LUKU 17

MUUTAMIA JAVA FX -SOVELLUKSIA

Tässä dokumentissa esitellään muutamia Java FX -sovelluksia.

Kaikista sovelluksista esitellään niiden ohjelmakoodi muutamine selityksineen ja lisäksi on kuva kunkin sovelluksen tuottamasta ikkunasta.

Alussa, heti seuraavalla aukeamalla, esitellään yksi Java CLI -sovellus, eli ohjelma joka toimii komentorivi-ikkunassa.

Viimeisellä sivulla on taulukko Java-kielen muuttujatyypeistä.

2015-10-11 Tiedosto luotu.
2015-10-11 Viimeisin muutos

Tässä esitellään ja luodaan taulukko johon voidaan tallettaa 100 **int**-arvoa. Taulukko vaatii 400 tavua muistia. **luvun_indeksi** on muuttuja jota käytetään taulukon indeksoinnissa. Kun **luvun_indeksi** on alussa arvossa 0, viitataan ensimmäiseen taulukon **lukutaulukko** paikkaan.

Muuttujan **luku_nappaimistolta** arvo kopioidaan taulukon **lukutaulukko** siihen positioon jonka määrittää muuttujan **luvun_indeksi** arvo. Kun muuttujan **luvun_indeksi** arvoa kasvatetaan tämän käskylauseen jälkeen, seuraava kokonaisluku taulukon seuraavaan paikkaan (positioon).

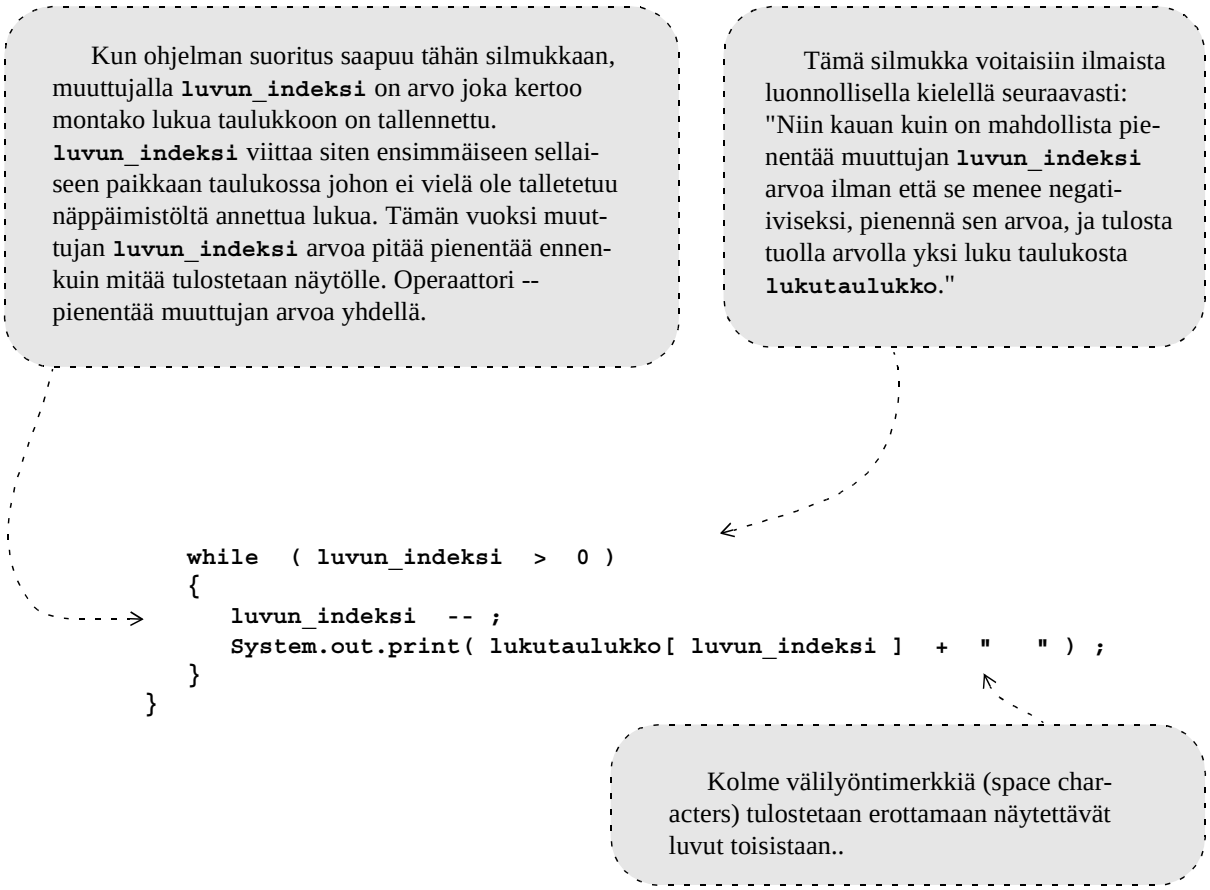
```
//  Takaperin.java  (c) Kari Laitinen

import java.util.* ;

class Takaperin
{
   public static void main( String[] ei_kaytossa )
   {
      Scanner nappaimisto = new Scanner( System.in ) ;

      int[] lukutaulukko         =  new  int[ 100 ] ;
      int   luvun_indeksi        =  0 ;
      int   luku_nappaimistolta  =  0 ;

      System.out.print("\n Tama ohjelma lukee kokonaislukuja nappaimistolta."
                  +  "\n Kun annetaan luku nolla, ohjelma tulostaa luvut"
                  +  "\n kaanteisessa jarjestyksessa. Aloita lukujen"
                  +  "\n syottaminen ja anna lopuksi nolla.\n\n") ;
      do
      {
         System.out.print( " "  +  luvun_indeksi  +  "  Anna kokonaisluku: ") ;
         luku_nappaimistolta  =  nappaimisto.nextInt() ;

         lukutaulukko[ luvun_indeksi ]  =  luku_nappaimistolta ;
         luvun_indeksi  ++ ;
      }
       while  ( luku_nappaimistolta  !=  0 ) ;

      System.out.print( "\n Luvut takaperin:  " ) ;

      while  ( luvun_indeksi  >  0 )
      {
         luvun_indeksi  -- ;
         System.out.print( lukutaulukko[ luvun_indeksi ]  +  "   " ) ;
      }
   }
}
```

Tämä **while**-silmukka tulostaa taulukon sisällön näytölle. Koska muuttujan **luvun_indeksi** pienenee silmukan sisällä, numerot tulevat tulostetuksi käänteisessä järjestyksessä.

Tämä **do-while**-silmukka päättyy kun käyttäjä syöttää näppäimistöltä nollan. Myös tuo nolla talletetaan taulukkoon. Silmukan toiminnan päättyessä muuttujalla **luvun_indeksi** on arvo joka kuvaa annettujen ja taulukkoon talletettujen lukujen määrää. Viimeisenä annettu nolla on myös laskettu mukaan tuohon lukujen määrään.

do-while-silmukoiden erityispiirre on että ne suoritetaan aina ainakin kerran. Tämä silmukka toimii myös siinä tapauksessa että näppäimistöltä annetaan pelkkä nolla.

Takaperin.java - 1.+ CLI-sovellus joka tulostaa annetut luvut käänteisessä järjestyksessä.



Takaperin.java - 1 - 1. Silmukka joka tulostaa luvut takaperin.

```
D:\javaohjelmat2>java Takaperin

 Tama ohjelma lukee kokonaislukuja nappaimistolta.
 Kun annetaan luku nolla, ohjelma tulostaa luvut
 kaanteisessa jarjestyksessa. Aloita lukujen
 syottaminen ja anna lopuksi nolla.

 0  Anna kokonaisluku: 22
 1  Anna kokonaisluku: 33
 2  Anna kokonaisluku: 444
 3  Anna kokonaisluku: 555
 4  Anna kokonaisluku: 6666
 5  Anna kokonaisluku: 7777
 6  Anna kokonaisluku: 88
 7  Anna kokonaisluku: 99
 8  Anna kokonaisluku: 0

 Luvut takaperin:  0   99   88   7777   6666   555   444   33   22
```

Takaperin.java - X. Tässä ohjelma on suoritettu antamalla sille 9 kokonaislukua.

Tällä ja seuraavalla sivulla esitellään Java FX -sovellus jossa luodaan erilaisia graafisia olioita standardiluokkien **`Text`**, **`Line`**, **`Rectangle`**, **`Circle`**, ja **`Arc`** avulla.

Suorakaidetta määriteltäessä annetaan parametreina suorakaiteen vasemman yläkulman koordinaatit sekä suorakaiteen leveys ja suorakaiteen korkeus.

```
//  ShapesDemoFX.java Copyright (c) Kari Laitinen

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.Group;
import javafx.scene.text.Text ;
import javafx.stage.Stage;
import javafx.scene.shape.* ;
import javafx.scene.paint.Color;

public class ShapesDemoFX extends Application
{
   public void start( Stage stage )
   {
      stage.setTitle( "ShapesDemoFX.java" ) ;

      Group group_for_shapes = new Group() ;

      Scene scene = new Scene( group_for_shapes, 896, 512 ) ;

      scene.setFill( Color.LIGHTYELLOW ) ;

      Text scene_size_text = new Text( 20, 20,
                                  "Scene size is " + (int) scene.getWidth()
                                  + " x " + (int) scene.getHeight() ) ;

      Line blue_horizontal_line = new Line( 64, 128, 512, 128 ) ;
      blue_horizontal_line.setStroke( Color.BLUE ) ;
      blue_horizontal_line.setStrokeWidth( 3 ) ;

      Rectangle cyan_square = new Rectangle( 64, 192, 128, 128 ) ;
      cyan_square.setFill( Color.CYAN ) ;

      Rectangle magenta_rectangle = new Rectangle( 266, 202, 128, 108 ) ;
      magenta_rectangle.setFill( Color.MAGENTA ) ;

      Rectangle frame_around_magenta_rectangle =
                                 new Rectangle( 256, 192, 148, 128 ) ;
      frame_around_magenta_rectangle.setFill( Color.TRANSPARENT ) ;
      frame_around_magenta_rectangle.setStroke( Color.BLUE ) ;
      frame_around_magenta_rectangle.setStrokeWidth( 3 ) ;


      Circle yellow_ball = new Circle( 512, 256, // center point (512, 256)
                                  64,       // radius is 64 points
                                  Color.YELLOW ) ;
      yellow_ball.setStroke( Color.BLUE ) ;
      yellow_ball.setStrokeWidth( 3 ) ;
```

ShapesDemoFX.java - 1: Ohjelma joka demonstroi graafisten olioiden käyttöä.

Tässä määritellään Pacman-kuvio jonka ’suu’ aukeaa oikealle. Pacmanin pään muoto on ympyrämäinen koska korkeutta ja leveyttä kuvaavat säteet on asetettu samoiksi. Suun määrittävässä koordinaatistossa nolla astetta on ’idässä’ ja 90 astetta on ’pohjoisessa’.

```
        Arc pacman_shape = new Arc( 704, 256, // center point
                                    64,  64, // width and height radiuses
                                    45,      // start angle in degrees
                                    270 ) ;  // length angle in degrees
        pacman_shape.setType( ArcType.ROUND ) ;
        pacman_shape.setFill( Color.LIGHTGREY ) ;
        pacman_shape.setStroke( Color.BLACK ) ;
        pacman_shape.setStrokeWidth( 3 ) ;

        // The 'pie slice' represents the 'missing' part of pacman shape

        Arc pie_slice = new Arc( 768, 256, 64, 64, 315, 90 ) ;
        pie_slice.setType( ArcType.ROUND ) ;
        pie_slice.setFill( Color.LIGHTGREY ) ;
        pie_slice.setStroke( Color.BLACK ) ;
        pie_slice.setStrokeWidth( 3 ) ;

        Line black_horizontal_line = new Line( 512, 384, 768, 384 ) ;
        black_horizontal_line.setStroke( Color.BLACK ) ;
        black_horizontal_line.setStrokeWidth( 3 ) ;

        group_for_shapes.getChildren().addAll( scene_size_text,
                                               blue_horizontal_line,
                                               cyan_square, magenta_rectangle,
                                               frame_around_magenta_rectangle,
                                               yellow_ball,
                                               pacman_shape, pie_slice,
                                               black_horizontal_line ) ;

        stage.setScene( scene ) ;
        stage.show();
    }

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

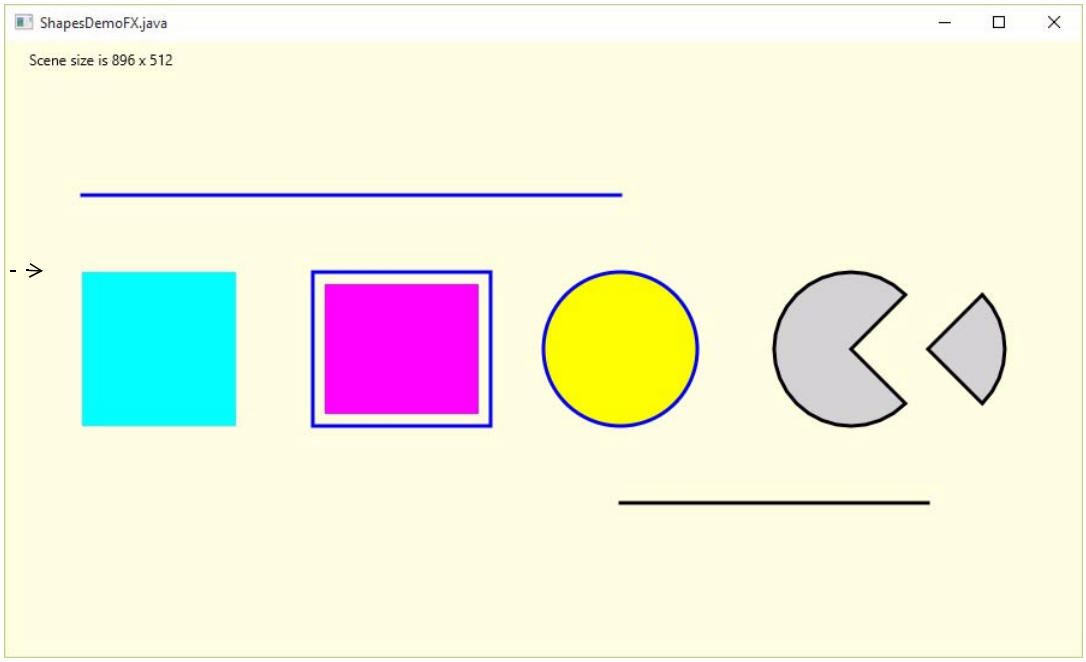
Tässä lisätään kaikki luodut graafiset oliot sovelluksen **Group**-olion lapsilistaan, minkä jälkeen kyseiset oliot tulevat näkyville ikkunaan.

Java FX -sovelluksissa käytettävä **main()**-metodi on näissä esimerkeissä aina samanlainen.

Tämän ohjelman tuottama näkymä on esitelty seuraavalla sivulla.

ShapesDemoFX.java - 2. Ohjelman loppuosa.

Tämän suorakaiteen (joka on siis neliö) vasen yläkulma on pisteessä (64, 192). Graafisen koordinaatiston nollapiste on ikkunan vasen yläkulma. x-koordinaatti kasvaa oikealle ja y-koordinaatti kasvaa alaspäin.



ShapesDemoFX.java - X. Ohjelman tuottama ikkuna.

Tällä ja seuraavilla kolmella sivulla esitelty ohjelma demonstroi kuinka Java FX -sovelluksissa reagoidaan erilaisiin hiiritapahtumiin.

Näiden **DoubleProperty**-olioiden arvoja muutetaan silloin kun hiirtä liikutellaan tämän sovelluksen ikkunassa. Ohjelma on rakennettu niin että näiden olioiden sisältämät arvot vaikuttavat näytöllä olevien ellipsien eli soikioiden paikkoihin.

```
//  MouseDemoFX.java Copyright (c) Kari Laitinen

import javafx.application.Application;
import javafx.scene.input.* ; // MouseEvent, MouseButton
import javafx.scene.* ;
import javafx.stage.Stage;
import javafx.geometry.* ; // Point2D
import javafx.scene.paint.Color;
import javafx.scene.shape.Ellipse ;
import javafx.beans.property.* ; // DoubleProperty etc.

public class MouseDemoFX extends Application
{
   Group group_for_ellipses = new Group() ;

   Ellipse large_ellipse ;
   Ellipse left_small_ellipse, middle_small_ellipse, right_small_ellipse ;

   DoubleProperty event_position_x  =  new SimpleDoubleProperty() ;
   DoubleProperty event_position_y  =  new SimpleDoubleProperty() ;

   public void start( Stage stage )
   {
      stage.setTitle( "MouseDemoFX.java" ) ;

      Scene scene = new Scene( group_for_ellipses, 800, 600 ) ;

      scene.setFill( Color.LAVENDER ) ;

      // The large ellipse will be in the middle of the Scene at
      // the beginning.

      large_ellipse = new Ellipse( scene.getWidth() / 2,
                                   scene.getHeight() / 2, 80, 105 ) ;

      // The small ellipses, which represent the mouse buttons,
      // are defined without a location.

      left_small_ellipse   = new Ellipse( 16, 30 ) ;
      middle_small_ellipse = new Ellipse( 16, 30 ) ;
      right_small_ellipse  = new Ellipse( 16, 30 ) ;

      // The small ellipses are not visible when the program starts running.

      left_small_ellipse.visibleProperty().setValue( false ) ;
      middle_small_ellipse.visibleProperty().setValue( false ) ;
      right_small_ellipse.visibleProperty().setValue( false ) ;
```

MouseDemoFX.java - 1: Ohjelma joka demonstroi hiiritapahtumiin reagointia.

Ohjelmassa suuri harmaa ellipsi kuvaa koko hiirtä ja pienet ellipsit kuvaavat hiiren nappeja.

Tässä asetetaan **DoubleProperty**-tyyppisille luokan datajäsenille kuuntelijat. Näiden avulla saadaan hiirtä ja sen nappeja kuvaavien ellipsien paikat muuttumaan silloin kun hiirtä liikutetaan.

```
large_ellipse.setFill( Color.GRAY ) ;

large_ellipse.setStroke( Color.BLACK ) ;

left_small_ellipse.setStroke( Color.BLACK ) ;
middle_small_ellipse.setStroke( Color.BLACK ) ;
right_small_ellipse.setStroke( Color.BLACK ) ;

// The following statements add listeners which specify how the
// ellipses are re-located when the mouse cursor position changes.
// The listeners are made with Lambda expressions.

event_position_x.addListener(

   ( observable_value, value, new_value ) ->
   {
      large_ellipse.setCenterX( new_value.doubleValue() ) ;

      left_small_ellipse.setCenterX( new_value.doubleValue() - 40 ) ;
      middle_small_ellipse.setCenterX( new_value.doubleValue() ) ;
      right_small_ellipse.setCenterX( new_value.doubleValue() + 40 ) ;
   }

) ;

event_position_y.addListener(

   ( observable_value, value, new_value ) ->
   {
      large_ellipse.setCenterY( new_value.doubleValue() + 75 ) ;

      left_small_ellipse.setCenterY( new_value.doubleValue() + 35 ) ;
      middle_small_ellipse.setCenterY( new_value.doubleValue() + 25 ) ;
      right_small_ellipse.setCenterY( new_value.doubleValue() + 35 ) ;
   }

) ;


group_for_ellipses.getChildren().addAll( large_ellipse,
                                         left_small_ellipse,
                                         middle_small_ellipse,
                                         right_small_ellipse ) ;
```

MouseDemoFX.java - 2: Ohjelman jatkoa.

Sovelluksen skenen eli käytännössä sen ikkunan taustaväri pannaan erilaiseksi silloin kun hiiri viedään sovelluksen ikkunan päälle.

Tässä tallennetaan hiiren kursorin paikan koordinaatit **DoubleProperty**-tyyppisiin datajäseniin joista ne sitten vaikuttavat hiirtä kuvaavien ellipsien paikkoihin.

```
scene.setOnMouseEntered( ( MouseEvent event ) ->
{
   scene.setFill( Color.LIGHTSKYBLUE ) ;
} ) ;

scene.setOnMouseExited( ( MouseEvent event ) ->
{
   scene.setFill( Color.LAVENDER ) ;
} ) ;

scene.setOnMousePressed( ( MouseEvent event ) ->
{
   event_position_x.setValue( event.getSceneX() ) ;
   event_position_y.setValue( event.getSceneY() ) ;

   // A small ellipse is made visible when a mouse button is pressed.

   if ( event.getButton()  ==  MouseButton.PRIMARY ) // Left
   {
      left_small_ellipse.visibleProperty().setValue( true ) ;
   }
   else if ( event.getButton()  ==  MouseButton.MIDDLE )
   {
      middle_small_ellipse.visibleProperty().setValue( true ) ;
   }
   else if ( event.getButton()  ==  MouseButton.SECONDARY ) // Right
   {
      right_small_ellipse.visibleProperty().setValue( true ) ;
   }

   //  The small ellipses will be shown with a different color
   //  if either the Ctrl or the Shift key is pressed down.

   if ( event.isControlDown() ==  true  ||
        event.isShiftDown()   ==  true )
   {
      left_small_ellipse.setFill( Color.MAGENTA ) ;
      middle_small_ellipse.setFill( Color.MAGENTA ) ;
      right_small_ellipse.setFill( Color.MAGENTA ) ;
   }
   else
   {
      left_small_ellipse.setFill( Color.YELLOW ) ;
      middle_small_ellipse.setFill( Color.YELLOW ) ;
      right_small_ellipse.setFill( Color.YELLOW ) ;
   }
} ) ;
```

MouseDemoFX.java - 3: Hiireen reagoivia Lambda-lausekkeita.

Tässä kaikki hiiritapahtumumiin reagoivat Lambda-lausekkeet kiinnitetään nimenomaan **Scene**-olioon.

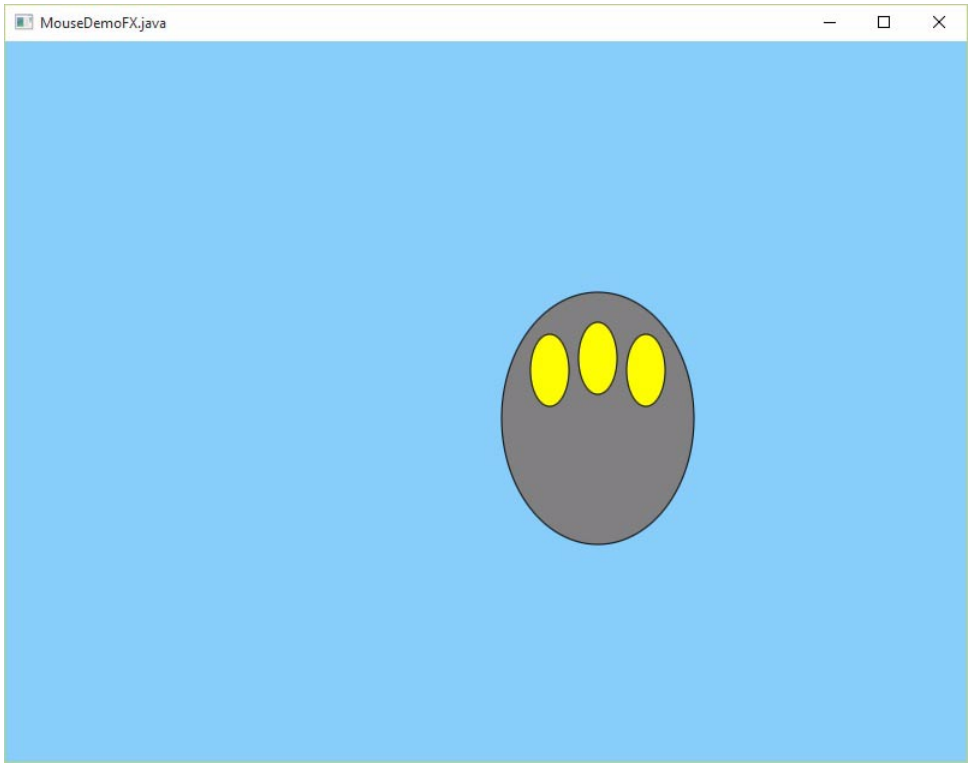
MouseEvent-tyyppisestä oliosta voidaan kaivaa tietoa hiiritapahtumaan liittyen. Tässä esimerkiksi otetaan selville mitä hiiren nappia käytettiin.

```
        //  The following statement specifies the actions when the mouse
        //  is moved without pressing its buttons.

        scene.setOnMouseMoved( ( MouseEvent event ) ->
        {
           event_position_x.setValue( event.getSceneX() ) ;
           event_position_y.setValue( event.getSceneY() ) ;
        } ) ;

        //  Dragging of the mouse means that it is moved while simultaneously
        //  pressing one or more of its buttons.

        scene.setOnMouseDragged( ( MouseEvent event ) ->
        {
           event_position_x.setValue( event.getSceneX() ) ;
           event_position_y.setValue( event.getSceneY() ) ;
        } ) ;

        scene.setOnMouseReleased( ( MouseEvent event ) ->
        {
           if ( event.getButton()  ==  MouseButton.PRIMARY ) // Left
           {
              left_small_ellipse.visibleProperty().setValue( false ) ;
           }
           else if ( event.getButton()  ==  MouseButton.MIDDLE )
           {
              middle_small_ellipse.visibleProperty().setValue( false ) ;
           }
           else if ( event.getButton()  ==  MouseButton.SECONDARY )
           {
              right_small_ellipse.visibleProperty().setValue( false ) ;
           }
        } ) ;

        stage.setScene( scene ) ;
        stage.show();
     }


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

MouseDemoFX.java - 4. Ohjelman loppuosa.

Alla olevassa kuvassa ohjelmaa on suoritettu siten että hiiri on tuotu sovelluksen ikkunaan ja kaikki hiiren napit on painettu yhtäaikaisesti alas.

Jos tätä ohjelmaa suoritetaan sellaisessa ympäristössä jossa ei ole käytössä kolmea hiiren nappia (esim. MacOS X), alla olevaa näkymää ei välttämättä saada aikaiseksi.



MouseDemoFX.java - X. Ohjelman tuottama ikkuna.

Tällä ja seuraavilla sivuilla esitellään sovellus joka näyttää kolmea palloa ikkunassa, ja noita palloja voidaan liikutella hiiren avulla.

Palloa varten on tässä johdettu **`Circle`**-luokasta oma luokka nimeltä **`Ball`**. **`Ball`**-luokan oliot voidaan aktivoida siten että niihin piirtyy paksumpi reuna.

```
//  MovingBallsWithMouseFX.java Copyright (c) Kari Laitinen

import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.input.MouseEvent;
import javafx.scene.* ;
import javafx.scene.layout.* ;
import javafx.stage.Stage;
import javafx.geometry.* ; // Point2D
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle ;
import javafx.collections.* ; // ObservableList etc.

class Ball extends Circle
{
   public Ball( double given_center_point_x,
                double given_center_point_y,
                double given_radius,
                Color  given_color )
   {
      super( given_center_point_x, given_center_point_y,
             given_radius, given_color ) ;

      setStroke( Color.BLACK ) ;
      setStrokeWidth( 2 ) ;
   }

   public void activate_ball()
   {
      setStrokeWidth( 6 ) ;
   }

   public void deactivate_ball()
   {
      setStrokeWidth( 2 ) ;
   }

   public void  move_this_ball( double movement_in_direction_x,
                                double movement_in_direction_y )
   {
      setCenterX( getCenterX()  +  movement_in_direction_x ) ;
      setCenterY( getCenterY()  +  movement_in_direction_y ) ;
   }

}
```

MovingBallsWithMouseFX.java - 1: Ohjelma jonka palloja voi liikutella hiirellä.

Tämän ’foreach’-silmukan avulla sovelluksen kaikille palloille määritellään hiireen reagointi. Hiiritapahtumien käsittely kiinnitetään nimenomaan **`Ball`**-olioihin. Näin ohjelman ajosysteemi pitää huolen siitä että hiiritapahtuma ’osuu’ oikeaan palloon.

```
public class MovingBallsWithMouseFX extends Application
{
   static final int SCENE_WIDTH  = 800 ;
   static final int SCENE_HEIGHT = 600 ;

   boolean ball_movement_going_on = false ;

   double previous_cursor_position_x, previous_cursor_position_y ;

   Group group_for_balls = new Group() ;

   private void set_mouse_activities_for_balls()
   {
      for ( Node child_in_list : group_for_balls.getChildren() )
      {
         Ball ball_in_list = (Ball) child_in_list ;

         ball_in_list.setOnMousePressed( ( MouseEvent event ) ->
         {
            if ( ball_movement_going_on == false )
            {
               ball_in_list.activate_ball() ;
               previous_cursor_position_x = event.getSceneX() ;
               previous_cursor_position_y = event.getSceneY() ;
               ball_movement_going_on = true ;
            }
         } ) ;

         ball_in_list.setOnMouseDragged( ( MouseEvent event ) ->
         {
            if ( ball_movement_going_on == true )
            {
               double mouse_movement_x  =  event.getSceneX()
                                        -  previous_cursor_position_x ;

               double mouse_movement_y  =  event.getSceneY()
                                        -  previous_cursor_position_y ;

               previous_cursor_position_x  =  event.getSceneX() ;
               previous_cursor_position_y  =  event.getSceneY() ;

               ball_in_list.move_this_ball( mouse_movement_x,
                                            mouse_movement_y ) ;

            }
         } ) ;
```

MovingBallsWithMouseFX.java - 2: Sovellusluokan alkuosa.

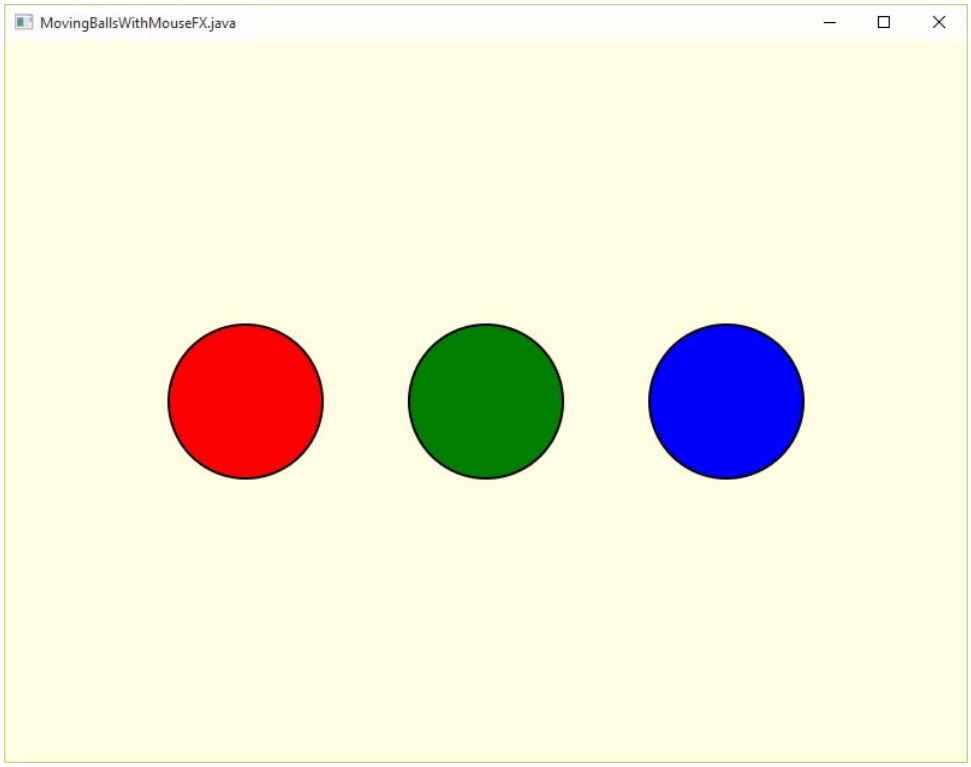
Tässä on edellisellä sivulla alkaneen 'foreach' -silmukan loppuosa. Tässä määritellään mitä tehdään kun hiiren nappi vapautetaan pallon päällä.

Ball-oliot luodaan tässä ja ne kiinnitetään heti sovelluksen **Group**-olion lapsilistaan.

```
            ball_in_list.setOnMouseReleased( ( MouseEvent event ) ->
            {
               if ( ball_movement_going_on == true )
               {
                  ball_in_list.deactivate_ball() ;
                  ball_movement_going_on = false ;
               }
            } ) ;
         }
      }

      public void start( Stage stage )
      {
         group_for_balls.getChildren().add(
              new Ball( SCENE_WIDTH / 4, SCENE_HEIGHT / 2, 64, Color.RED ) ) ;


         group_for_balls.getChildren().add(
              new Ball( SCENE_WIDTH / 2, SCENE_HEIGHT / 2, 64, Color.GREEN ) ) ;


         group_for_balls.getChildren().add(
             new Ball( SCENE_WIDTH * 3 / 4, SCENE_HEIGHT / 2, 64, Color.BLUE ) ) ;


         // Now the Ball objects are stored as 'children' of the Group.
         // The following method call specifies what will happen when
         // the balls are operated with the mouse.

         set_mouse_activities_for_balls() ;

         Scene scene = new Scene( group_for_balls, SCENE_WIDTH, SCENE_HEIGHT ) ;

         scene.setFill( Color.LIGHTYELLOW ) ;

         stage.setTitle( "MovingBallsWithMouseFX.java" ) ;
         stage.setScene( scene ) ;
         stage.show();
      }


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

MovingBallsWithMouseFX.java - 3. Ohjelman loppuosa.

Alla olevassa kuvassa sovellus on juuri käynnistetty eikä palloja ole liikutettu vielä ollenkaan.



MovingBallsWithMouseFX.java - X. Ohjelman tuottama ikkuna.

Table 5-1. Java-ohjelmointikielen muuttujatyypit.

Tyyppi	Muistin määrä	Lukualue
`int`	`4 tavua 32 bittiä`	`-2,147,483,648 ... 2,147,483,647` `-80000000H ... 7FFFFFFFH`
`short`	`2 tavua 16 bittiä`	`-32,768 ... 32,767` `-8000H ... 7FFFH`

`long`	`8 tavua 64 bittiä`	`-9,223,372,036,854,775,808 ...`
		`9,223,372,036,854,775,807`
		`-8000000000000000H ...`
		`7FFFFFFFFFFFFFFFH`

`byte`	`1 tavu 8 bittiä`	`-128 ... 127` `-80H ... 7FH`
`boolean`	`1 tavu 8 bittiä`	`false tai true`
`char`	`2 tavua 16 bittiä`	`Unicode 0 ... 65535`
`float`	`4 tavua 32 bittiä`	`Tarkkuus: 7 desimaalista numeroa` `Alue: +/- 1.5e-45 ... 3.4e38`[a]
`double`	`8 tavua 64 bittiä`	`Tarkkuus: 15 desimaalista numeroa` `Alue: +/- 5.0e-324 ... 1.7e308`
`muistiosoite`[b]	`4 tavua 32 bittiä`	`Riittää osoittamaan 4,294,967,296 tavua (4 gigatavua) keskusmuistia.`

a. Tässä annetut lukualueet ovat likimääräisiä.

b. "muistiosoite" ei ole Javan muuttujatyyppi. Tämä taulukon rivi kertoo kuinka paljon muistia varataan (allokoidaan) silloin kun niin sanottu olioviittaaja esitellään. Olioviittaajat ovat eräänlaisia muuttujia joihin talletetaan heap-muistiin luotujen olioiden osoitteet.