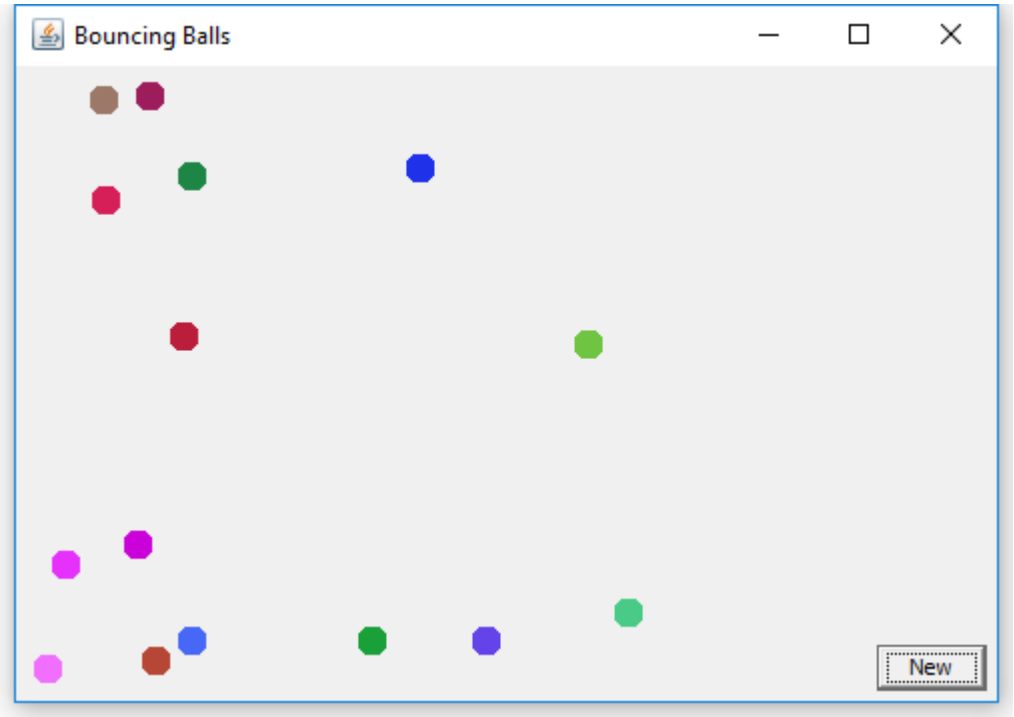


Assignment 4b

Content Application Development Day 4
Lecture
The learned concepts of object-oriented programming, methods, parameters, and the scope of variables have been reviewed during the lecture. The lecture introduced programming of the Arduino and EVShield.
Tutorial
In the tutorial you practise creating classes and methods, and the use of basic sample drawing commands. You also learn how a program can respond to the actions of the user.

At the end of day 4 you will be able to
- Design and apply classes and methods
- Use global and local variables
- Use drawing commands in a user interface.
- Apply various types of Event Handler.

Reading between day 4 and day 5
Book: Chapters 3,4 of "Head First Java", or chapter 3.9 through 4 from the book "*Aan de slag met Java" [Getting Started with Java].*

Extra exercise
You can go through the exercises from chapters 3.9 and 4 to increase your understanding of the subject matter.

Who is this assignment for?
This assignment is intended for those who follow the Application Development course, but who do not participate in the SmartProducts project, or prefer to practice their Java programming skills instead of Arduino programming.

Appendix
The *DrawingPanel.java* file is part of this assignment.

Bouncing balls

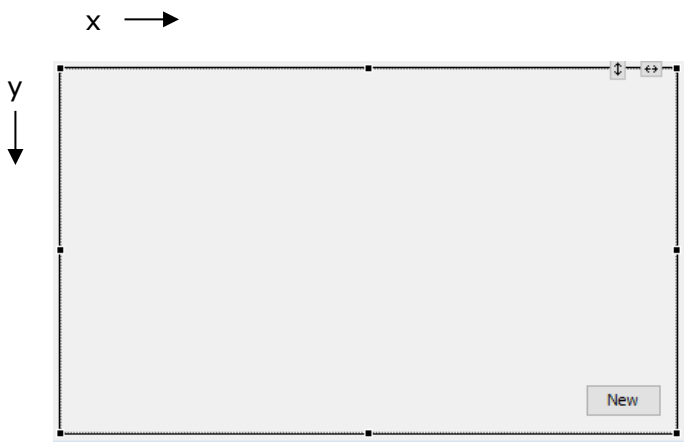
For this assignment, we create a basic game in which a ball appears when a button is pressed. It bounces on the borders of the application. When you click with the mouse on or near the ball, the ball disappears.
Upon appearing, the ball is given a fixed speed and a random color and position.

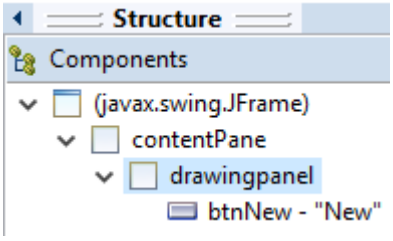
Tutorial steps

1. Create the userinterface
2. Design a class for the Ball
3. Add the DrawingPanel class
4. Create the Event Handlers
5. Respond to resize of the window

1. Create the userinterface

Create a new project "Assignment4b". Create a user interface based on a JFrame Form, with a Panel filling the entire space and in the Panel a Button which function is to make new balls.
Remember to set the layout of both the contentPane and the panel to *GroupLayout*.
Give the Panel the variable name *drawingpanel*.





Because drawing will be done in this panel, the x and y directions are show here. The origin is in the top left of the panel.

2. Design a class for the Ball

Have a look at the **slides of lecture 3**, which discusses mapping a "Ball" in the real world to a class as an example for designing a class.

Add a class to the project, name it "Ball". Consider the properties the ball should have. Add these as class variables, all of which are private.

The Ball should also know the width and height of the panel, so add two class variables for that also. The reason for this is that the Ball must change its path when reaching a border. Name these variables *panelWidth* and *panelHeight*.

Later, the panel must check if the user has clicked near a ball. That's why the panel must be able to retrieve the position of the ball. Think of method(s) for the Ball class that can retrieve the position (getters). You may generate these automatically: Choose *Source > Generate Getters and Setters…*, for *x* and *y*, select *getX()* and *getY()*.

Constructor
The Ball class must also have a constructor that gets the dimensions of the panel as parameters and sets the other class variables. You can automatically generate a constructor in Eclipse via: *Source > Generate Constructor using Fields*. Make sure when selecting the fields you only select the class variables you have defined for the dimensions of the panel, *panelWidth* and *panelHeight*.
The constructor must set a random position (x, y) and a random color. Below is an example of the code for defining a random x. This code uses the Random class. Read the documentation of the

Random class on how the *nextInt()* method of that class works. Generating a random y value is done in a similar way, except that it depends on the height of the Panel (*panelHeight*). Obviously, the names of the variables in this example may differ from those you have chosen.

```
Random r = new Random();
x = r.nextInt(panelWidth);
```

You can use the Color class to define a random color. The constructor of this class can receive three integers from 0-255 as parameters that define the R, G, and B values of the color (RGB). Check the documentation of the Color class if it is still unclear.

Drawing method

Add a drawing method, as you have learned in assignment 2:

```
public void paintComponent(Graphics g) {


}
```

The drawing method uses two drawing commands: 1 to set the color to the color of the Ball (using the class variable for the color you have added) and 1 to draw a filled circle. A circle is the same as an oval of which the width and height are the same.
Each time the ball is drawn, the position must be updated and checked if the Ball has not hit a border. It is useful, therefore, to create a separate method for this and call this method before the ball is drawn in the drawing method. This method will control the movement of the Ball:

Method for moving the Ball:

Add an empty method for moving the ball. Each time the ball is drawn, the x, y position is adjusted according to the speed. Suppose the speed in x direction is *dx*, then moving the ball will have to increase x:

```
x = x + dx
```

You must add class variables for the speed (dx and dy) and give them initial values (eg. 5).

The Ball must also be able to check if it hits a border of the panel. As this is a bit tricky, we give you the code that controls x and checks the borders as an example. Add this to the method you have created for moving the ball:

```
// increase x with dx:
x = x + dx;
if (x < 0) { // hit left side of panel?
   x = 0;
   dx = -dx; // velocity flips (change direction)
}
if (x + size >= panelWidth) { // hit right side of panel?
   x = panelWidth - size;
   dx = -dx; // velocity flips (change direction)
}
```

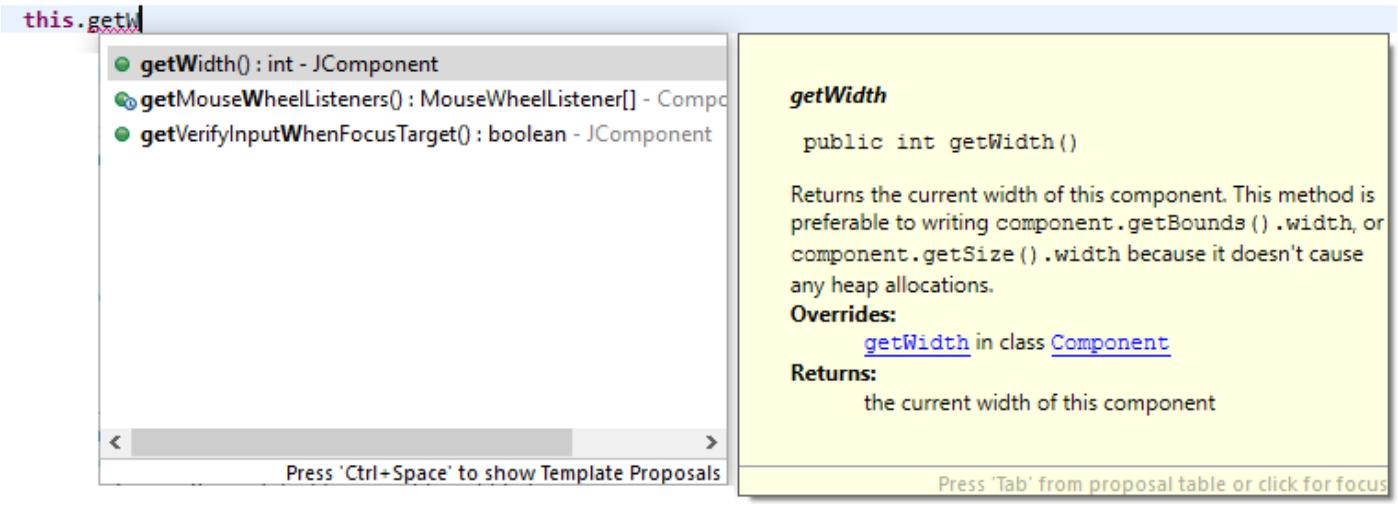
The code for the y will be very similar, which you may figure out yourself. The names of the variables in your class may be different. In this example *x* (and *y*) represent the position of the Ball, *dx* is the speed in the x direction, *size* is the diameter of the Ball and *panelWidth* the width of the panel.

3. Add the DrawingPanel class

Copy the DrawingPanel.java file you have been given from the zip file to the project. This class executes the drawing and has methods for adding a Ball: *add()*, and checking whether a Ball has been clicked: *zap()*.

After you have added the DrawingPanel class, errors will occur. This is logical, for instance because the method names you have chosen in the Ball class may not match the names used in there. You can solve this by checking the errors and adjusting the name, so the correct method of your Ball class is used.

Finish the *add()* method, a line is missing at `// ...`, where a new Ball must be created using a *new* statement. Also check the next line, where the Ball is added to the list. That tells you what the variable name for the Ball should be.
The constructor of the Ball class requires two parameters, the width and the height of the panel. You can retrieve these using two get-methods of the class: type the word *this*, followed by a dot, then you get a list of all methods:

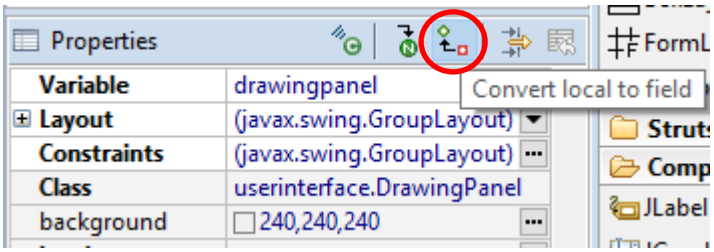


Continue typing "get" and the list displays all methods that start with "get". Find the methods that get the width (type **`this.getW`**) and height and use these as parameters for the constructor.

After all errors are fixed, save the project (*File* > *Save All*).

We are going to use DrawingPanel to draw in the user interface. Go to the userinterface class and switch to the Window Builder, right-click the Panel drawingpanel and use the *Morph* function to change the Panel into a DrawingPanel. Check at assignment 2 if you do not remember how to do this.

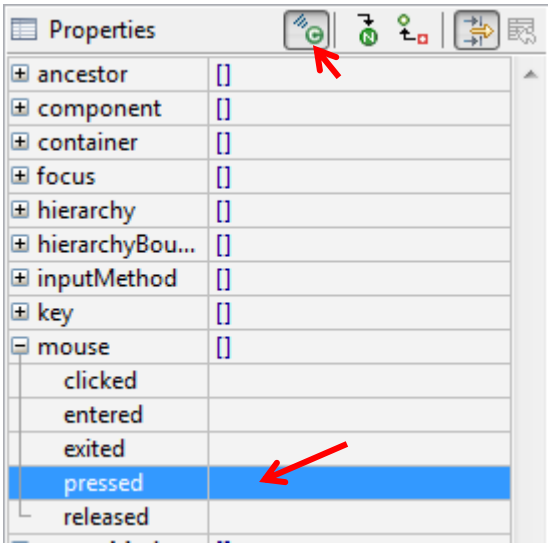
To be able to use the drawingpanel in methods of the userinterface class, this must be a class variable, instead of a variable that exists only in the constructor. Therefore, the declaration and initialization must be separated from each other. Make sure that the declaration is placed at the top in the class. You also did this in assignment 2. Another way to do this is click on the button "convert to local field":



4. Create the Event Handlers

We need two Event Handlers: one that responds when pressing the button and one that responds to mouse clicks in the Panel. The one for the button is simple: double-click on the button in the Window Builder. That Event Handler must call the *add()* method of the drawingpanel.

You can create the second Event Handler that responds to mouse clicks in the Panel as follows: select the drawingpanel in the Window Builder. At Properties click on Show Events , expand "mouse", and double-click inside the field behind "pressed":



You are then taken to the code of the Event Handler: There, add code to get the position and pass it to the zap-method of drawingpanel:

```
public void mousePressed(MouseEvent e) {
    // get position from mouse event:
    int x = e.getY();
    int y = e.getY();
    // pass position to zap-method of drawingpanel:
    drawingpanel.zap(x, y);
}
```

Test the application.

5. Respond to resize of the window

If you change the size of the window (try it), the balls will still keep moving in the original space. To fix this, we need to tell the balls their bounds have changed. First, add a method to the `Balls` class, which can change the class-variables `panelWidth` and `panelHeight`. A header for this method can be:

```
public void setBounds(int w, int h)
```

Add this method and write its code.

Next, add a method to the `DrawingPanel` class which calls the newly added method `setBounds()` for all balls. You will have to add a for-loop for this. Look at the `paintComponent()` method of class `DrawingPanel`. It contains a for-loop which draws all balls. Copy it to this new method. A header for this method can be:

```
public void resetBounds()
```

Inside the for-loop, instead of calling the `paintComponent()` of each ball, call the `setBounds()` method, with the width and height of the panel as parameters. How to get the width and height of the panel was explained in step 3.

Now we need to add an eventhandler which can handle a 'resized' event of a window. select the drawingpanel in the Window Builder. At Properties click on Show Events , expand "component", and double-click inside the field behind "resize" (this was explained in detail in step 4).
In this eventhandler, add a call to the `resetBounds()` method of the drawingpanel.

Test the application. Do the balls bounce on the edges of the window properly, after a resize of the window?

Finish

Add a title to the application and customize the Look-and-feel (see Assignment 2). Add comments (as you learned in Assignment 1). Do this for all Java files you have created yourself. Do not forget to enter the author details in the comments at the @author field.

Have assignment checked

Give a demonstration of the assignment to the lecturer or an assistent. Requirements for a 'pass' for this assignment:

- Comments and author details have been provided with the code.
- A ball with a random color appears at a random location on the screen when clicking on the button.
- The ball moves with a constant speed and bounces against the borders of the application.
- The ball disappears when you click near or at the ball.

Summary

Today you have written Java code and learned how to design a class and to apply it to create multiple objects.
In addition, you have learned the following.

- Creating an interactive user interface.
- Applying various types of Event Handler.
- Designing a basic class.
- Using basic drawing commands in Java.