
public class ShowText extends Application {
  @Override
  public void start(Stage primaryStage) {
    // Create a pane to hold the texts
    Pane pane = new Pane();
    pane.setPadding(new Insets(5, 5, 5, 5));
    Text text1 = new Text(20, 20, "Programming is fun");
    text1.setFont(Font.font("Courier", FontWeight.BOLD,
      FontPosture.ITALIC, 15));
    pane.getChildren().add(text1);

    Text text2 = new Text(60, 60, "Programming is fun\nDisplay text");
    pane.getChildren().add(text2);

    Text text3 = new Text(10, 100, "Programming is fun\nDisplay text");
    text3.setFill(Color.RED);
    text3.setUnderline(true);
    text3.setStrikethrough(true);
    pane.getChildren().add(text3);

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane);
    primaryStage.setTitle("ShowText");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
```

Line


javafx.scene.shape.Line
-startX: DoubleProperty
-startY: DoubleProperty
-endX: DoubleProperty
-endY: DoubleProperty
+Line()
+Line(startX: double, startY: double, endX: double, endY: double)
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.
The x-coordinate of the start point.
The y-coordinate of the start point.
The x-coordinate of the end point.
The y-coordinate of the end point.
Creates an empty Line.
Creates a Line with the specified starting and ending points.

ShowLine

```
public class ShowLine extends Application {
  @Override
  public void start(Stage primaryStage) {
    // Create a scene and place it in the stage
    Scene scene = new Scene(new LinePane(), 200, 200);
    primaryStage.setTitle("ShowLine");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
}
```




```
class LinePane extends Pane {
  public LinePane() {
    Line line1 = new Line(10, 10, 10, 10);
    line1.endXProperty().bind(widthProperty().subtract(10));
    line1.endYProperty().bind(heightProperty().subtract(10));
    line1.setStrokeWidth(5);
    line1.setStroke(Color.GREEN);
    getChildren().add(line1);

    Line line2 = new Line(10, 10, 10, 10);
    line2.startXProperty().bind(widthProperty().subtract(10));
    line2.endYProperty().bind(heightProperty().subtract(10));
    line2.setStrokeWidth(5);
    line2.setStroke(Color.GREEN);
    getChildren().add(line2);
  }
}
```

Rectangle

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.shape.Rectangle

```
-x: DoubleProperty
-y:DoubleProperty
-width: DoubleProperty
-height: DoubleProperty
-arcWidth: DoubleProperty

-arcHeight: DoubleProperty
```

```
+Rectangle()
+Rectanlge(x: double, y:
   double, width: double,
   height: double)
```

The x-coordinate of the upper-left corner of the rectangle (default 0).

The y-coordinate of the upper-left corner of the rectangle (default 0).

The width of the rectangle (default: 0).

The height of the rectangle (default: 0).

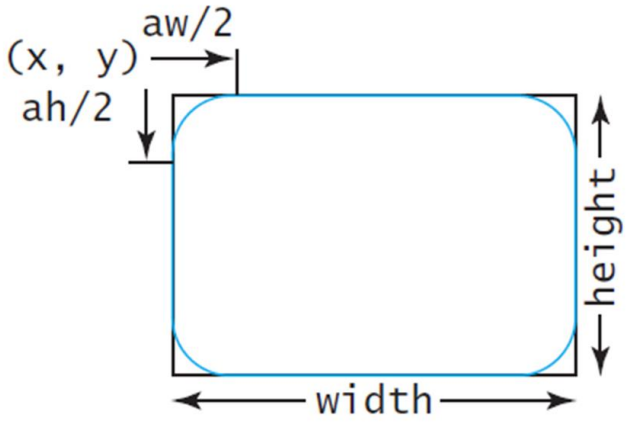
The `arcWidth` of the rectangle (default: 0). `arcWidth` is the horizontal diameter of the arcs at the corner (see Figure 14.31a).

The `arcHeight` of the rectangle (default: 0). `arcHeight` is the vertical diameter of the arcs at the corner (see Figure 14.31a).

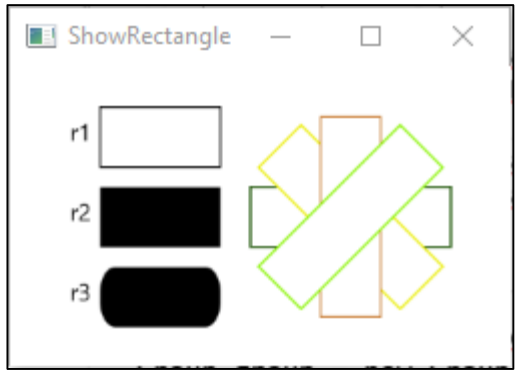
Creates an empty `Rectangle`.

Creates a `Rectangle` with the specified upper-left corner point, width, and height.

ShowRectangle



(a) Rectangle(x, y, w, h)




```
@Override
public void start(Stage primaryStage) {
  // Create rectangles
  Rectangle r1 = new Rectangle(25, 10, 60, 30); //x, y, b, h
  r1.setStroke(Color.BLACK);
  r1.setFill(Color.WHITE);
  Rectangle r2 = new Rectangle(25, 50, 60, 30);
  Rectangle r3 = new Rectangle(25, 90, 60, 30);
  r3.setArcWidth(15);
  r3.setArcHeight(25);

  // Create a group and add nodes to the group
  Group group = new Group();
  group.getChildren().addAll(new Text(10, 27, "r1"), r1,
    new Text(10, 67, "r2"), r2, new Text(10, 107, "r3"), r3);

  for (int i = 0; i < 4; i++) {
    Rectangle r = new Rectangle(100, 50, 100, 30);
    r.setRotate(i * 360 / 8);
    r.setStroke(Color.color(Math.random(), Math.random(),
      Math.random()));
    r.setFill(Color.WHITE);
    group.getChildren().add(r);
  }
  Scene scene = new Scene(new BorderPane(group), 250, 150);
  // setTitle, setScene og show ...
```

Circle

javafx.scene.shape.Circle

```
-centerX: DoubleProperty
-centerY: DoubleProperty
-radius: DoubleProperty
```

```
+Circle()
+Circle(x: double, y: double)
+Circle(x: double, y: double,
   radius: double)
```

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

The x-coordinate of the center of the circle (default 0).

The y-coordinate of the center of the circle (default 0).

The radius of the circle (default: 0).

Creates an empty `Circle`.

Creates a `Circle` with the specified center.

Creates a `Circle` with the specified center and radius.

Ellipse

javafx.scene.shape.Ellipse

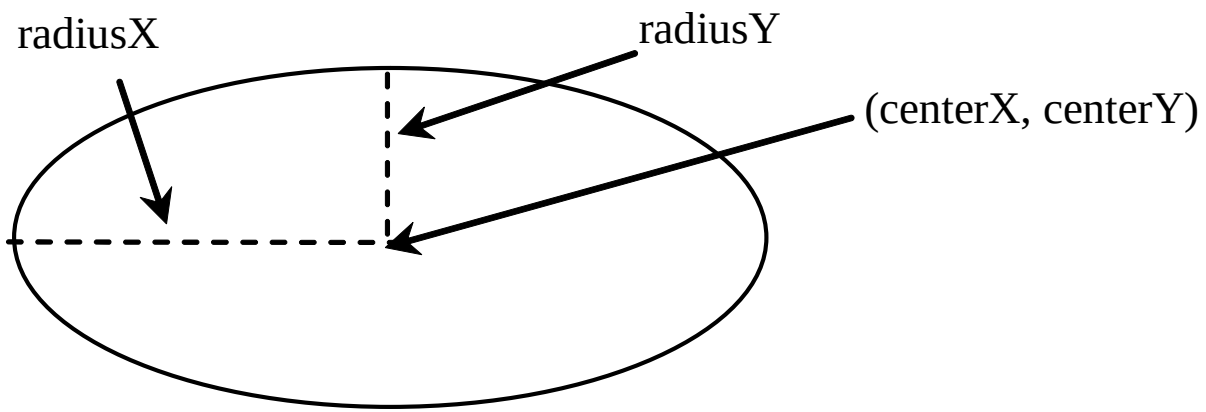
```
-centerX: DoubleProperty
-centerY: DoubleProperty
-radiusX: DoubleProperty
-radiusY: DoubleProperty
```

```
+Ellipse()
+Ellipse(x: double, y: double)
+Ellipse(x: double, y: double,
   radiusX: double, radiusY:
   double)
```

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

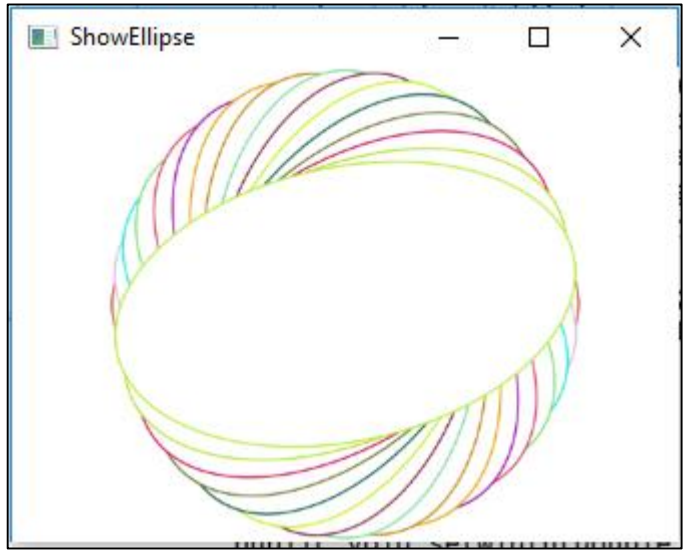
The x-coordinate of the center of the ellipse (default 0).
The y-coordinate of the center of the ellipse (default 0).
The horizontal radius of the ellipse (default: 0).
The vertical radius of the ellipse (default: 0).

Creates an empty `Ellipse`.
Creates an `Ellipse` with the specified center.
Creates an `Ellipse` with the specified center and radiuses.



ShowEllipse

```
@Override
  public void start(Stage primaryStage) {
    // Create a scene and place it in the stage
    Scene scene = new Scene(new MyEllipse(), 300, 200);
    primaryStage.setTitle("ShowEllipse");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
```




```
class MyEllipse extends Pane {
  private void paint() {
    getChildren().clear();
    for (int i = 0; i < 16; i++) {
      // Create an ellipse and add it to pane
      Ellipse e1 = new Ellipse(getWidth() / 2, getHeight() / 2,
        getWidth() / 2 - 50, getHeight() / 2 - 50);
      e1.setStroke(Color.color(Math.random(), Math.random(),
        Math.random()));
      e1.setFill(Color.WHITE);
      e1.setRotate(i * 180 / 16);
      getChildren().add(e1);
    }
  }

  @Override
  public void setWidth(double width) {
    super.setWidth(width);
    paint();
  }

  @Override
  public void setHeight(double height) {
    super.setHeight(height);
    paint();
  }
}
```

Arc – buelinje/kakestykke

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

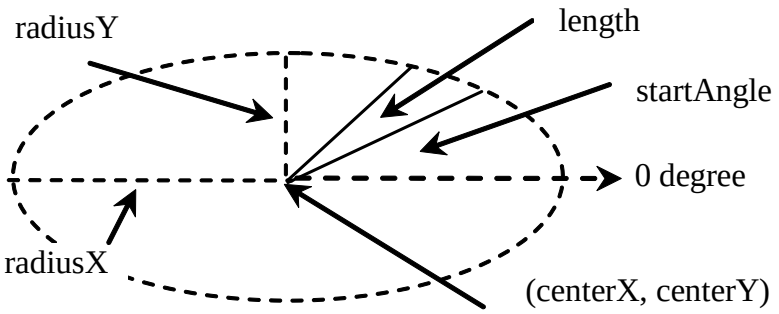
javafx.scene.shape.Arc

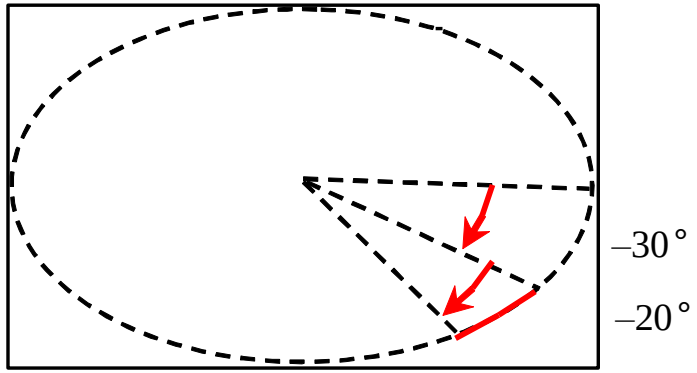
```
-centerX: DoubleProperty
-centerY: DoubleProperty
-radiusX: DoubleProperty
-radiusY: DoubleProperty
-startAngle: DoubleProperty
-length: DoubleProperty
-type: ObjectProperty<ArcType>
```

```
+Arc()
+Arc(x: double, y: double,
   radiusX: double, radiusY:
   double, startAngle: double,
   length: double)
```

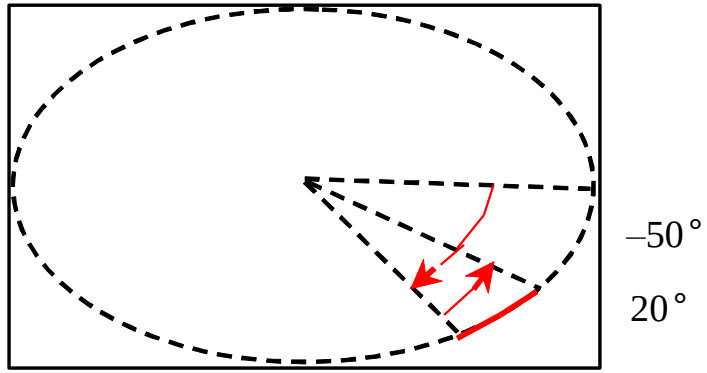
The x-coordinate of the center of the ellipse (default 0).	
The y-coordinate of the center of the ellipse (default 0).	
The horizontal radius of the ellipse (default: 0).	
The vertical radius of the ellipse (default: 0).	
The start angle of the arc in degrees.	
The angular extent of the arc in degrees.	
The closure type of the arc (`ArcType.OPEN`, `ArcType.CHORD`, `ArcType.ROUND`).	
Creates an empty `Arc`.	
Creates an `Arc` with the specified arguments.	

Arc eksempel





(a) Negative starting angle –30° and negative spanning angle –20°



(b) Negative starting angle –50° and positive spanning angle 20°

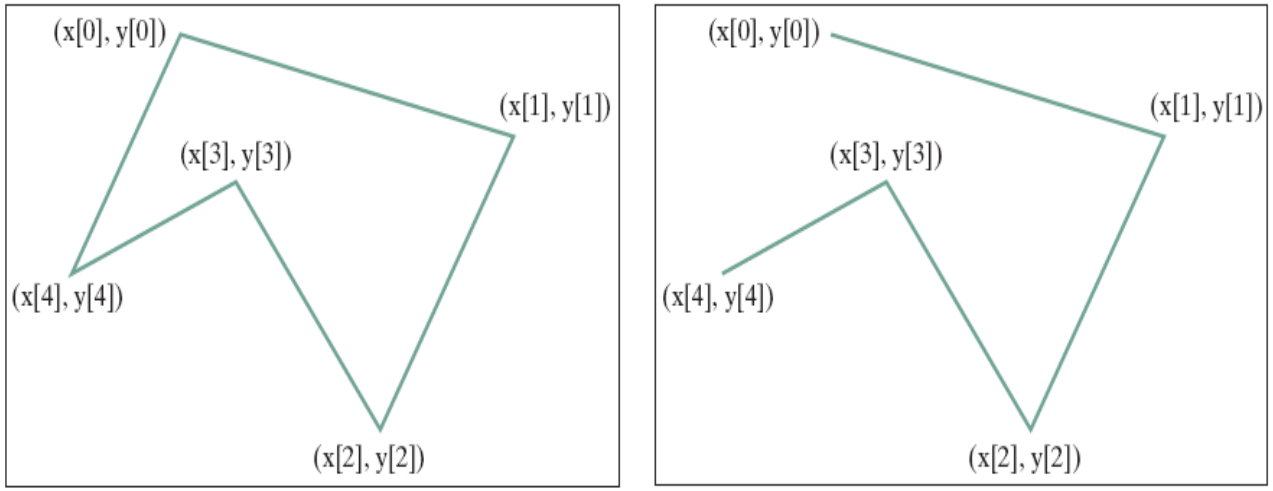
ShowArc

```
@Override
  public void start(Stage primaryStage) {
    // Create an arc
    Arc arc1 = new Arc(150, 100, 80, 80, 30, 35);
    arc1.setFill(Color.RED); // Set fill color
    arc1.setType(ArcType.ROUND); // Set arc type

    Arc arc2 = new Arc(150, 100, 80, 80, 30 + 90, 35);
    arc2.setFill(Color.WHITE);
    arc2.setType(ArcType.OPEN);
    arc2.setStroke(Color.BLACK);

    Arc arc3 = new Arc(150, 100, 80, 80, 30 + 180, 35);
    arc3.setFill(Color.WHITE);
    arc3.setType(ArcType.CHORD);
    arc3.setStroke(Color.BLACK);

    Arc arc4 = new Arc(150, 100, 80, 80, 30 + 270, 35);
    arc4.setFill(Color.GREEN);
    arc4.setType(ArcType.CHORD);
    arc4.setStroke(Color.BLACK);
```

```
    // Create a group and add nodes to the group
    Group group = new Group();
    group.getChildren().addAll(
      new Text(210, 40, "arc1: round"),
      arc1, new Text(20, 40, "arc2: open"), arc2,
      new Text(20, 170, "arc3: chord"), arc3,
      new Text(210, 170, "arc4: chord"), arc4);

    // Create a scene and place it in the stage
    Scene scene = new Scene(new BorderPane(group),
      300, 200);
    primaryStage.setTitle("ShowArc");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
```

The JavaFX Group component is a container component which applies no special layout to its children. All child components (nodes) are positioned at (0,0). A JavaFX Group component is typically used to apply some effect or transformation to a set of controls as a whole - as a group.

Polygon & Polyline



(a) Polygon (b) Polyline

javafx.scene.shape.Polygon
`+Polygon()`
`+Polygon(double... points)`
`+getPoints(): ObservableList<Double>`

The `getter` and `setter` methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

Creates an empty polygon.

Creates a polygon with the given points.

Returns a list of double values as x- and y-coordinates of the points.

Case study: The ClockPane

- I dette eksempelet utvikles et panel som kan vise klokkeslett i form av ei klokke med visere. I siste del av kapittel 15 – animasjon – ser boka på hvordan vi skal få klokka til å gå. Animasjon er ikke pensum i høst, derfor legger vi ikke vekt på dette eksempelet.
- Men les gjerne på egen hånd!

Kapittel 15: Events

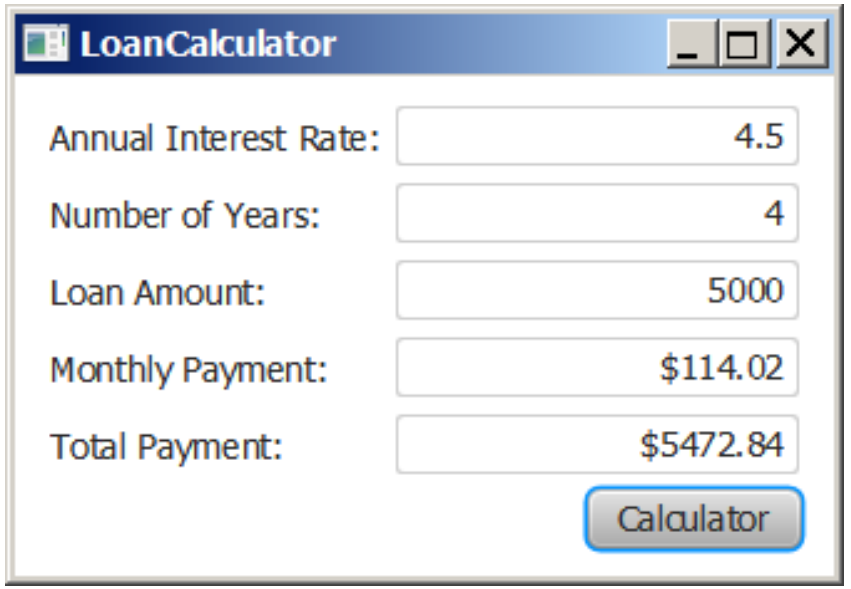
Vi ønsker at det grafiske brukergrensesnittet skal bli responsivt, reagere på brukeraksjoner.

Eksempel: lånekalkulator. Bruker oppgir:

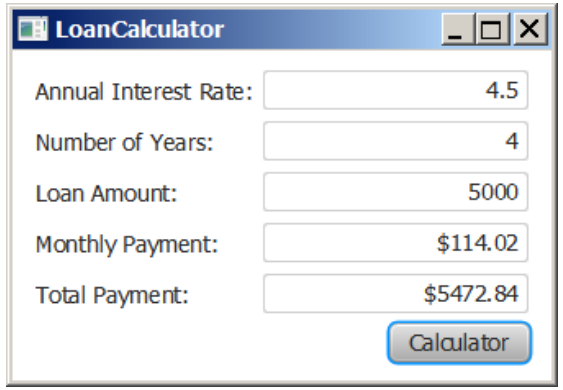
- Lånebeløp
- Rente
- Antall år

Trykk på «Calculator» så får du månedlig beløp og totalbeløp ut.
For å få til dette må vi behandle eventer/begivenheter.

Ved event-drevet programmering er det typisk brukeren som styrer rekkefølgen av begivenheter gjennom eventene.

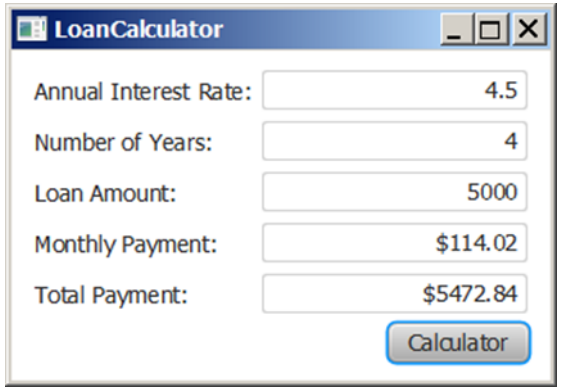


Lånekalkulator I




```
public class LoanCalculator extends Application {
  private TextField tfAnnualInterestRate = new TextField();
  private TextField tfNumberOfYears = new TextField();
  private TextField tfLoanAmount = new TextField();
  private TextField tfMonthlyPayment = new TextField();
  private TextField tfTotalPayment = new TextField();
  private Button btCalculate = new Button("Calculate");


  @Override
  public void start(Stage primaryStage) {
    // Create UI
    GridPane gridPane = new GridPane();
    gridPane.setHgap(5);
    gridPane.setVgap(5);
    gridPane.add(new Label("Annual Interest Rate:"), 0, 0);
    gridPane.add(tfAnnualInterestRate, 1, 0);
    gridPane.add(new Label("Number of Years:"), 0, 1);
    gridPane.add(tfNumberOfYears, 1, 1);
    gridPane.add(new Label("Loan Amount:"), 0, 2);
    gridPane.add(tfLoanAmount, 1, 2);
    gridPane.add(new Label("Monthly Payment:"), 0, 3);
    gridPane.add(tfMonthlyPayment, 1, 3);
    gridPane.add(new Label("Total Payment:"), 0, 4);
    gridPane.add(tfTotalPayment, 1, 4);
    gridPane.add(btCalculate, 1, 5);
```

Lånekalkulator II




```
  // Set properties for UI
  gridPane.setAlignment(Pos.CENTER);
  tfAnnualInterestRate.setAlignment(Pos.BOTTOM_RIGHT);
  tfNumberOfYears.setAlignment(Pos.BOTTOM_RIGHT);
  tfLoanAmount.setAlignment(Pos.BOTTOM_RIGHT);
  tfMonthlyPayment.setAlignment(Pos.BOTTOM_RIGHT);
  tfTotalPayment.setAlignment(Pos.BOTTOM_RIGHT);
  tfMonthlyPayment.setEditable(false);
  tfTotalPayment.setEditable(false);
  GridPane.setHalignment(btCalculate, HPos.RIGHT);

  // Create a scene and place it in the stage
  Scene scene = new Scene(gridPane, 400, 250);
  primaryStage.setTitle("LoanCalculator");
  primaryStage.setScene(scene);
  primaryStage.show();
}  // end start
```

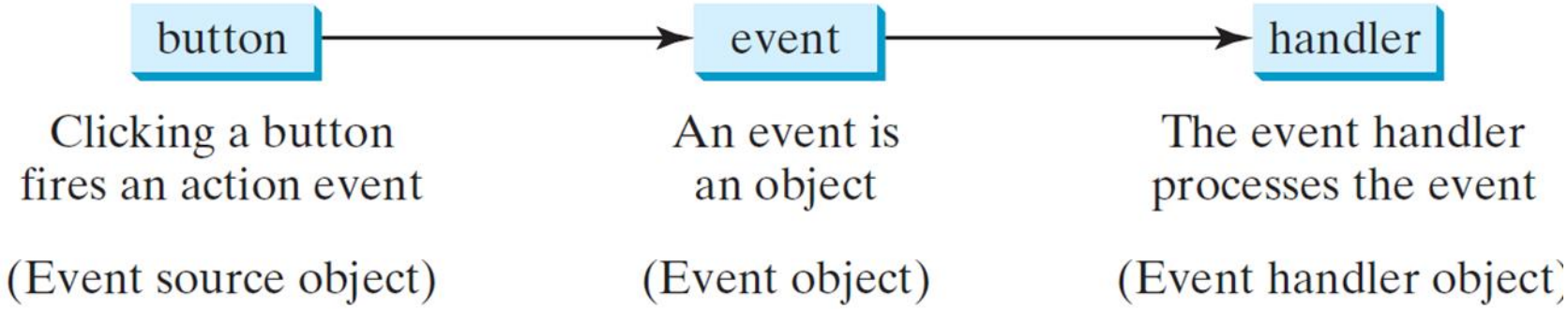
GUI’et *ser* pent ut, men det er dødt! Vi må gi det liv.

Event-handler

Tre objekter er involvert i prosessen:

1. Noe skjer i GUI'et, f.eks. at bruker trykker på en knapp («Event source object»)
2. Et event-objekt opprettes, og sendes til -
3. Event handler objekt som skal respondere på eventet – få noe til å skje

Event source og *event* er oftest fra biblioteket, *event handler* skriver vi selv!

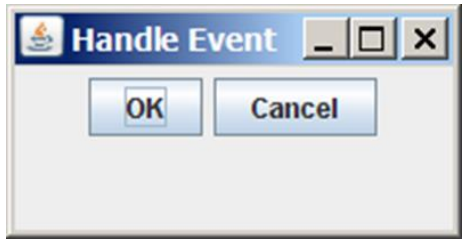


For at systemet skal greie å sende event-objektet til programmet vårt,
må programmet vårt oppføre seg i henhold til en kontrakt – interface!

class HandleEvent

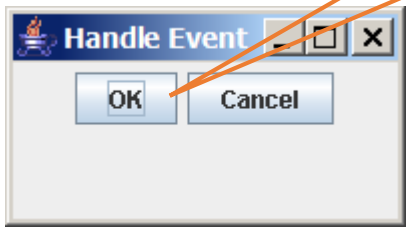
```
public class HandleEvent extends Application {
  @Override
  public void start(Stage primaryStage) {
    // Create a pane and set its properties
    HBox pane = new HBox(10);
    pane.setAlignment(Pos.CENTER);
    Button btOK = new Button("OK");
    Button btCancel = new Button("Cancel");
    OKHandlerClass handler1 = new OKHandlerClass();
    btOK.setOnAction(handler1);
    CancelHandlerClass handler2 =
      new CancelHandlerClass();
    btCancel.setOnAction(handler2);
    pane.getChildren().addAll(btOK, btCancel);

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane);
    primaryStage.setTitle("HandleEvent");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
```

```
class OKHandlerClass implements
  EventHandler<ActionEvent> {
  @Override
  public void handle(ActionEvent e) {
    System.out.println("OK button clicked");
  }
}

class CancelHandlerClass implements
EventHandler<ActionEvent> {
  @Override
  public void handle(ActionEvent e) {
    System.out.println("Cancel button clicked");
  }
}
```



Følg utførelsen: 1 – 2 – 3

```
public class HandleEvent extends Application {
  public void start(Stage primaryStage) {
    …
    OKHandlerClass handler1 = new OKHandlerClass();
    btOK.setOnAction(handler1);
    CancelHandlerClass handler2 = new CancelHandlerClass();
    btCancel.setOnAction(handler2);
    …
    primaryStage.show(); // Display the stage
  }
}

class OKHandlerClass implements
EventHandler<ActionEvent> {
  @Override
  public void handle(ActionEvent e) {
    System.out.println("OK button clicked");
  }
}
```



1. main-metoden kaller launch, som new'er HandleEvent applikasjonen og kaller start. show sørger for at vinduet vises.

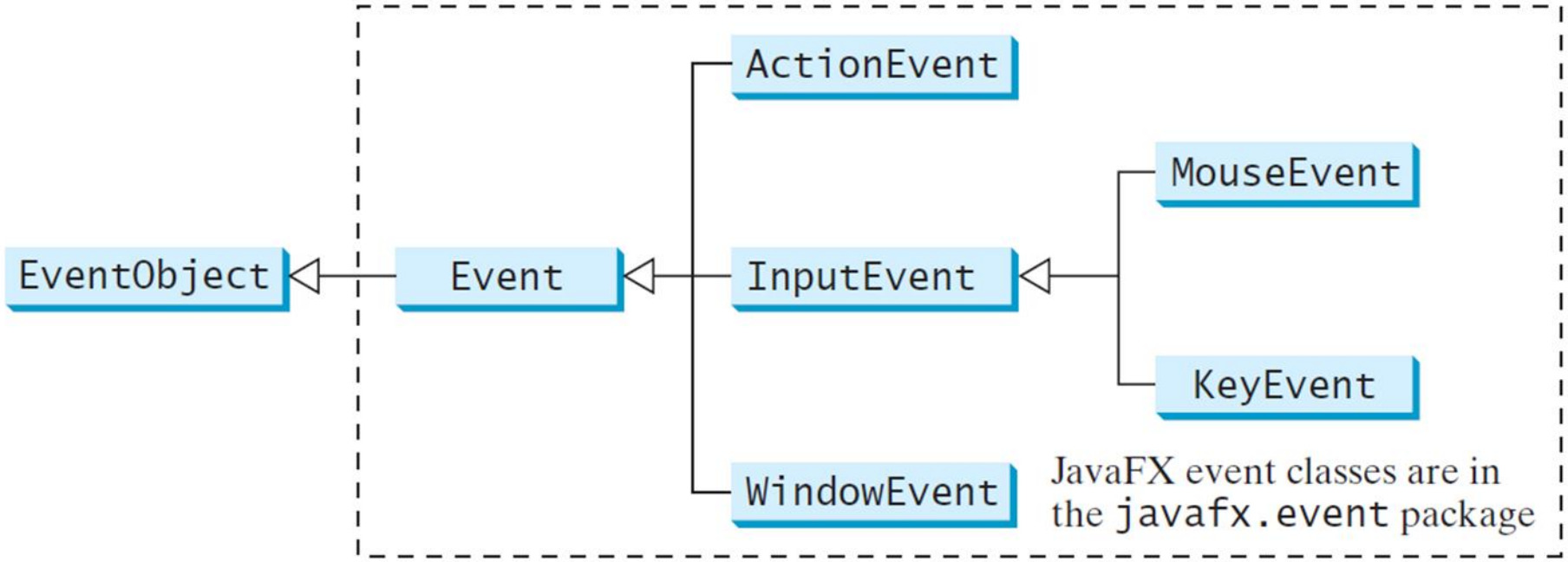
2. Bruker klikker «OK»

3. JVM'et sørger for å kalle handle-metoden i EventHandler objektet.

22

Ulike eventer

- Et event er et signal til programmet om at noe har hendt
- Eventet genereres oftest av eksterne hendelser: museklikk, tastetrykk osv.
- Event-klasser:



Informasjon i eventet

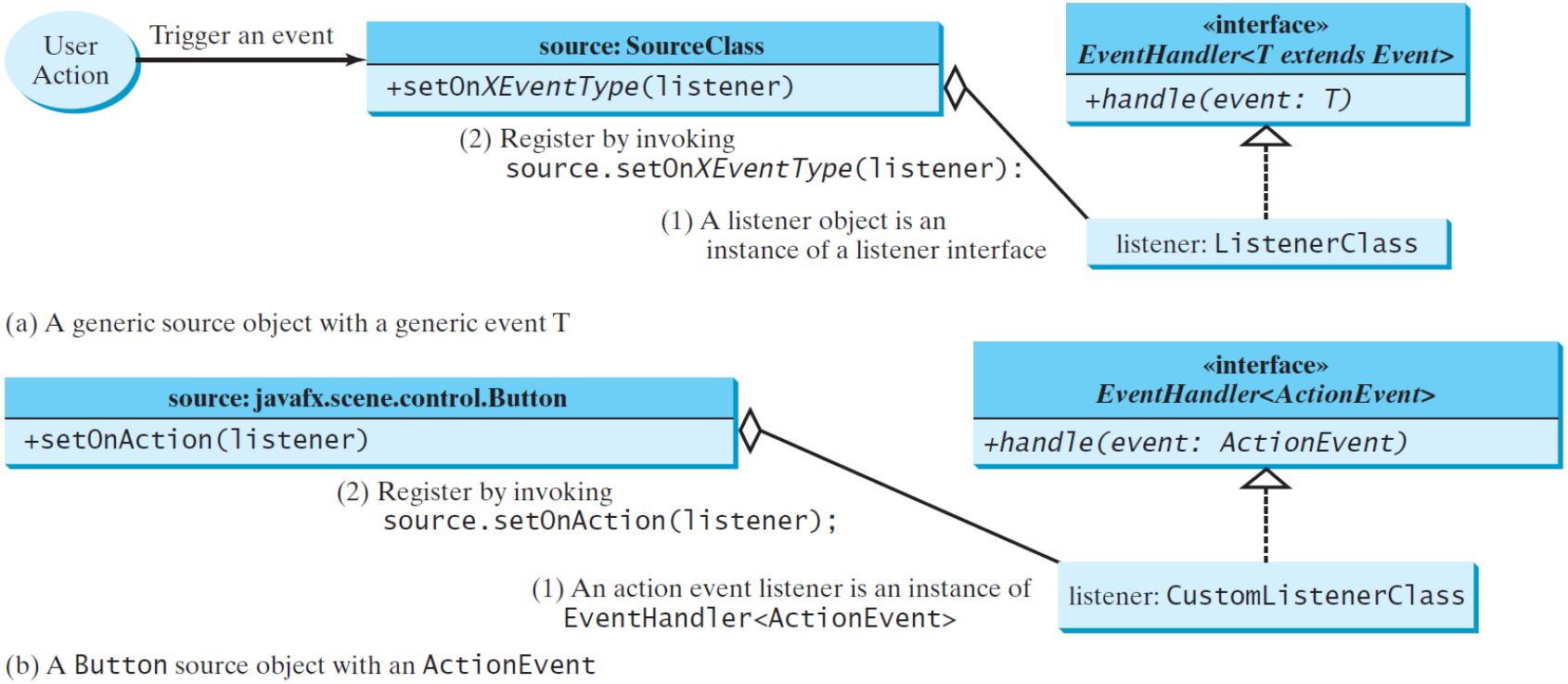
- Ulike typer eventer inneholde forskjellig informasjon
- Alle eventer har getSource()-metode som returnerer source-object
- Kan altså la flere knapper bli håndtert av samme handler, som da må spørre hvilken knapp som genererte eventet
- Alternativt ha en handler for hver knapp, de trenger ikke spørre!
- Mouse-eventer kan fortelle typisk hvor på skjermen det skjedde
- Key-eventer kan fortelle hvilken tast

Noen brukerhandlinger

User Action	*Source Object*	*Event Type Fired*	*Event Registration Method*
Click a button	`Button`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Press Enter in a text field	`TextField`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Check or uncheck	`RadioButton`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Check or uncheck	`CheckBox`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Select a new item	`ComboBox`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Mouse pressed	`Node, Scene`	`MouseEvent`	`setOnMousePressed(EventHandler<MouseEvent>)`
Mouse released			`setOnMouseReleased(EventHandler<MouseEvent>)`
Mouse clicked			`setOnMouseClicked(EventHandler<MouseEvent>)`
Mouse entered			`setOnMouseEntered(EventHandler<MouseEvent>)`
Mouse exited			`setOnMouseExited(EventHandler<MouseEvent>)`
Mouse moved			`setOnMouseMoved(EventHandler<MouseEvent>)`
Mouse dragged			`setOnMouseDragged(EventHandler<MouseEvent>)`
Key pressed	`Node, Scene`	`KeyEvent`	`setOnKeyPressed(EventHandler<KeyEvent>)`
Key released			`setOnKeyReleased(EventHandler<KeyEvent>)`
Key typed			`setOnKeyTyped(EventHandler<KeyEvent>)`

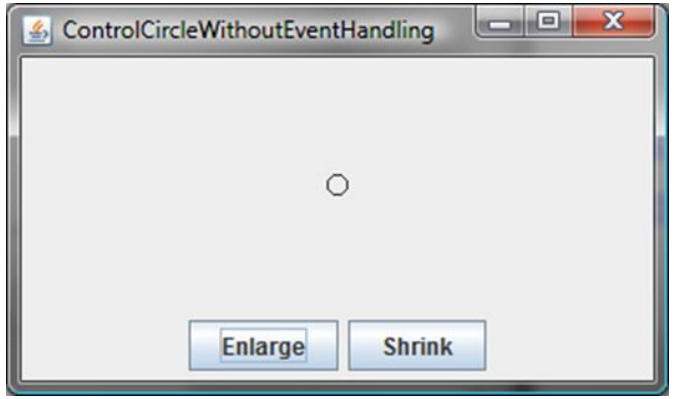
The Delegation Model

Nok en komplisert figur. Ikke bruk alt for mye tid på den. Poenget er at den som skal håndtere eventet, «handler», må være objekt av en klasse som implementerer EventHandler< eventType > der eventType er gitt på forrige lysark. Klassen må implementere metoden «handle» med en parameter av eventType. Eksempel:




```
Button btOK = new Button("OK");
OKHandlerClass handler = new OKHandlerClass();
btOK.setOnAction(handler);
```

ControlCircle v.1

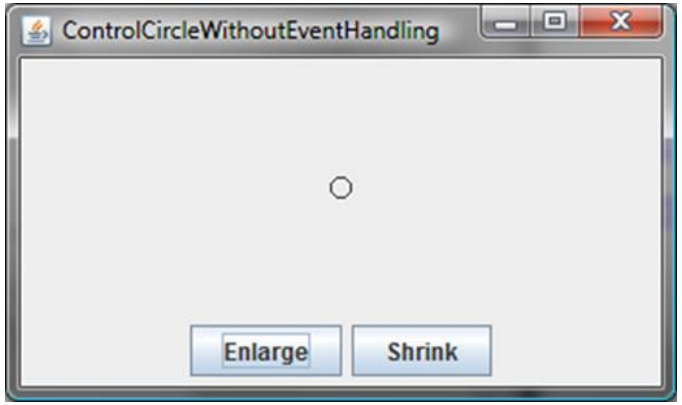


Versjon 1 bygger grensesnittet, men ingen aktivitet – ingen eventhåndtering.

```
public class ControlCircleWithoutEventHandling extends Application {
  @Override
  public void start(Stage primaryStage) {
    StackPane pane = new StackPane();
    Circle circle = new Circle(50);
    circle.setStroke(Color.BLACK);
    circle.setFill(Color.WHITE);
    pane.getChildren().add(circle);

    HBox hBox = new HBox();
    hBox.setSpacing(10);
    hBox.setAlignment(Pos.CENTER);
    Button btEnlarge = new Button("Enlarge");
    Button btShrink = new Button("Shrink");
    hBox.getChildren().add(btEnlarge);
    hBox.getChildren().add(btShrink);

    BorderPane borderPane = new BorderPane();
    borderPane.setCenter(pane);
    borderPane.setBottom(hBox);
    BorderPane.setAlignment(hBox, Pos.CENTER);

    Scene scene = new Scene(borderPane, 200, 150);
    primaryStage.setTitle("ControlCircle");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
```

ControlCircle v.2

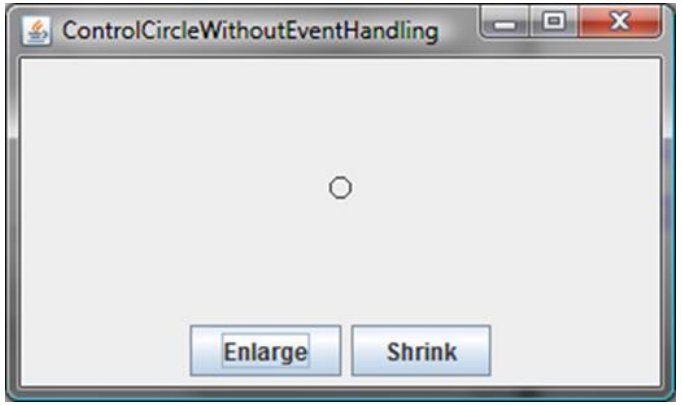
Versjon 2 med EventHandler
Del 1: start-metoden




```
public class ControlCircle extends Application {
  private CirclePane circlePane = new CirclePane();

  @Override
  public void start(Stage primaryStage) {
    HBox hBox = new HBox();
    hBox.setSpacing(10);
    hBox.setAlignment(Pos.CENTER);
    Button btEnlarge = new Button("Enlarge");
    Button btShrink = new Button("Shrink");
    hBox.getChildren().add(btEnlarge);
    hBox.getChildren().add(btShrink);

    btEnlarge.setOnAction(new EnlargeHandler());

    BorderPane borderPane = new BorderPane();
    borderPane.setCenter(circlePane);
    borderPane.setBottom(hBox);
    BorderPane.setAlignment(hBox, Pos.CENTER);

    Scene scene = new Scene(borderPane, 200, 150);
    primaryStage.setTitle("ControlCircle");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
```

ControlCircle v.2

Versjon 2 med EventHandler
Del 2: class EnlargeHandler
og class CirclePane




```
  class EnlargeHandler implements EventHandler<ActionEvent> {
    @Override
    public void handle(ActionEvent e) {
      circlePane.enlarge();
    }
  }  // End class EnlargeHandler
}  // End class ControlCircle

class CirclePane extends StackPane {
  private Circle circle = new Circle(50);

  public CirclePane() {
    getChildren().add(circle);
    circle.setStroke(Color.BLACK);
    circle.setFill(Color.WHITE);
  }

  public void enlarge() {
    circle.setRadius(circle.getRadius() + 2);
  }

  public void shrink() {
    circle.setRadius(circle.getRadius() > 2 ?
      circle.getRadius() - 2 : circle.getRadius());
  }
}
```

EnlargeHandler har tilgang til circlePane fordi den ligger ***inni*** ControlCircle!

Nytt Java-stoff:

- Inner class – indre klasse – har vi nå sett eksempel på
- Anonym indre klasse – klasse uten navn
- Lambda expressions – enda kortere syntaks

Inner class

- «Handler»-klassene, også kalt lytter-klasser, brukes bare inni applikasjons-klassa
- Kan derfor også defineres inni applikasjon-klassa!
- Dermed har de tilgang til alt i applikasjonsklassa, her brukes circlePane
- Den indre klassa kan være static eller ikke-static, her forutsettes **ikke-static**.
- Den indre klassa kan ha de vanlige beskyttelsesgradene
- Instanser av indre klasse opprettes oftest av objekt av ytre klasse

Generelt eksempel

```
public class ShowInnerClass {
  private int data;

  // A method in the outer class
  public void m() {
    // Do something
    InnerClass instance = new InnerClass();
  }

  // An inner class
  class InnerClass {
    // A method in the inner class
    public void mi() {
      // Directly reference data and
      // method defined in outer class
      data++;
      m();
    }  // End of mi()
  }  // End class InnerClass
}  // End class ShowInnerClass
```

Anonym indre klasse

- Indre klasse uten navn.
- Lager klasse og oppretter ett objekt i samme operasjon, trenger da ikke navn
- Må arve fra superklasse eller implementere grensesnitt
- Gir kortere kode når vi bruker dette til lyttere

Med indre klasse

```
public void start(Stage primaryStage){
  // Omitted

  btEnlarge.setOnAction(
    new EnlargeHandler());
}

class EnlargeHandler
  implements EventHandler<ActionEvent>{
  public void handle(ActionEvent e){
    circlePane.enlarge();
  }
}
```

På veg...

```
public void start(Stage primaryStage){
  // Omitted

  btEnlarge.setOnAction(
    new class EnlargeHandler
  implements EventHandler<ActionEvent>(){
    public void handle(ActionEvent e){
      circlePane.enlarge();
    }
  })
}
```

Med anonym indre klasse

```
public void start(Stage primaryStage){
  // Omitted

  btEnlarge.setOnAction(
    new EventHandler<ActionEvent>(){
    public void handle(ActionEvent e){
      circlePane.enlarge();
    }
  })
}
```

Lambda uttrykk

- Enda kortere syntaks enn med anonym indre klasse. Eksempel:

```
public void start(Stage primaryStage){
  // Omitted

  btEnlarge.setOnAction(
    new EventHandler<ActionEvent>(){
    public void handle(ActionEvent e){
      circlePane.enlarge();
    }
  })
}
```

```
public void start(Stage primaryStage){
  // Omitted

  btEnlarge.setOnAction( e -> {
    circlePane.enlarge();
  })
}
```

- Kan brukes når anonym indre klasse bare har én metode – implementerer grensesnitt med én metode, såkalt *funksjonelt grensesnitt*
- «e» i eksempelet er fomalparameter – kan brukes i metoden
- Et lambda-uttrykk kan sees som en anonym metode i en anonym klasse!

Lambda uttrykk

Anta én parameter og én setning som skal utføres. Da er disse fire likeverdige:

```
(ActionEvent e) -> {
  circlePane.enlarge(); }        // Alt er med

(e) -> {
  circlePane.enlarge(); }        // Dropper parametertype

e -> {
  circlePane.enlarge(); }        // Dropper også parentes rundt parameter

e -> circlePane.enlarge()        // Dropper også {} rundt kode
```

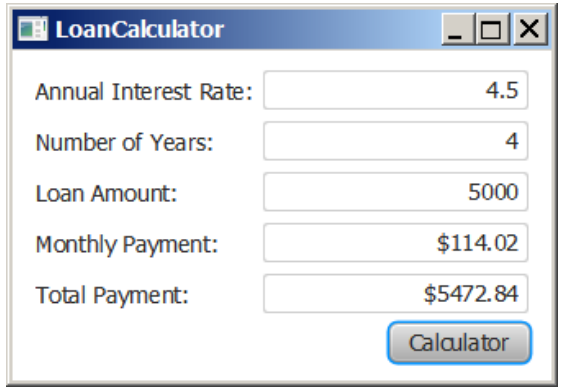
Dersom flere parametere eller parametertype må du ha parentes rundt.

Dersom flere setninger må du ha {} rundt.

Merk! Ikke semikolon!

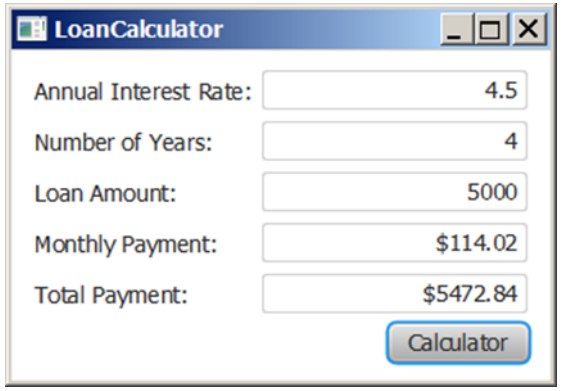
Vi skal ikke forsøke å lære alt rundt Lambda-uttrykk. Følg mønsteret her ved enkle tilfeller!

Lånekalkulator II




```
public class LoanCalculator extends Application {
  private TextField tfAnnualInterestRate = new TextField();
  private TextField tfNumberOfYears = new TextField();
  private TextField tfLoanAmount = new TextField();
  private TextField tfMonthlyPayment = new TextField();
  private TextField tfTotalPayment = new TextField();
  private Button btCalculate = new Button("Calculate");


  @Override
  public void start(Stage primaryStage) {
    // Create UI
    GridPane gridPane = new GridPane();
    gridPane.setHgap(5);
    gridPane.setVgap(5);
    gridPane.add(new Label("Annual Interest Rate:"), 0, 0);
    gridPane.add(tfAnnualInterestRate, 1, 0);
    gridPane.add(new Label("Number of Years:"), 0, 1);
    gridPane.add(tfNumberOfYears, 1, 1);
    gridPane.add(new Label("Loan Amount:"), 0, 2);
    gridPane.add(tfLoanAmount, 1, 2);
    gridPane.add(new Label("Monthly Payment:"), 0, 3);
    gridPane.add(tfMonthlyPayment, 1, 3);
    gridPane.add(new Label("Total Payment:"), 0, 4);
    gridPane.add(tfTotalPayment, 1, 4);
    gridPane.add(btCalculate, 1, 5);
```

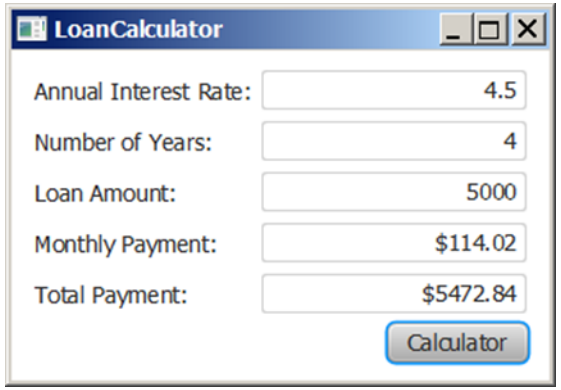
Lånekalkulator II




```
  // Set properties for UI
  gridPane.setAlignment(Pos.CENTER);
  tfAnnualInterestRate.setAlignment(Pos.BOTTOM_RIGHT);
  tfNumberOfYears.setAlignment(Pos.BOTTOM_RIGHT);
  tfLoanAmount.setAlignment(Pos.BOTTOM_RIGHT);
  tfMonthlyPayment.setAlignment(Pos.BOTTOM_RIGHT);
  tfTotalPayment.setAlignment(Pos.BOTTOM_RIGHT);
  tfMonthlyPayment.setEditable(false);
  tfTotalPayment.setEditable(false);
  GridPane.setHalignment(btCalculate, HPos.RIGHT);

  // Process events
  btCalculate.setOnAction(e -> calculateLoanPayment());

  // Create a scene and place it in the stage
  Scene scene = new Scene(gridPane, 400, 250);
  primaryStage.setTitle("LoanCalculator");
  primaryStage.setScene(scene);
  primaryStage.show();
}  // end start
```

Vedd klikk på btCalculate skal calculateLoanPayment-metoden kalles!

Lånekalkulator III




```
private void calculateLoanPayment() {
   // Get values from text fields
   double interest = Double.parseDouble(tfAnnualInterestRate.getText());
   int year = Integer.parseInt(tfNumberOfYears.getText());
   double loanAmount = Double.parseDouble(tfLoanAmount.getText());

   // Create a loan object. Loan defined in Listing 10.2
   Loan loan = new Loan(interest, year, loanAmount); // Denne har ikke GUI!

   // Display monthly payment and total payment
   tfMonthlyPayment.setText(String.format("$%.2f", loan.getMonthlyPayment()));
   tfTotalPayment.setText(String.format("$%.2f", loan.getTotalPayment()));
 }
```