UNIVERSITY OF TWENTE.

APPLICATION DEVELOPMENT

LECTURE 5: GAME TECHNIQUES & MOTION SENSING, INHERITANCE, USERINTERFACES

```
class AppDev {
```


```
}
```

Part of SmartProducts

INTRODUCTION

APPLICATION DEVELOPMENT

Fjodor van Slooten
W241 *(Horst-wing West)*
f.vanslooten@utwente.nl

- Doing assignments & project progress
- Game techniques & motion sensing
- Class: inheritance
- Userinterfaces
- Assignment

slides @ vanslooten.com/appdev

INSERTING CODE… AT RIGHT SPOT

To get the charachter, we need to read it from the BLE connection. Add the following piece of code at the end of the loop() method to realize this:

```
if(bleserial.available()){
    char1 = bleserial.read(); // read first character of string received via BLE
}
```

```
// Get humidity event and print its value.
dht.humidity().getEvent(&event);
if (isnan(event.relative_humidity)) {
  Serial.println(F("Error reading humidity!"));
}
else {
  if ( char1 == 'h' ) {
    // first character received was a 'h'
    Serial.print(F("Humidity: "));
    Serial.print(event.relative_humidity);
    Serial.println(F("%"));
  }
  // display humidity on display:
  dtostrf(event.relative_humidity, 3, 1, buf); // print fl
  display.drawString(10, 7, buf);

  //read charactar from the BLE connection
  if (bleserial.available()) {
    char1 = bleserial.read(); // read first character of string received via BLE
  }
}
}
```

```
void setup() {

} // end of setup

void loop() {

} // end of loop
```

How to avoid this?

Use Auto format to structure your code!

> next slide

CLASS DESIGN: PRODUCT CLASS

ASSIGNMENT 4, **STEP 2**

- Complains that example in lecture was Arduino code...
- Other examples on class design in previous lectures

Arduino (.h file):

```
class Product {
private:
   String name;
   int price; // in cents
public:
   // constructor (fills in name and price)
   Product(String n, int p);
   // methods (getters)
   String getName();
   int getPrice();
}
```

Java:

```
public class Product {
   private String name;
   private int price; // in cents

   // constructor (fills in name and price)
   public Product(String n, int p);
   // methods (getters)
   public String getName() {
   }
   public int getPrice() {
   }
}
```

The constructor and getters can be automatically added via the Source menu (*Source* > *Generate* …)

ASSIGNMENT 4A

STEP 5

- **It will become difficult doing an assignment without really understanding the code…**
- For instance, you cannot do step 5 (of the Arduino variant) if you do not understand how reading button values and analog values works
- If you do not understand the code in the given example sketch, step 5 will be very hard to solve!!

ASSIGNMENT 4B

STEP 5

- **It will become difficult doing an assignment without really understanding code…**
- Example… if you read this:

`... setButtonLabel(int ..., String ...)`

This method must set the text of the Button with the given button number (first parameter of type int) using the given text (second parameter type String). The first parameter of the method is, therefore, the button number, the second the text for the button. The integer can have a value between 0 and 5 (for a maximum of 5 buttons).

```
public void setButtonLabel(int button, String text) {
}       public void setButtonLabel(int b, String t) {
        }       public void setButtonLabel(int number, String message) {
                }       public void setButtonLabel(int buttonNumber, String textButton) {
                        }       public void setButtonLabel(int kingArthur, String herMajestyTheQueen) {
                                }
```

ASSIGNMENT 4B

STEP 5

- **It will become difficult doing an assignment without really understanding code…**
- Example… if you read this:

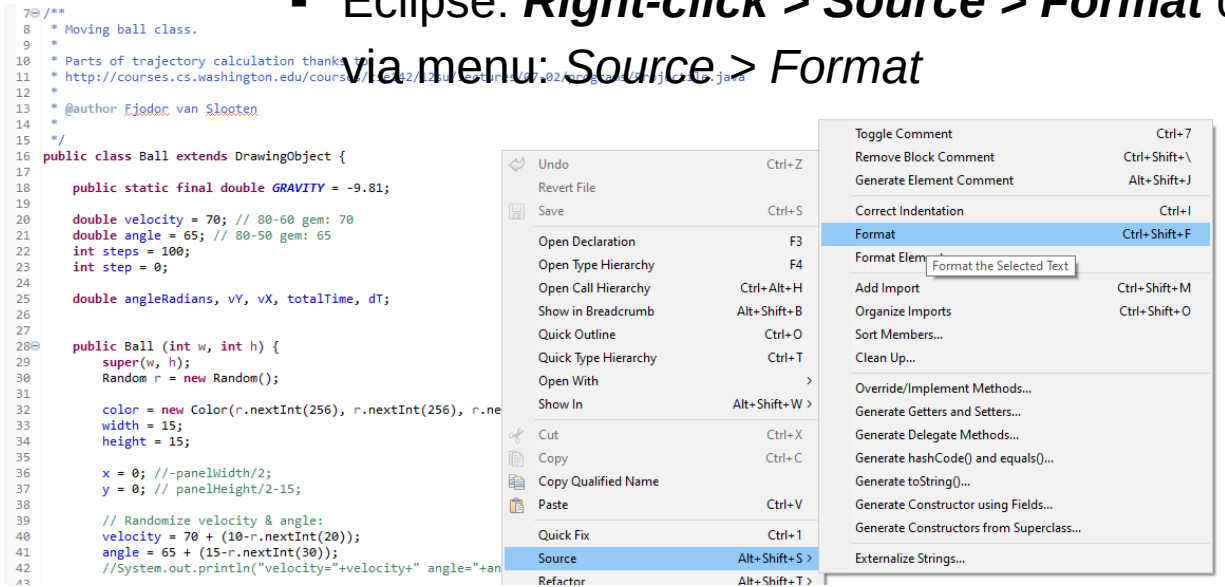
`... setButtonOutOfStock(int ...)`

Communicate that the relevant product has run out (with given button number) by setting the corresponding button to "disabled." The button will be grey and can no longer be clicked. You can use the *setEnabled()* method of the Buttons for this: call it with *setEnabled(false)*.

```
public void setButtonOutOfStock(int button) {
    if (button==0) btn1.setEnabled(false);
    else ... // if button is 1, 2, 3, ...
}
```

AUTO FORMAT

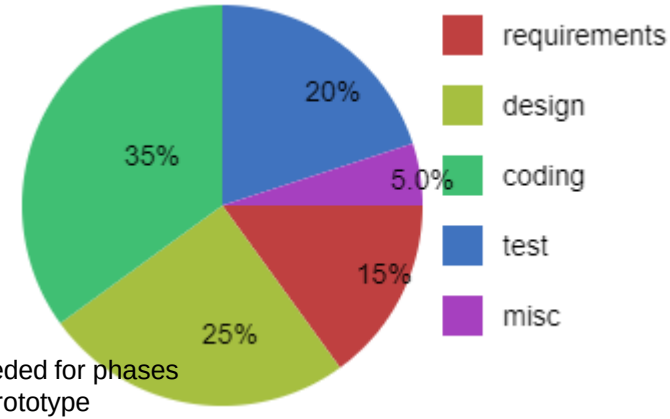
- Arduino: ***Right-click > Auto Format*** or via menu: *Tools > Auto Format*
- Eclipse: ***Right-click > Source > Format*** or via menu: *Source > Format*



PROTOTYPE FOR PROJECT

HOW DO YOU DO?

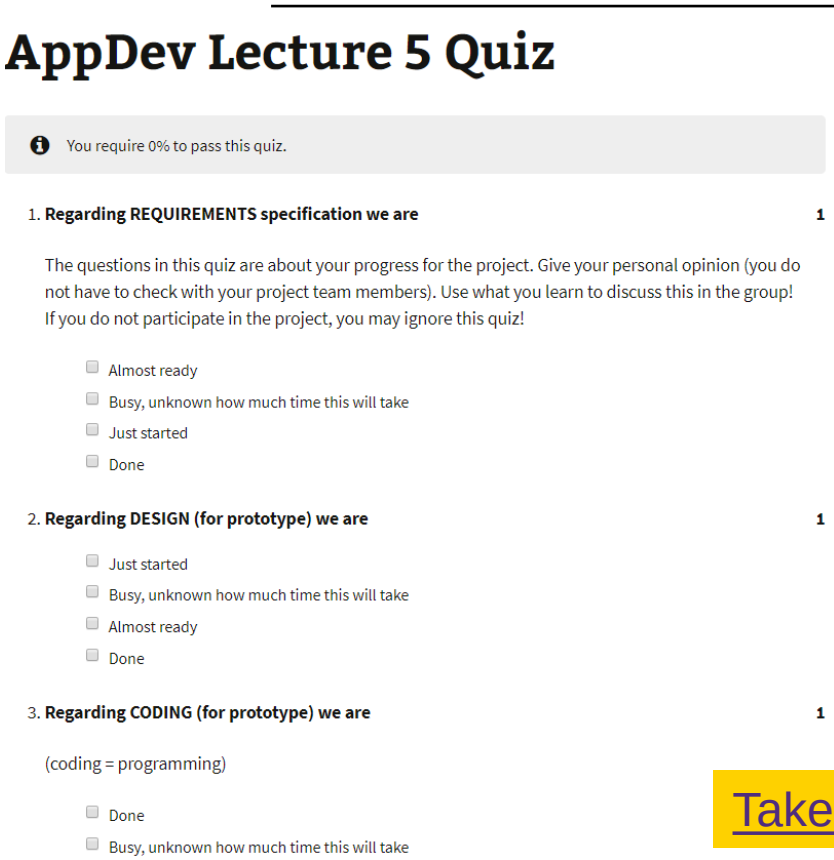
- Now we enter second half of project
- Usually, developing software (coding + testing) takes 55% of time (design + requirements = 40%)
- So to have enough time left, you should ***now*** start developing prototype



Estimation of time needed for phases in development of a prototype

DISCUSS DESIGN WITH TUTOR?

PROJECT PROGRESS



AppDev Lecture 5 Quiz

You require 0% to pass this quiz.

1. **Regarding REQUIREMENTS specification we are** 1

The questions in this quiz are about your progress for the project. Give your personal opinion (you do not have to check with your project team members). Use what you learn to discuss this in the group! If you do not participate in the project, you may ignore this quiz!

- Almost ready
- Busy, unknown how much time this will take
- Just started
- Done

2. **Regarding DESIGN (for prototype) we are** 1

- Just started
- Busy, unknown how much time this will take
- Almost ready
- Done

3. **Regarding CODING (for prototype) we are** 1

(coding = programming)

- Done
- Busy, unknown how much time this will take

- See questions in quiz with this lecture!
- Discuss outcome of quiz questions in your group & with tutor

Take this quiz now

GAME TECHNIQUES & MOTION SENSING

- Combine:
  - **ArrayList** to store game elements
  - **Timer** to draw game elements frequently (e.g. each 20ms)

- Next lecture (Arduino):
  - Use accelerometer/gyro sensor to ‘sense’ movement, use that to control the game

TIMER

Used in today's assignment

Library: **`javax.swing.Timer`**

- Declare:

```
// declare timer, e.g. as class variable:
Timer t;
```

Animation: Use timer to draw something while changing its position

- In constructor:

```
// initialize timer:
// clock ticks every 20ms; call repaint() at each clock tick:
t = new Timer(20, (e) -> repaint() );
t.start(); // start the timer
```

Method *repaint()* is called every clock tick

TIMER

```
public class DrawingPanel extends JPanel {

   private Timer t;

   public DrawingPanel() {
      // initialize timer:
      // repaint all elements at each clock tick:
      t = new Timer(20, (e) -> repaint() );
      t.start(); // start the timer
   }

   protected void paintComponent(Graphics g) {
      for (DrawingObject o : gameElements) { // for each element...
         o.paintComponent(g)
      }
   }

   public void stop() {
      timer.stop();
   }
}
```

Timer starts immediately

Call *repaint()* method at every clock tick

Method to stop timer

ARRAYLIST (REVISITED)

STORE GAME ELEMENTS

Class variable:

```
// class variable which can hold a list of game elements:
ArrayList<DrawingObject> gameElements;
```

Constructor, initialize the list (make a new empty list):

```
// create the list of game elements (which are DrawingObjects):
gameElements = new ArrayList<DrawingObject>();
```

Add a Ball:

```
public void newBall() {
    Ball b = new Ball(getWidth(), getHeight());
    gameElements.add(b); // add Ball b to the list of game elements
}
```

DRAW ALL GAME ELEMENTS

WITH A FOR-LOOP

Balls drawn over launcher… ☹

javatpoint.com/for-each-loop

- For-each loop, simple…

```
for (DrawingObject e: gameElements) {  // for-each game element
    e.paintComponent(g); // draw the element
}
```

Balls drawn behind launcher… ☺

- But what if we want to draw last elements first? (reverse order)

```
// for each element in list...
for (int i=gameElements.size()-1; i>=0; i--) { // order last > first
    gameElements.get(i).paintComponent(g);     // draw element i
}
```

GAME: RANDOM THINGS HAPPEN…

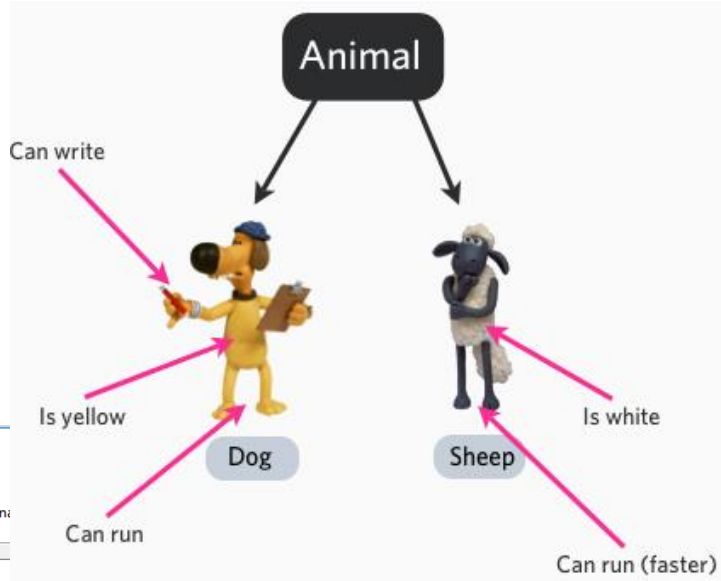
- Launch new Balls unexpectedly…

```
// class variable for Random generator:
private Random r;

// in constructor, initialize Random generator:
r = new Random();

// in method, use Random generator:
// there is a random chance of 1 in 20 on a new ball:
if (r.nextInt(20) == 1) {
    newBall();
}
```

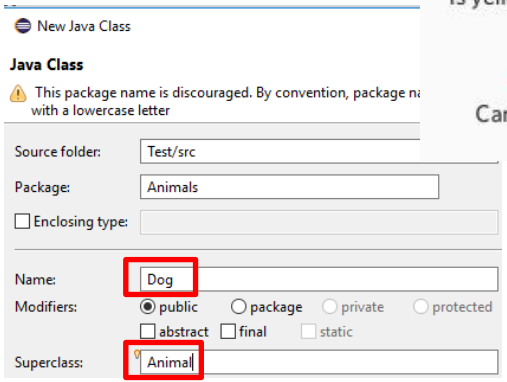
Score: 1

INHERITANCE

- New class inherits from existing
- Existing: superclass
- New: sub/derived class
- Sub is often extension (with new/other methods/properties)

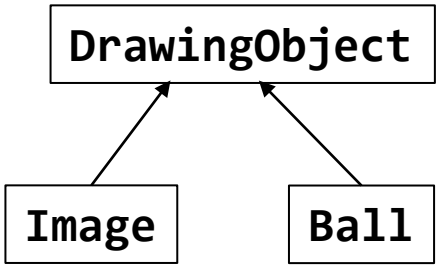



```
public class Dog extends Animal {
}
```



INHERITANCE

- ArrayList 'accepts' family members
- Who is who? **`instanceof`**



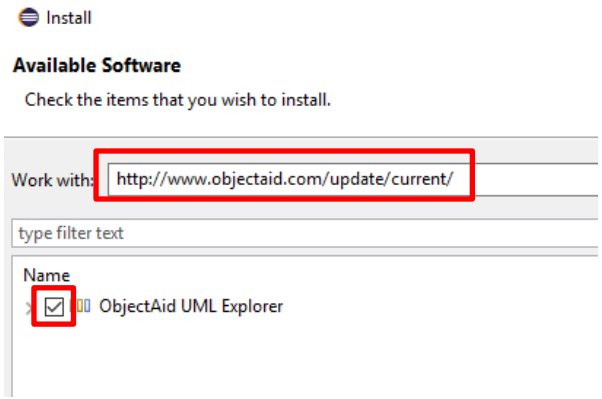
DrawingObject is superclass of Image and Ball

```
ArrayList<DrawingObject> list;

list = new ArrayList<DrawingObject>();

Image i = new Image();
Ball b = new Ball();
list.add(i);
list.add(b);

// what is item x in the list?
if ( list.get(x) instanceof Ball )
   System.out.println("It is a Ball!");
```

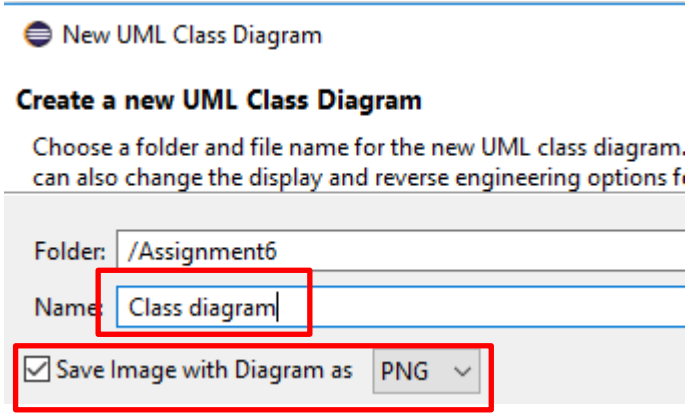
Is the element in the list of the type **Ball**? Or short: is this a **Ball**?

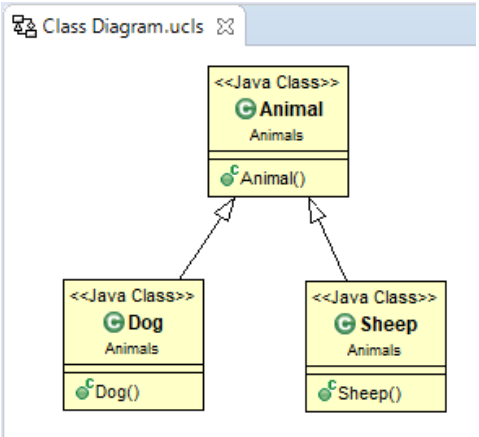
INHERITANCE: CLASS DIAGRAM

- Choose ***Help > Install New Software*** from menu
- Follow instructions as detailed here



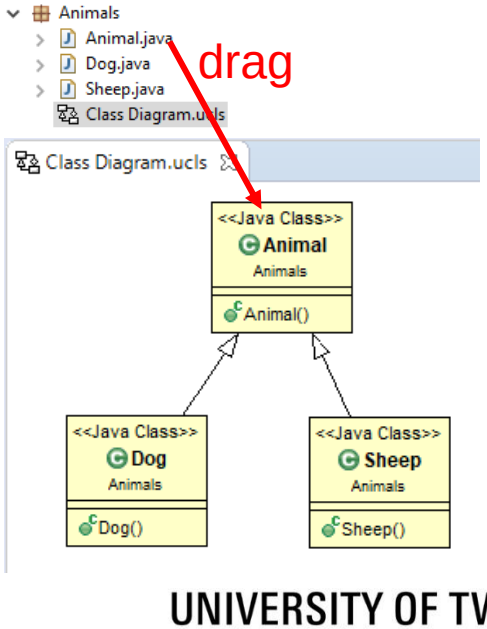
Use:
File > New > Other, choose *Object Aid UML Diagram > Class Diagram*

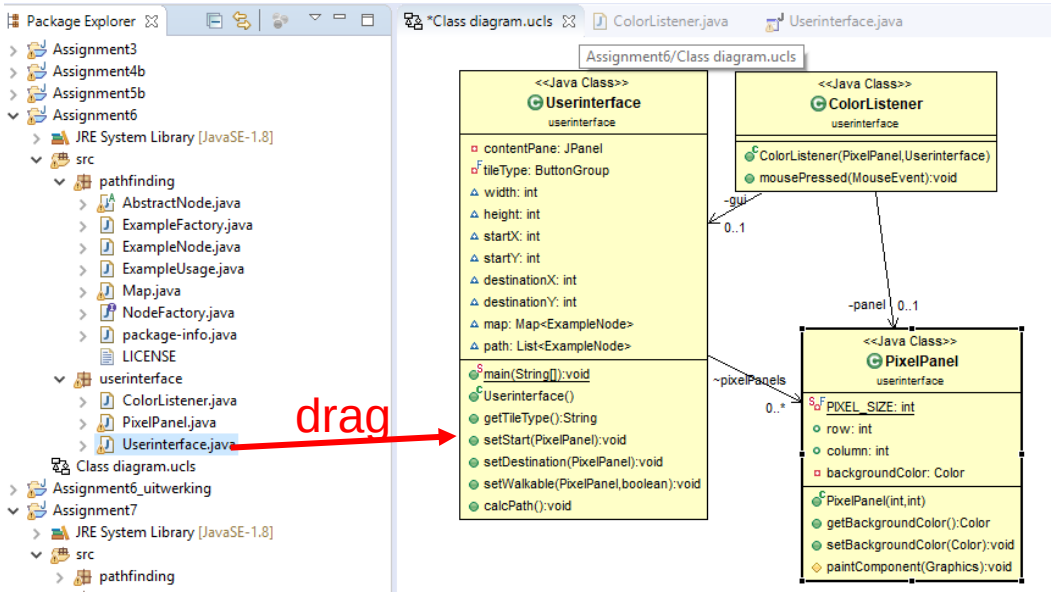




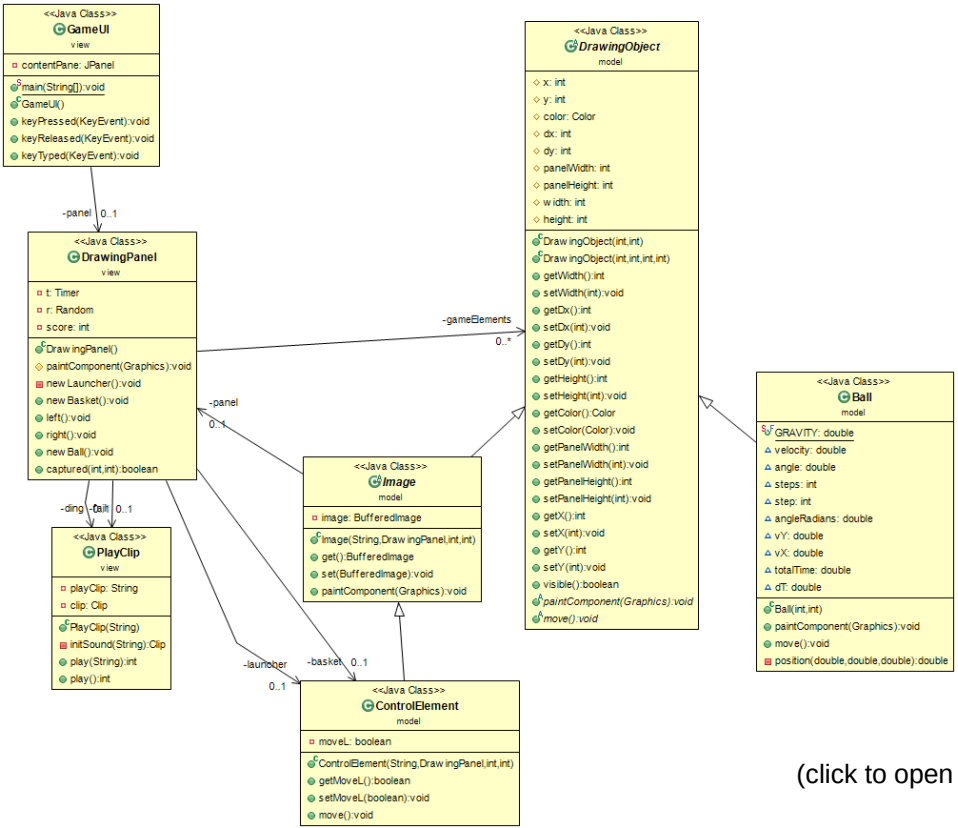
INHERITANCE: CLASS DIAGRAM

- Use: *File > New > Other*, choose *Object Aid UML Diagram > Class Diagram*
- Drag classes from package explorer into diagram





CLASS DIAGRAM



(click to open larger version)

Make class diagrams with Eclipse plugin.

File > *New* > *Other*, ObjectAid Class Diagram

Open the class diagram (its in the Package Explorer)

Drag classes (.java files) from Package Explorer into the diagram

Install the plug-in

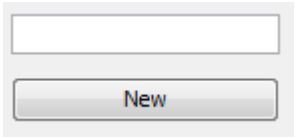
USER INTERFACES

- Add images to style UI elements
- Layout, layers
- Borders, icons
- Advanced UI elements
- Create your own UI elements

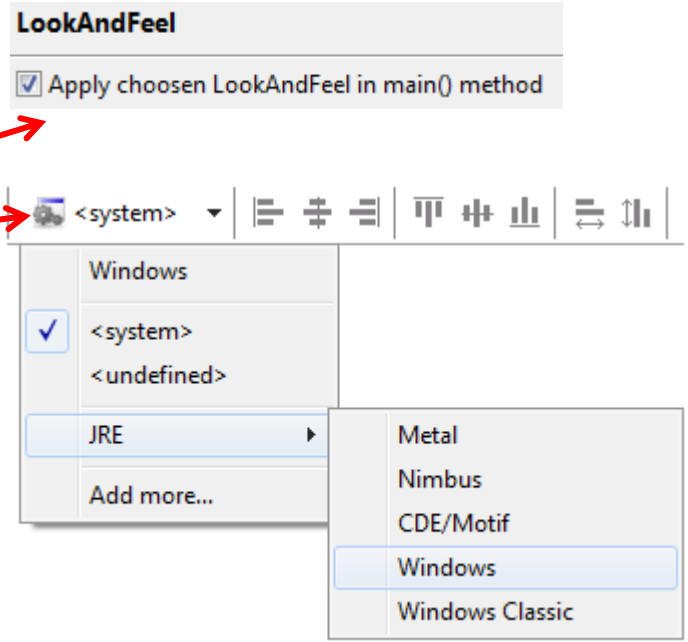
JKnob.java

Example: RoundPlayPauseButton

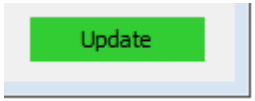
LOOK & FEEL



- Once: *Window > Preferences, WindowBuilder > Swing > LookAndFeel*
- Per application: Top of WindowBuilder



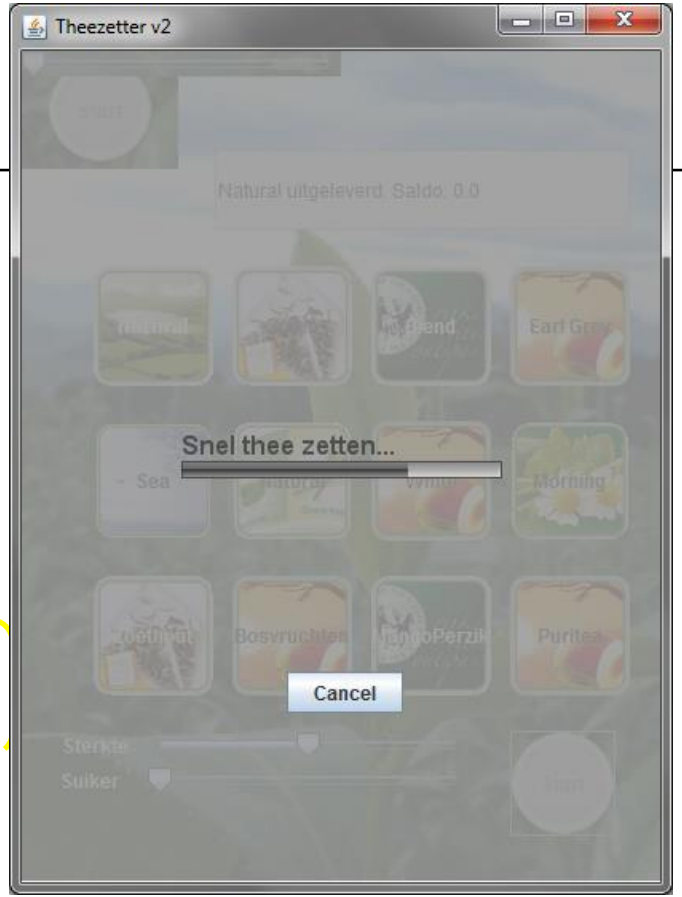
Flat-design button?:




```
JButton button = new JButton("Update");
button.setUI((ButtonUI) BasicButtonUI.createUI(button));
button.setBackground(new Color(50, 205, 50));
button.setBorder(null);
```

PANEL

- Is container
- Separate parts of UI
- Each own layout
- Turn on/off: **setVisible()**



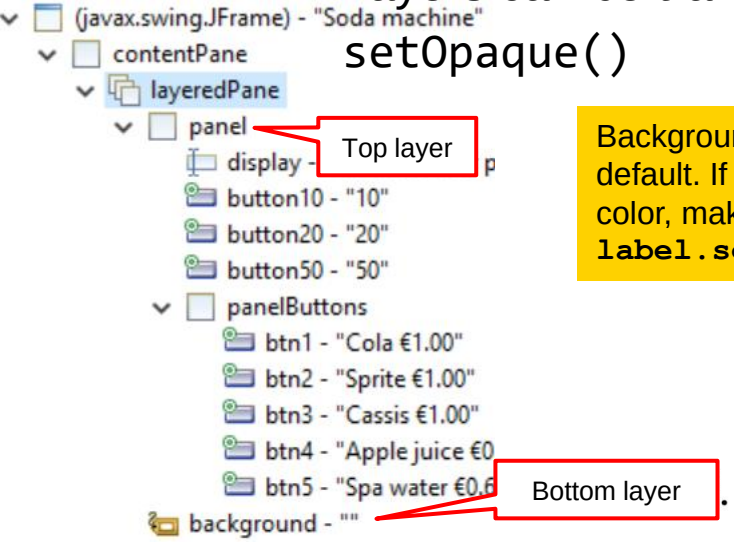
Panel with 12 buttons in *Grid Layout*

`jPanelMain`: main program
`jPanelProgress`: semi transparent progress bar

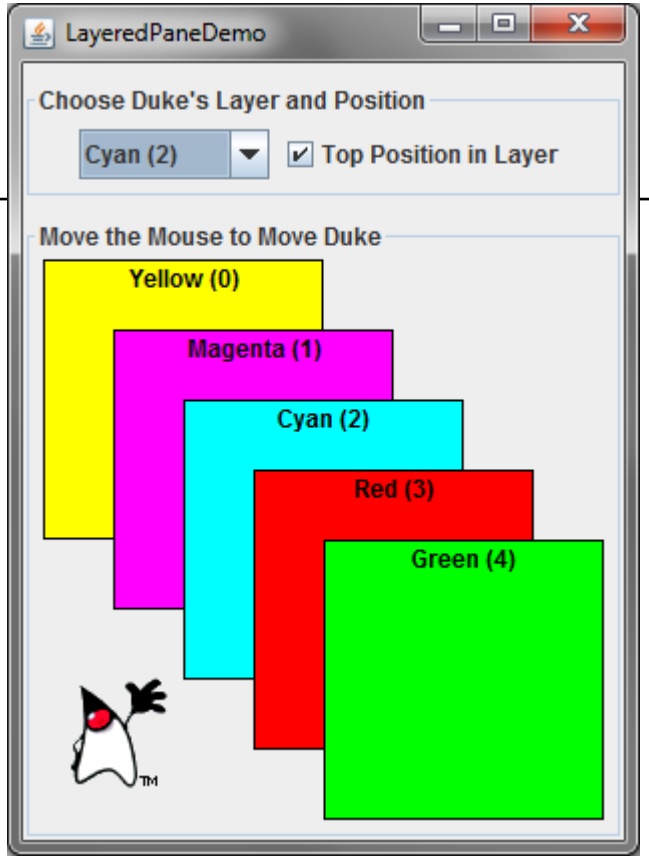
Example: Tea Maker

LAYERED PANE

- Build user interface in layers
- It is possible to turn layers on and off `setVisible()`
- Layers can be transparent and overlap `setOpaque()`



Background of a label is transparent by default. If you want to assign a background color, make it opaque first:
`label.setOpaque(true)`

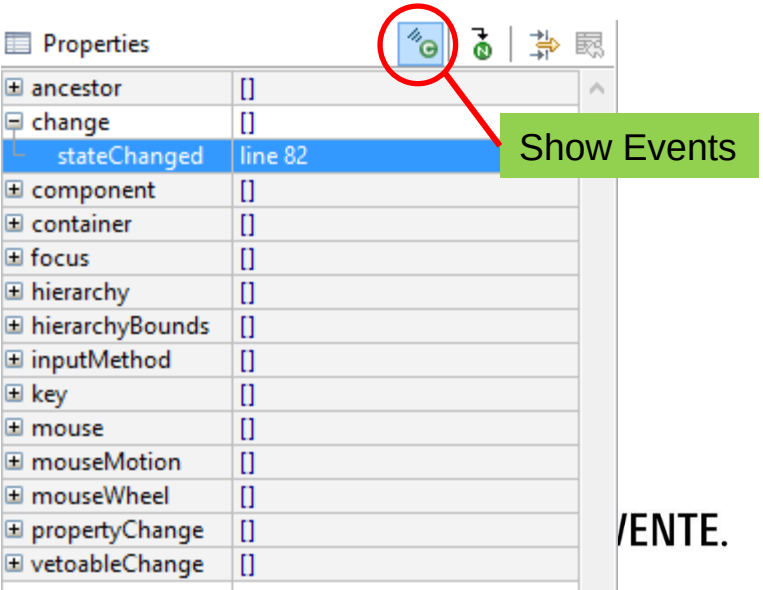


Example & demo: Assignment 5b & How to Use Layered Panes

EVENTS: LISTEN TO KEYSTROKES

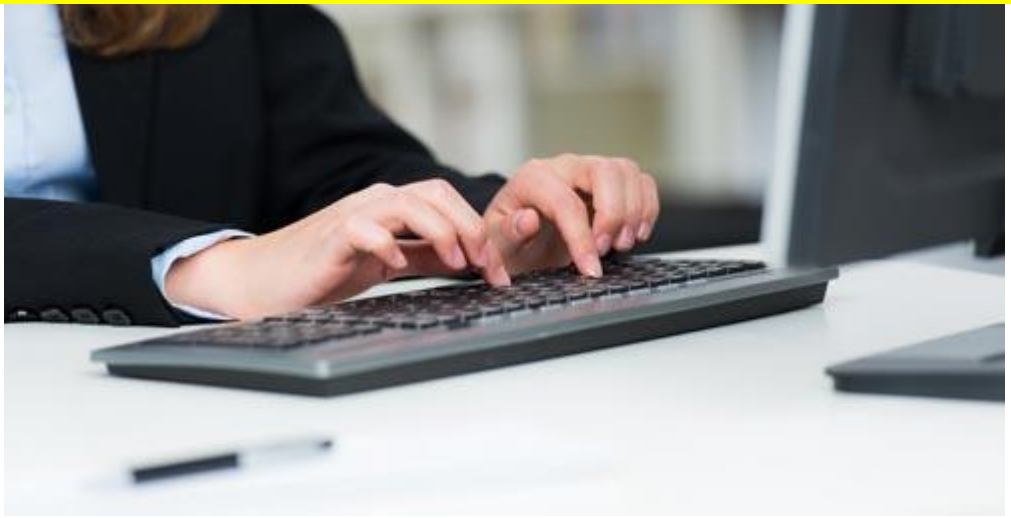
KEYS PRESSED ON KEYBOARD

- Double-click element: propertyChange
- Change tab: stateChanged




```
GameGUI extends JFrame implements KeyListener {
```

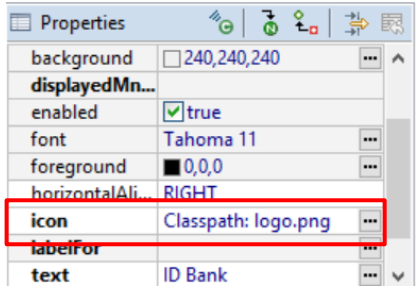
```
@Override
public void keyPressed(KeyEvent e) {
  System.out.println("key=" + e.getKeyCode() );
}
```



/ENTE.

MEDIA: IMAGES AND SOUNDS

- Images
  - Use as icon*
  - Or draw in a panel**



- Sound
  - Use class **PlayClip**, part of today's assignment

```
// read image from file:
File file = new File("images/"+filename);
try {
    image = ImageIO.read(file);
} catch (Exception e) {
    System.err.println("Unable to read "+filename);
    return;
}
// draw image:
if (image!=null)
    g.drawImage(image, x, y, panel);
```

```
PlayClip sound = new PlayClip();

// check if basket catched something:
if (checkCatched(basket.getX(), basket.getY()))
    sound.play("sound/Ding.wav");
```

LAST YEAR: WHEELED VEHICLES & LEGO

- Interested? Check out last years presentations
- Would like to use Lego for project?
  - Ask Fjodor, one project group member can pickup Lego Mindstorms set

[Lego: Build remote controlled Rover car with Arduino brain]

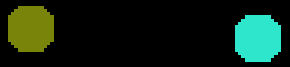
ASSIGNMENT #5

This afternoon: teacher available for help with project

- Build Catch-the-ball game

For this assignment, you will make a simple game in which balls are launched, which we then must catch. The user of the game controls a basket in which the balls must be collected. The aim is to catch as many balls as possible and earn points that way.

The basket is located at the bottom of the screen and can only move to the left and right. The balls come from a launcher which launches balls with random trajectories and speeds.



vanslooten.com/appdev

Score: 1