

UNIVERSITY OF TWENTE.

# APPLICATION DEVELOPMENT

LECTURE 6: VARIOUS TOPICS OF JAVA & ARDUINO,  
MOTION SENSING, USERINTERFACES

```
class AppDev {
```



```
}
```



Part of **SmartProducts**



# INTRODUCTION

## APPLICATION DEVELOPMENT

- Programming issues & basics
- Steps from design to code
- (Prototyping) Userinterfaces
- Arduino: motion sensing in 3d
- Assignment

slides @ [vanslooten.com/appdev](https://vanslooten.com/appdev)

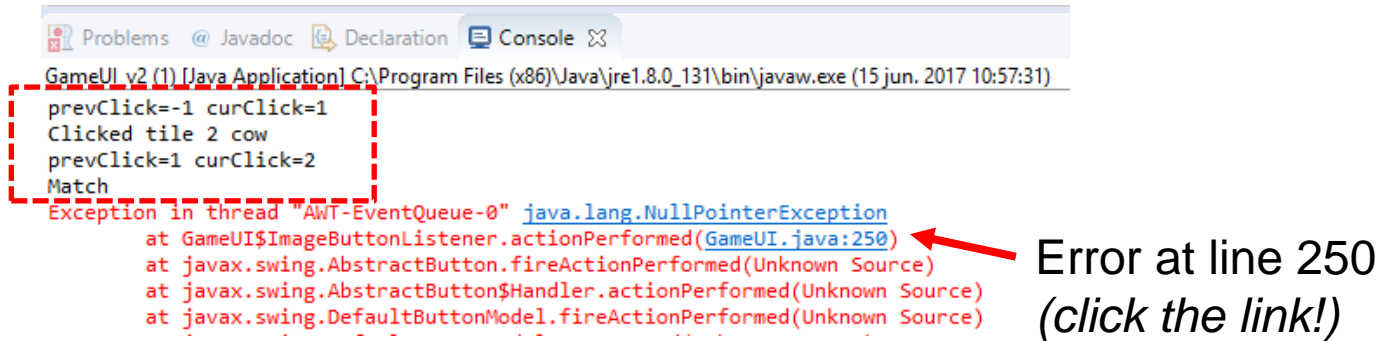
Fjodor van Slooten  
W241 (*Horst-wing West*)  
f.vanslooten@utwente.nl



No lecture next week,  
next lecture Friday June 12th

# CRASH...? APPLICATION NOT WORKING?

1. Scroll up in Console
2. Click on error (in own code) to go there



```
GameUI v2 (1) [Java Application] C:\Program Files (x86)\Java\jre1.8.0_131\bin\javaw.exe (15 jun. 2017 10:57:31)
prevClick=-1 curClick=1
Clicked tile 2 cow
prevClick=1 curClick=2
Match
Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
    at GameUI$ImageButtonListener.actionPerformed(GameUI.java:250)
    at javax.swing.AbstractButton.fireActionPerformed(Unknown Source)
    at javax.swing.AbstractButton$Handler.actionPerformed(Unknown Source)
    at javax.swing.DefaultButtonModel.fireActionPerformed(Unknown Source)
```

Error at line 250  
(click the link!)

Finding a problem: [debug](#) or use ***System.out.println()***

# VIDEO DEMO OF CIRCUIT

---

- In Loom you can enlarge webcam view:  
(e.g. to demonstrate breadboard)
- Make sure your face is in the center
- Avoid distracting noise (close window/door of your room)
- **Check the video** before you add it on Canvas (*as a media comment*)



*bit larger*

*full screen*

# LEARNING PROGRAMMING

---



- The 'ideal' way: step-by-step learning the basics, gradually increase complexity
- AppDev: learn-by-example, learn basics mixed with advanced examples ([throw in the deep method](#))
- Why??
- You must deliver a prototype in under 10 weeks
- No room in IDE programme for intro to programming
  - Solve this? Learn more about the basics by self-study (those 2 hours a week mentioned in lecture 1!)

# BASICS

- Find right spot to add code?:

- In **constructor**: can be anywhere in constructor
- At end of constructor: at last line
- Insert a **class variable**

[Test your knowledge with the quiz of this lecture](#)

```
21 public class DrawingPanel extends JPanel {  
22  
23     private ArrayList<DrawingObject> gameElements;  
24  
25     private ControlElement basket, launcher;  
26  
27     private Timer t;  
28  
29     private Random r;  
30  
31     private int score = 0;  
32  
33     private PlayClip ding, fail;  
34  
35     public DrawingPanel() { ← start of constructor  
36         super();  
37         // create a list of game elements (which are DrawingObjects):  
38         gameElements = new ArrayList<DrawingObject>();  
39  
40         // initialize timer:  
41         // repaint all elements at each clock tick:  
42         t = new Timer(20, (e) -> repaint() );  
43         t.start(); // start the timer  
44  
45         // initialize the Random generator:  
46         r = new Random();  
47  
48         // initialize sounds with the sound files:  
49         ding = new PlayClip("sound/Ding.wav");  
50         fail = new PlayClip("sound/Fail.wav");  
51     } ← end of constructor
```

# FIND RIGHT SPOT TO ADD CODE?

1. Add a method
2. Add code (in a method)

Important: format code, so you do not accidentally add code inside an other piece of code!

Scroll all the way down:

```
145 }  
146 }
```

What are these brackets??

```
145 } // end of last method  
146 } // end of class  
147
```

Add new method:

```
145 } // end of last method  
146  
147 public void aNewMethod() {  
148  
149 }  
150 } // end of class  
151
```

```
protected void paintComponent(Graphics g) {  
    // TODO Auto-generated method stub  
    super.paintComponent(g);  
  
    if (basket==null) { // if no basket is made yet, c  
        newBasket();  
        newLauncher();  
    }  
  
    // draw all elements:  
    for(DrawingObject e: gameElements) { // for-each g  
        e.paintComponent(g); // draw the element  
    }
```

[Test your knowledge with the quiz of this lecture](#)

## BASICS

### Common terms

- variable** piece of data that can hold a value
- object** represents something in real world
- class** definition of an object (like a blueprint or recipe)
- attribute** variable, part of a class, also called: field, member- or instance variable
- method** set of statements (code), part of a class

### Hello World

```
public class HelloWorld {  
  
    public static void main(String[] args) {  
        // Write line of text to console:  
        System.out.println("Hello World!");  
    }  
}
```

### Built-in types

Type	Example values	Common operators
------	----------------	------------------

int	3, 400, -200, 0	+ - / * %
-----	-----------------	-----------

boolean	true false	&&    !
---------	------------	---------

char	'a' '3', '/', '%'	
------	-------------------	--

String	"Hello" "world"	+
--------	-----------------	---

**Java Cheat Sheet**

int, boolean, char are primitive types, String is a class.

## VARIABLES AND TYPES

### Declare

```
int x;
```

### Declare & assign value

```
int x = 0;
```

### Type is an object

```
String name = "Fjodor";
```

## COMMENTS

```
/*  
Multiple lines of comments  
*/
```

```
// single-line comment
```

```
/**  
@author  
@parameter  
Java-doc style of comment  
*/
```

[More on Javadoc](#)

## CONDITIONS

### Booleans

```
boolean b = true;
```

### Boolean expression

```
10 < 3  
wave >= 100
```

## CLASS

### Definition

```
public class Person {  
}
```

## OBJECT

Declare an object *p* of type *Person*:

```
Person p;
```

### Declare & initialize

Object *p* becomes a new *Person*:

```
Person p = new Person();
```

Object *p* is an instance of the class *Person*.

## METHOD

### Definition: class with two methods

```
public class Person {  
    public int determineLength() {  
        return 89;  
    }  
    public void save() {  
    }  
}
```

## LOOPS

### For

```
for (int x=0; x<10; x++) {  
}
```

## INHERITANCE

### Definition

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

## WRITING STYLE

*variable names* start with a lowercase letter;  
*class names* start with a capital letter;  
indent-code blocks;

## CODE STRUCTURE

source file (.java)

class

method1  
statement;

method2  
statement;  
statement;

What goes in a source file?



# DIFFERENCE: METHODS & CONSTRUCTORS

- Method
- Constructor

Constructor has same name as the class and no return-type.  
It is used to initialize the object:  
*construct an object* from the class  
(the class acts as a kind of recipe).

Usually, it contains code to give class-variables a value.

```
Dog rufus = new Dog();
```

→ Constructor call:  
*construct a new Dog*

```
public class Dog {  
    String breed;  
    int age;  
    String color;  
  
    void bark() {  
    }  
  
    void run() {  
    }  
  
    void sit() {  
    }  
}
```

```
public class Dog {  
    String breed;  
    int age;  
    String color;  
  
    public Dog() {  
    }  
  
    void bark() {  
    }  
  
    void run() {  
    }  
  
    void sit() {  
    }  
}
```

# COMMENTS

---

- Line
- Multiple lines
- Javadoc

```
// single-line comment
```

```
/*  
Multiple lines of comments  
*/
```

```
/**  
 * @author Fjodor  
 * @parameter  
 * Java-doc style of comment  
 */
```

# MATH

---

- PI
- abs()
- cos(), sin()
- pow()
- etc...

Java:

```
double circumference = car_wheel_diam * Math.PI;  
unsigned int degrees = (Math.abs(distance)/circumference) * 360;
```

C++:

```
double circumference = car_wheel_diam * PI;  
unsigned int degrees = (abs(distance)/circumference) * 360;
```

[arduino.cc/en/Math/H](https://arduino.cc/en/Math/H)

Outside/device/hardware/use

Is it clear what parts the hardware will have? (sensors, actuators, interface: buttons, display, ...)

Is it clear if and how (in general) the device will communicate? (Bluetooth, Wifi, Lora, ...)

Is there a scenario for use and is it linked to the software?

Software/code

Is there a specific list of requirements for the software?

Are high-level/general functions specified/What must the software do? Is there a goal?

Is there a separation in sub-functions/modules? (are high-level functions divided into sub-functions)

Is it clear what goes in/out? (input/output/data exchanged)?

Is there a clear design on flow of the application? (decisions/conditions/loops/things that repeat)

Is there information on the platform(s) used? Eg. Java/Desktop, Arduino, ...

Is it achievable within the given time (is the project not too complex)

# FROM DESIGN TO CODE

- Finalize your design
- Add class diagram, specify functions/modules, split into sub-functions/methods
- Add flow chart
- Take a good look at outcome of milestone meeting (rubrics you received)

# FROM DESIGN TO CODE

---

- Small steps, iterate (while designing, already perform small tests)
- Test sensors, build small parts, write small test programs using examples
- Later on: put smaller parts together

Do not hesitate to  
ask for help

Making mistakes is  
the best way to learn

Rules of thumb:

- Don't try to design everything up front
- Just start (its better to start with sloppy code full of mistakes, than to postpone and wait for a better design)
- Never write more than **10 lines** of code without testing

# FROM DESIGN TO CODE: HOW TO TEST?

---

- Call a method, see result...
- Print statements!

Arduino/C++:

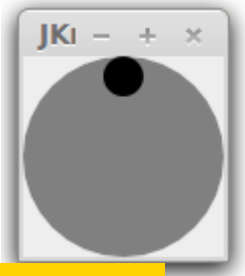
```
void setup() {  
    // test methodX:  
    object.methodX(); // what happens?  
    Serial.println("MethodX just finished");  
    // Check output of print-statements in Serial Monitor  
}  
  
void loop() {  
    // get the reading(s) from sensor  
    light = lightSensor.readRaw();  
    Serial.print("light="); Serial.println(light);  
}
```

Java:

```
public MachineUI() { // constructor  
    // test methodX:  
    object.methodX(); // what happens?  
    System.out.println("MethodX just finished");  
    // Check output of print-statements in Console  
}  
  
void read() {  
    // get the reading(s) from sensor  
    light = lightSensor.readRaw();  
    System.out.println("light="+light);  
}
```

# USER INTERFACES

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



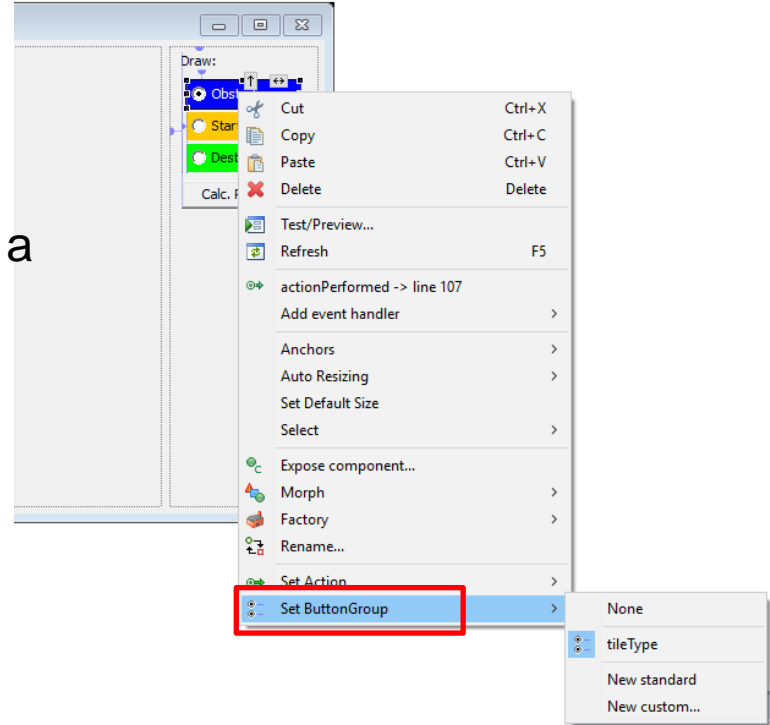
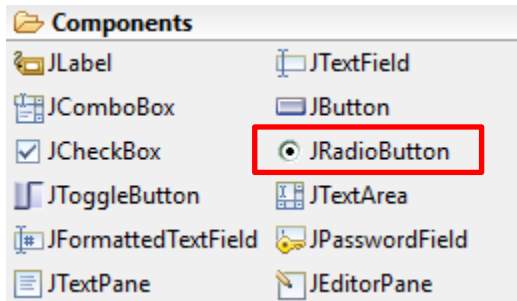
[JKnob.java](#)

Example: [RoundPlayPauseButton](#)



# RADIO BUTTONS

- Select **one** from set of buttons
- Add buttons
- Group them: by adding them to a **ButtonGroup**

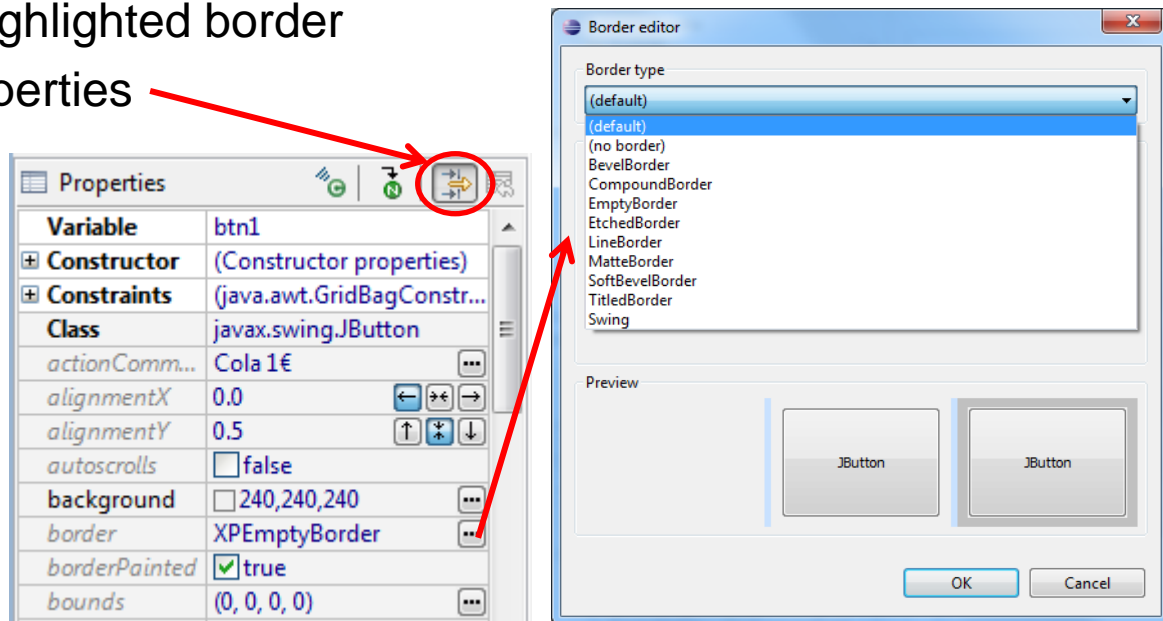




# BORDERS

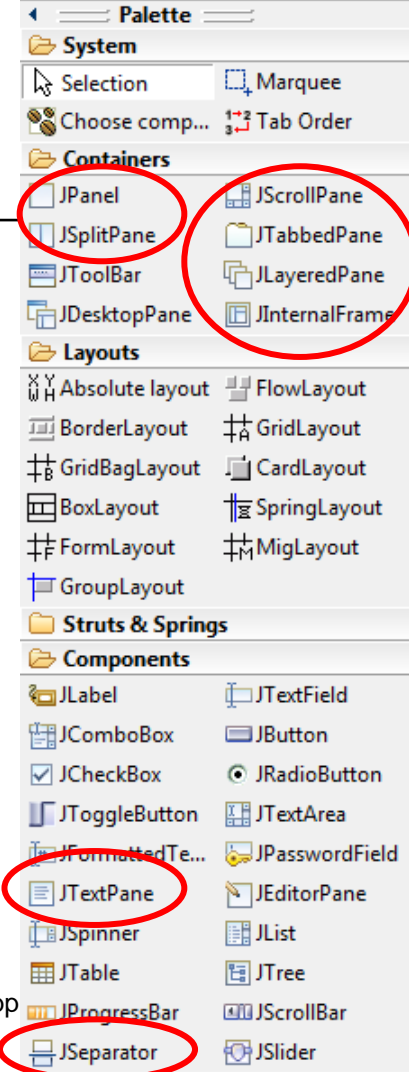
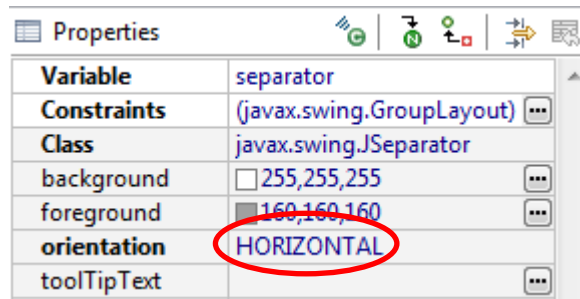
CAN BE SET ON (ALMOST) ALL UI COMPONENTS

- Assignment 6b: highlighted border
- Via advanced properties



## DIVIDE/ORGANISE PARTS OF UI

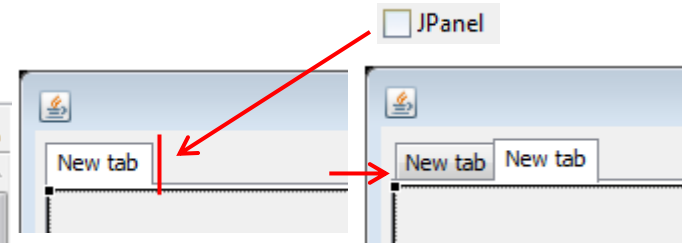
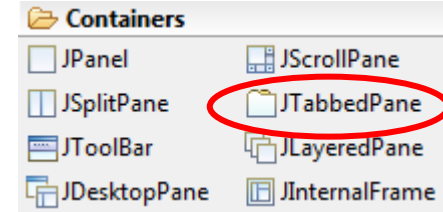
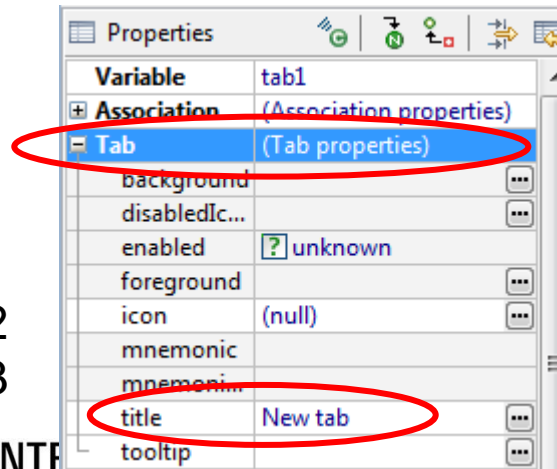
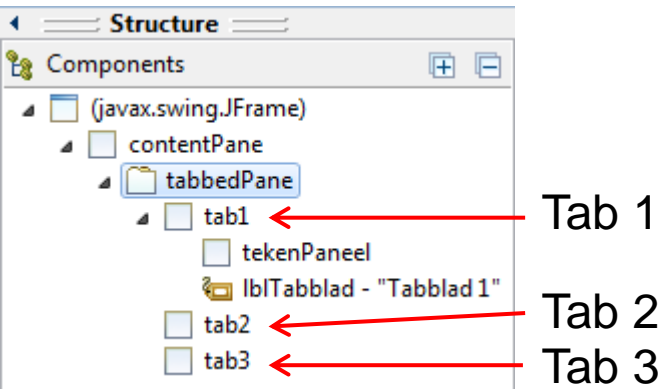
- Separators
- Tabs (Tabbed Pane)
- Split Pane
- Scroll Pane
- Layered Pane



# TABS

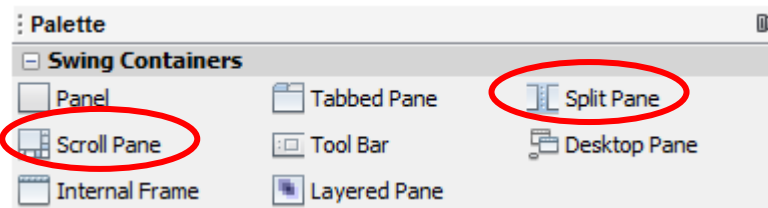
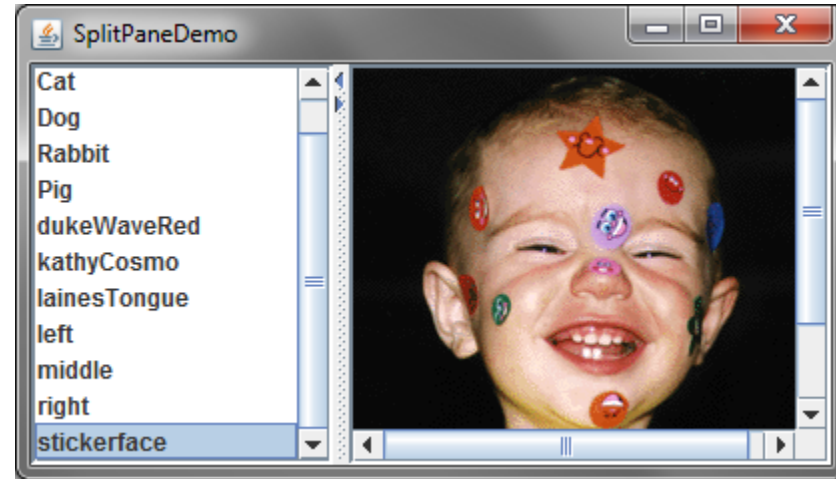
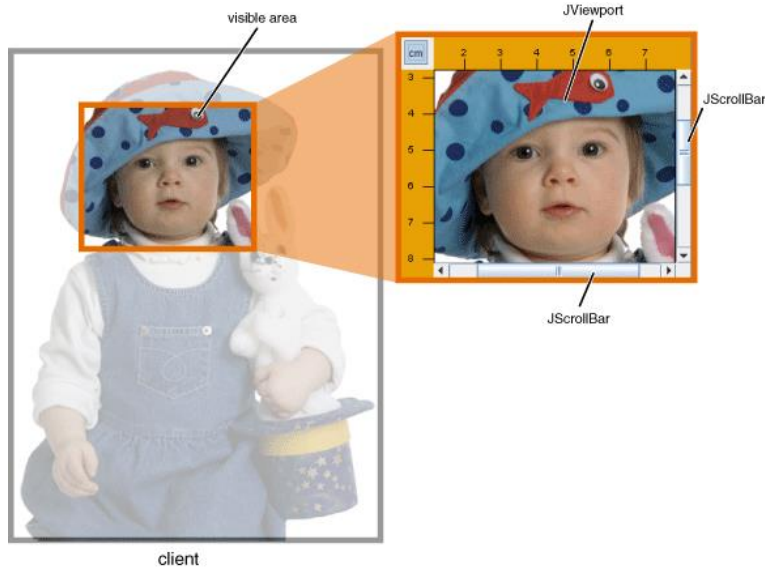
## TABBED PANE

1. Place Tabbed Pane
2. Drag a Panel onto it (becomes 1st tab) (create user interface in that panel)
3. Add more tabs:
  - Place new Panel next to tab
  - If green plus-sign appears, release



Read & demo: [How to Use Tabbed Panes](#)

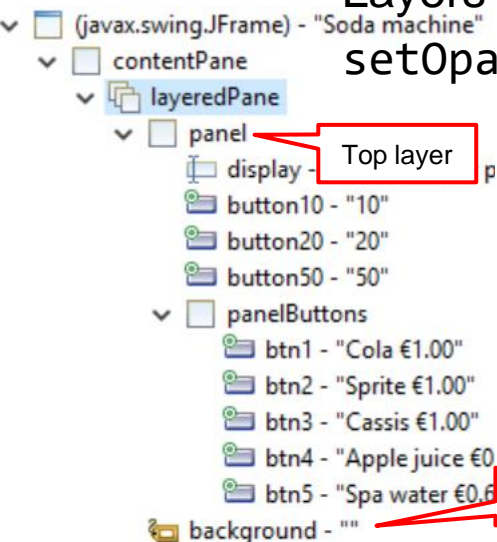
# SCROLL & SPLIT PANE



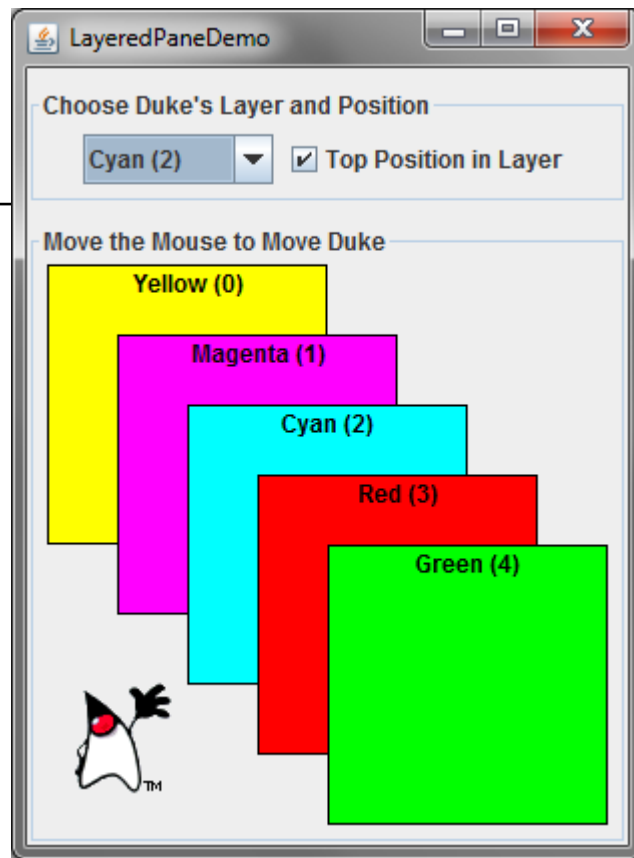
Read & demo: [How to Use Split Panes](#), [Scroll Panes](#)

# 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)`




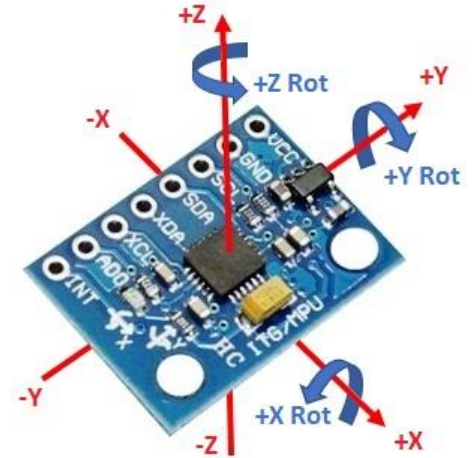
Example & demo: Assignment 5b  
& [How to Use Layered Panes](#)

# ACCELEROMETER & GYROSCOPE SENSOR

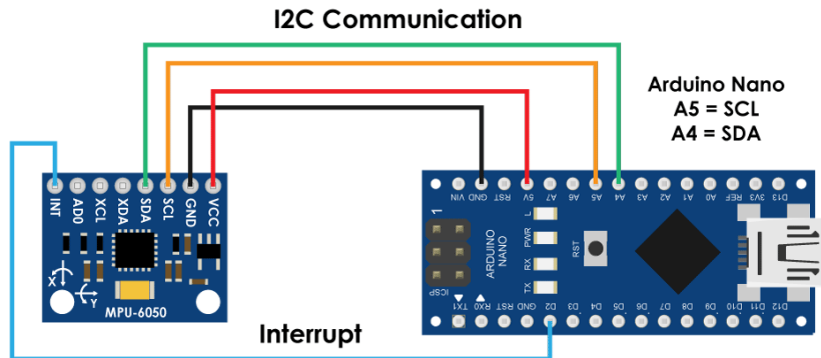
MPU-6050

Used in assignment 6

- Can detect movements (acceleration) & orientation (gyro) in 3d (x,y,z).
- You have one in your phone also...
- [Learn more about Gyroscopes](#) 

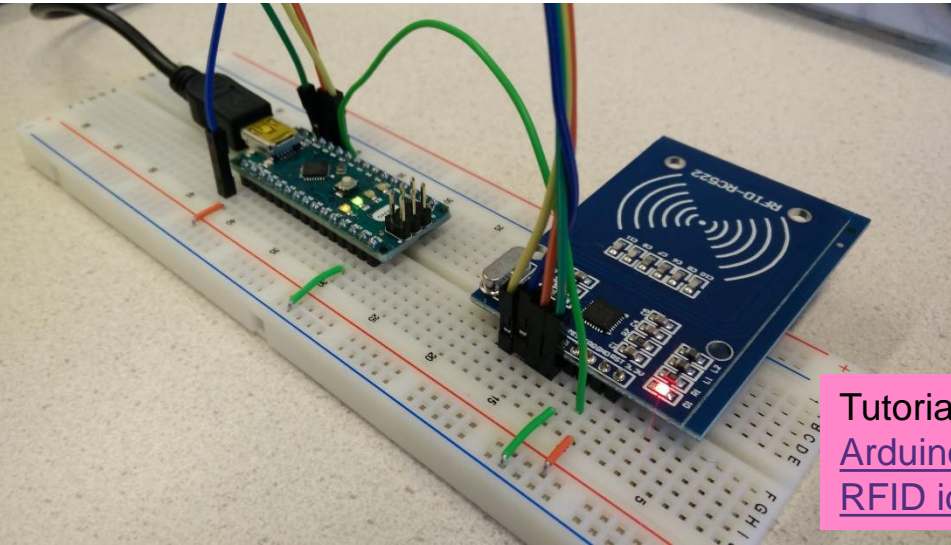
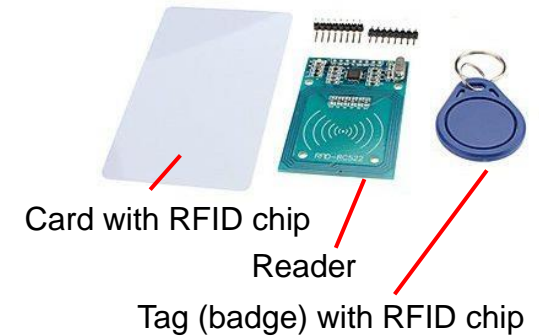


[Tutorial: intro on how it works](#)



# RFID / NFC

- RFID: Radio-Frequency Identification
- E.g. to uniquely identify a tag
- NFC: communicate with 'intelligent' tag/device



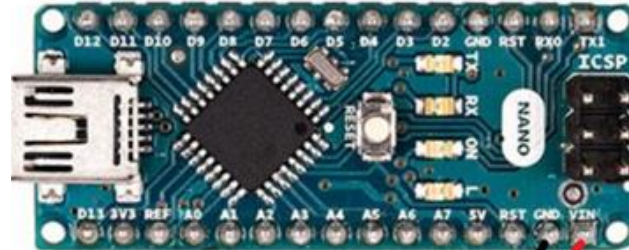
You can borrow this,  
or buy it (€4.50)

Tutorials: Build an RFID reader with Arduino and the RC522 RFID module, Add RFID identification to a Java application

# ARDUINO: BATTERY POWER

- 9V battery + battery clip<sup>€0,50</sup>, connect to GND + Vin
- Or 3x AA battery holder<sup>€1,00</sup> to GND + Vin (3 AA batteries in series)
- Example

Never connect higher voltage (>5V) to any other pin than Vin!!!  
Vin can handle up to 12V



Vin

Connecting a powerbank to the micro usb port of the Nano will not work in most cases, but you can try ([explanation, possible work-arounds](#))



# CLASS REFERENCING

---

- In a method of GameUI you can call a method (of an object) of DrawingPanel:
- But how to do it the other way around?  
(call method of GameUI from a method in DrawingPanel)

```
public void keyPressed(KeyEvent e) {  
    // remove comments from println statement below to reveal codes of keys:  
    // show keycode of pressed key:  
    // System.out.println("keyPressed " + e.getKeyCode());  
    switch (e.getKeyCode()) {  
        case 66: // b  
            panel.newBall();  
            break;  
        case 37: // left arrow  
            panel.left();  
            break;  
        case 39: // right arrow  
            panel.right();  
            break;  
    }  
}
```

# CLASS REFERENCING

PASS A REFERENCE OF AN INSTANCE OF OBJECT TO ANOTHER OBJECT

In **DrawingPanel**, we need to call a method of **GameUI**  
eg. to send score...:

```
// send score to Arduino:  
gui.send("Score: "+score);
```

1. Declare reference as a class-variable
2. Initialize it in constructor (add a new constructor!)
3. Change constructor call in GameUI (use **this**):

```
public class DrawingPanel extends JPanel {  
    GameUI gui; // reference to userinterface
```

```
public DrawingPanel(GameUI u) {  
    this(); // call base constructor  
    gui = u;  
}
```

change: `panel = new DrawingPanel();`

into: `panel = new DrawingPanel(this);`

[More info about 'this'](#)

# GRADE FOR APPDEV

---

- How is the final grade calculated for Application Development?

This is the formula: **number\_of\_passed\_assignments / 9 \* 10**

Seven assignments (1-7) and two practicals count towards the final grade. So passing this course requires at least 5 passed assignments.

- Will there be an exam?

No, this year there will be no (written) exam. The assignments and practicals are the only things that count towards the grade.

$$5 \div 9 \times 10 = 5.555555555555556$$

[More in the FAQ](#)

# PREPARE: ASSIGNMENT 7

---

- Free assignment
- You can do anything you like/wish/...
- E.g. a piece of code you must/want to write for the project
- A topic you would like to learn more about
- No inspiration? Sample assignments available
- Unsure? [Hand-in proposal on Canvas](#) before next lecture (June 12th)
- [More info here](#)

# ASSIGNMENT #6

No lecture next week,  
next lecture Friday June 12th

- Build a motion-based game controller
- Two possibilities:
  1. Use MPU-6050
  2. Stream accelerometer/gyro data from your Phone to the game (Dabble App, via Bluetooth)
- Two-way communication between game (on PC) and controller (Arduino)