UNIVERSITY OF TWENTE.

APPLICATION DEVELOPMENT

LECTURE 2: SOFTWARE DESIGN; DRAWING, VARIABLES & PROPERTIES, TYPES; MATH

```
class AppDev {
}
```

Part of SmartProducts

INTRODUCTION

APPLICATION DEVELOPMENT

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

- Software design
- Drawing, Userinterfaces
- Variables and expressions
- Math class
- Assignment 2

slides @ vanslooten.com/appdev

ABOUT ASSIGNMENT 1

Some of you had issues:

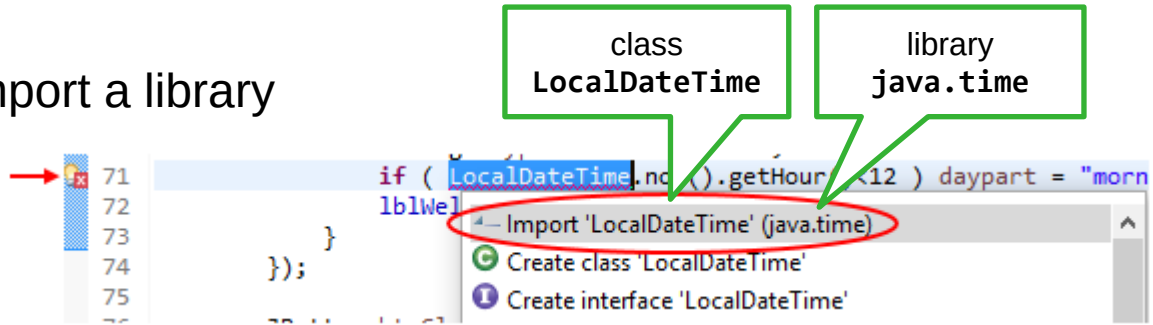
- Installing Eclipse
- Using Eclipse (workspace/projects/WindowBuilder)

- Most of these have been solved, if you still have problems, look in the forum or ask for help today

A 'golden rule': never move files or folders which are part of your Eclipse workspace: they disappear from Eclipse.

ABOUT ASSIGNMENT 1

- Import a library



- Respond to ENTER key pressed


```
// make btnOk the default button when ENTER is pressed:
getRootPane().setDefaultButton(btnOk);
```

SOFTWARE DESIGN

- Client: "Create an application that can draw one or more shapes in a user-defined color"

- Design a userinterface: *sketch* (Human Factors)
- Determine requirements

OBJECTS: WRITE A RECIPE

Analyze the world around you

- Class (describe properties and methods) and (later) specify in a class diagram
- Work out methods in pseudo-code:
  - In "plain language" write down instructions step by step

```
class Dog {
   // properties:
   int hairLength;
   int age;

   // methods:
   run();
   bark();
   sit();
}
```

Map

Objects

The Dog-class is used in this tutorial also

‘Recipe’

SOFTWARE DESIGN

External functions

- Product functions
  - Can do
  - Behaves
  - Looks
  - Is

the box

How?
What?

- Internal functions
  - Consists of
  - Working principles
  - Specs

How does it work?
What's inside (the box)?

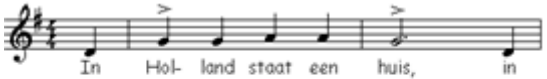
APPDEV: ROLE IN PROJECT

How does it work?
What's inside (the box)?

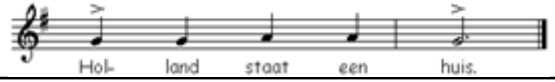
- First design iteration, answer:
  - Consists of …?
  - Working principles: how does it work/behave? → Parts, components
  - → Internal functions/behaviors
  - Specs… what type, size, color… → Properties/variables

Application (design) specifications

WHAT MAKES THEM PLAY?

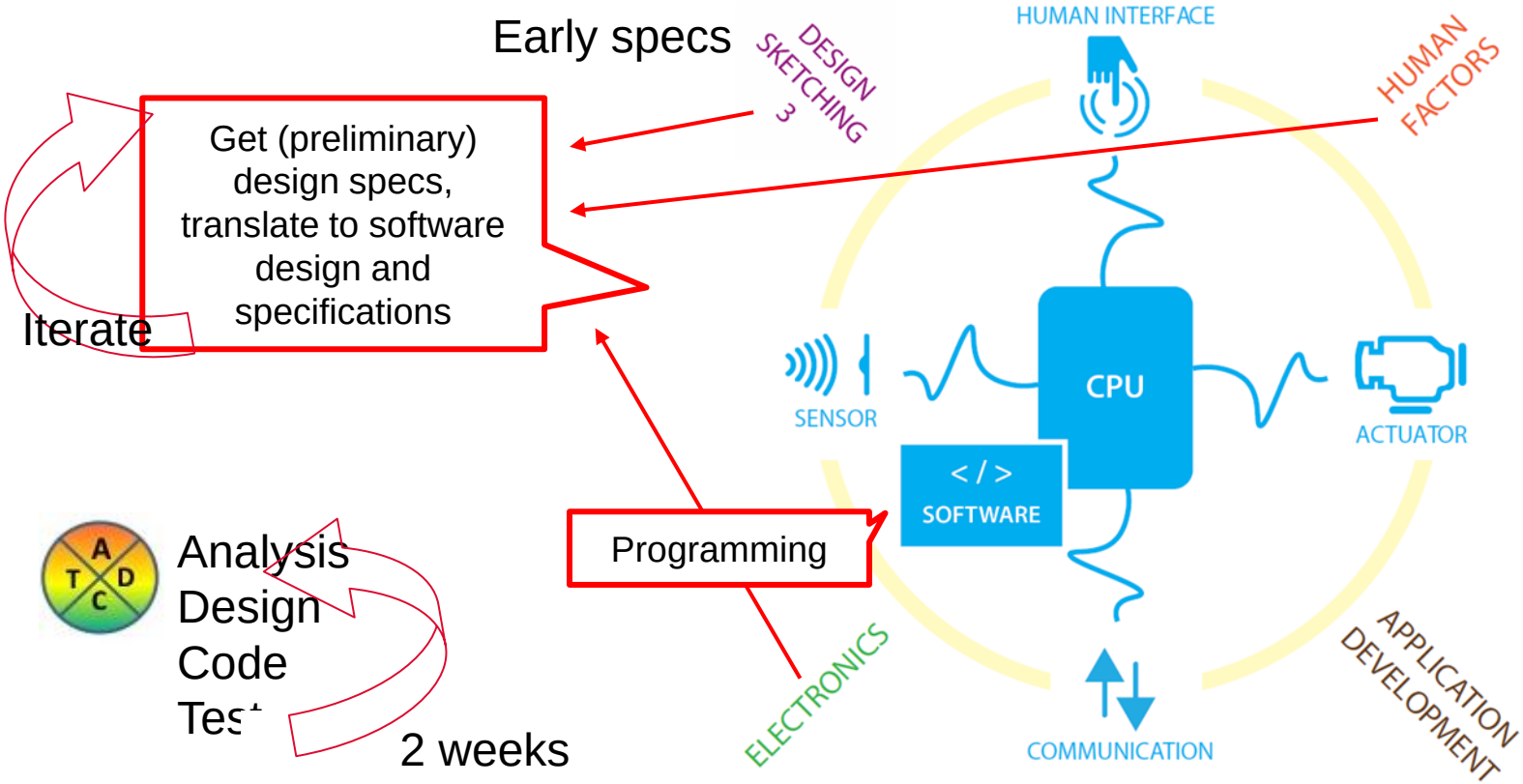


program



- (Internal functions) > components (are like orchestra-members)
- Conductor = controller; plays **piece of music** = Application (the program)

DEPENDENCIES



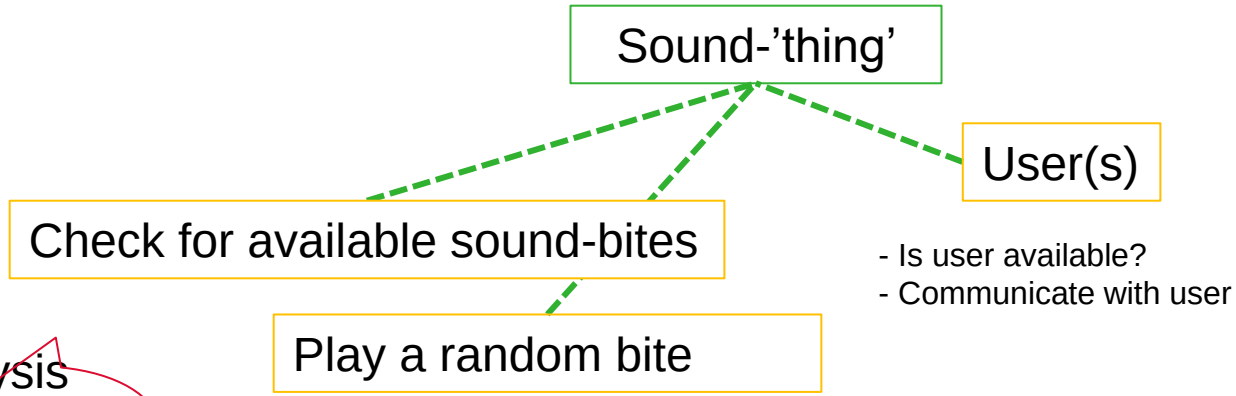
APPLICATION DESIGN SPECIFICATIONS

- Summary of the data (what the program knows/remembers)
  - Import/Export: What is entered/does the user do? What is being done? Measured?
- Describe internal functions (how it acts) and/or (forms of) behavior
  - Process: What happens to the data/input?
  - Result/Store/Communicate: Should something happen? How is result presented/delivered?
- Layout modules: the main components and connections between them

TOP > DOWN DESIGN

ROUGH, MEDIUM, FINE

- From *rough* (general/quick-n-dirty) to *fine* (detailed, objects, features, actions)



Analysis
Design
Code
Test

2 weeks

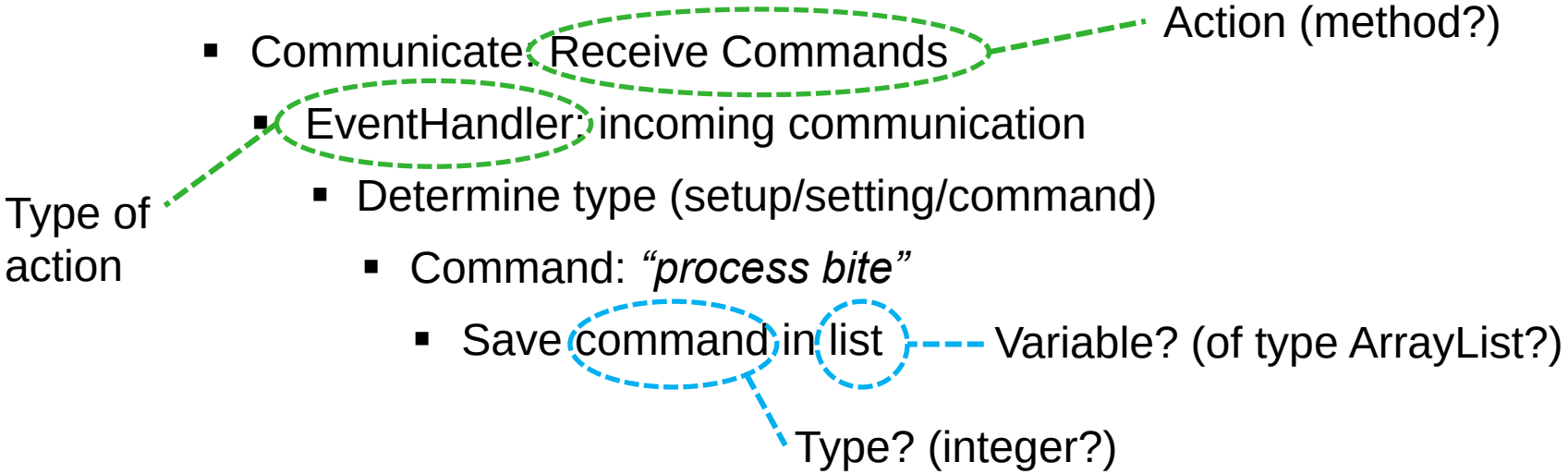
TOP > DOWN DESIGN

ROUGH, MEDIUM, FINE

- User: Algorithms / Behavior?
  - Detection position user
  - Get position of handle
  - Give feedback
  - Receive commands (from user): on/off/check/…
- Inputs & outputs
  - Position handle (x, y?)
  - Command (code/key)
  - Feedback / Status (Sound/Light/Screen/Move/…)
  - Control panel / remote
  - Display (LCD? OLED?)
  - Buttons: On / Off / Push / Point

TOP > DOWN DESIGN

ROUGH, MEDIUM, FINE

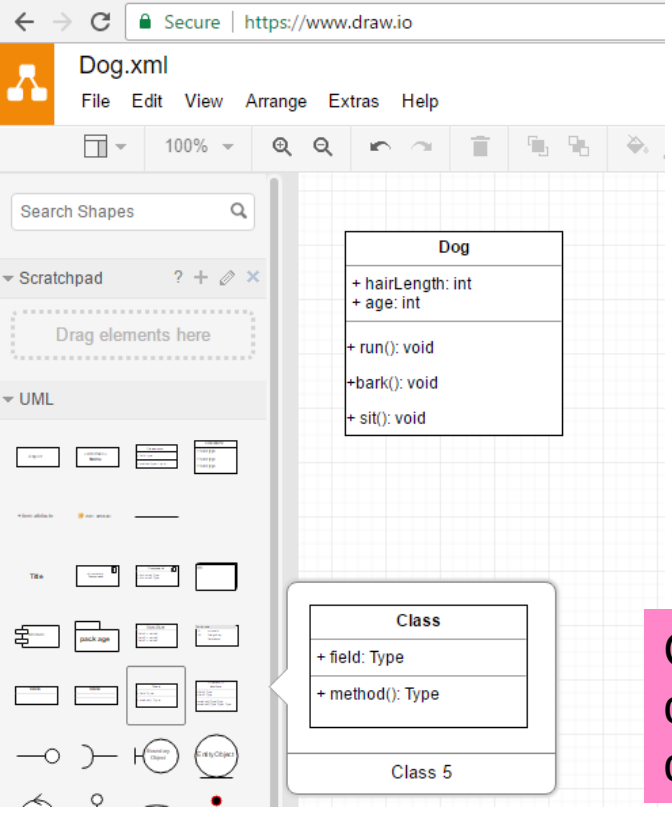


Next iteration: convert properties and methods to classes.
Detail methods in Pseudocode.

DRAW A CLASS DIAGRAM

USE DRAW.IO WEBSITE TO CREATE DIAGRAMS

Like flowcharts, class diagrams



Properties: things an object has/knows/stores

Methods: things an object can **do** (actions/behavior)

```
class Dog {
    // properties:
    int hairLength;
    int age;

    // methods:
    run();
    bark();
    sit();
}
```

Convert specs from design to class-design: class-diagram + class/methods in pseudo code

ELABORATE METHOD IN PSEUDO CODE

PSEUDO

```
// method that handles running:
run(int speed) {
 if dogs sits, stand-up (drive motors of rear legs)
 adjust power to motors dependent on speed
 turn-on motors in forward direction
}
```

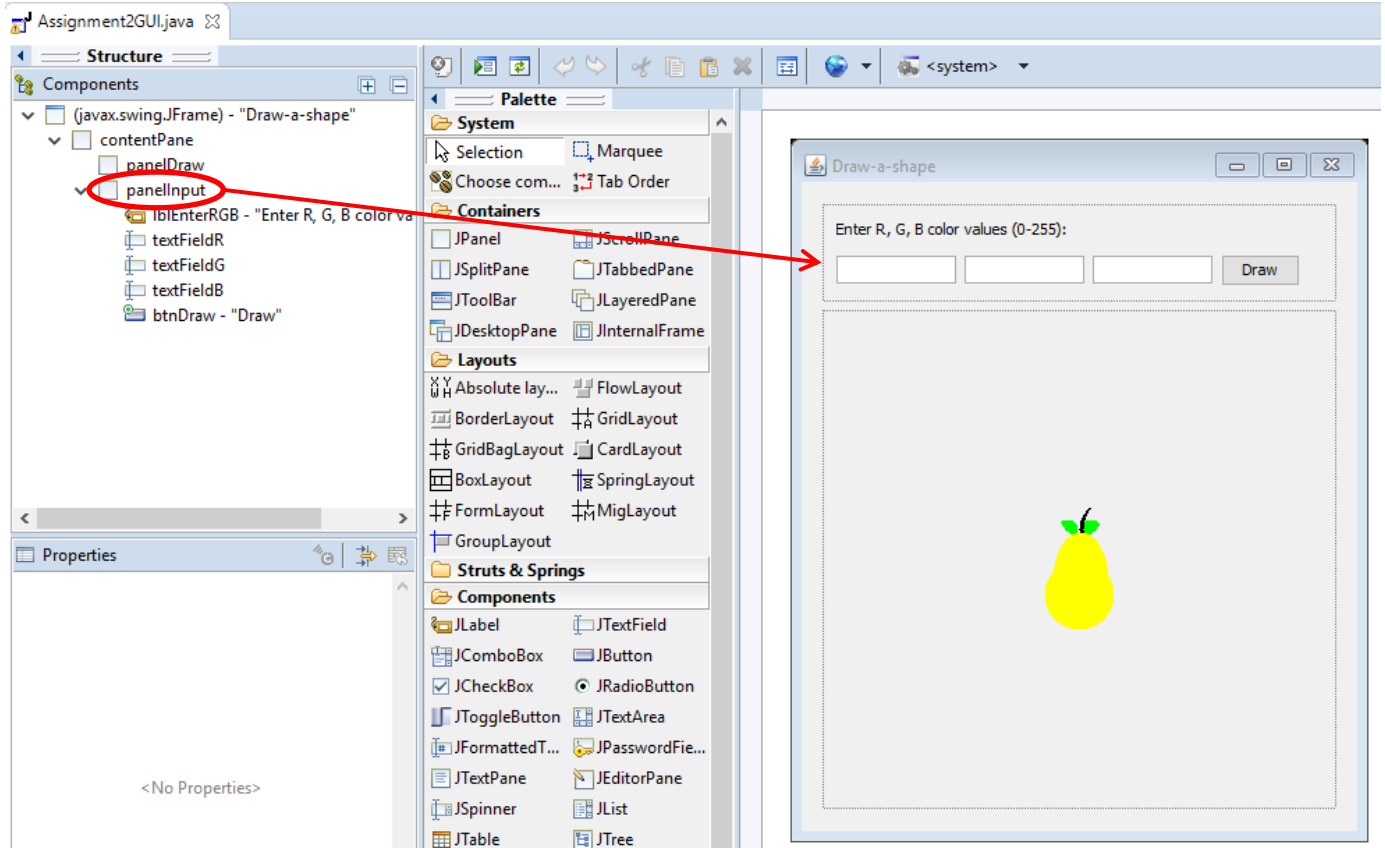
CODE:

```
// method that handles running :
public void run(int speed) {
 // if dogs sits, stand-up (drive motors of rear legs)
 if (sitting) Motor.A.rotate(60);

 // adjust power to motors dependent on speed
 Motor.A.power(speed);
 Motor.C.power(speed);

 // turn-on motors in forward direction
 Motor.A.forward();
 Motor.C.forward();
}
```

Pseudocode returns as comments

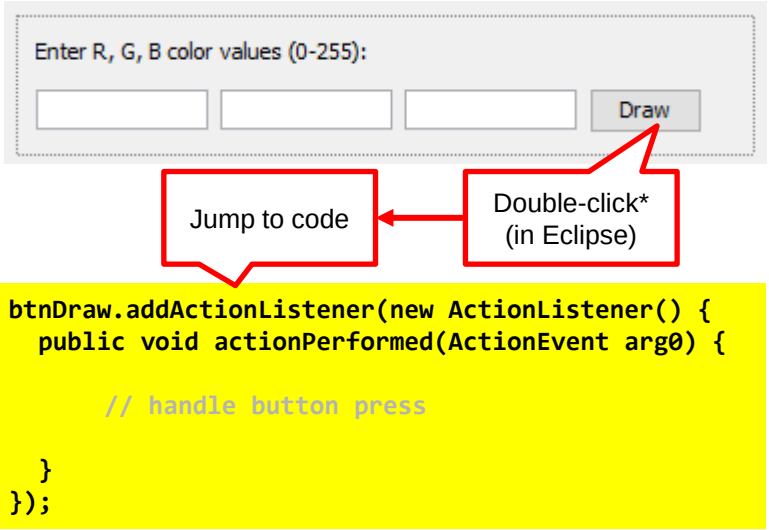
USER INTERFACES

DESIGN AN APP THAT CAN DRAW SHAPES IN SPECIFIED COLOR



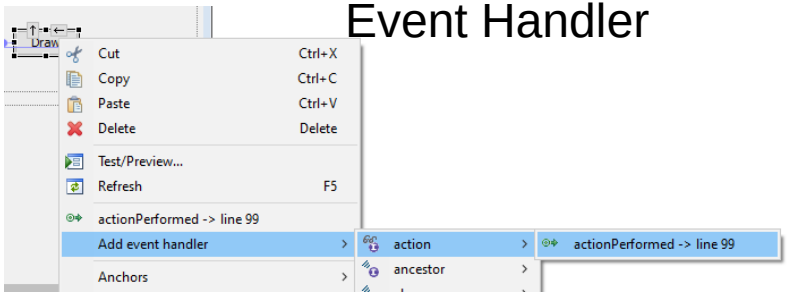
EVENT HANDLING

- What is an ‘event’?
  - Mouse-click
  - Press on a button
  - Key stroke
- Process an event?
  - Special method will handle: Event Handler

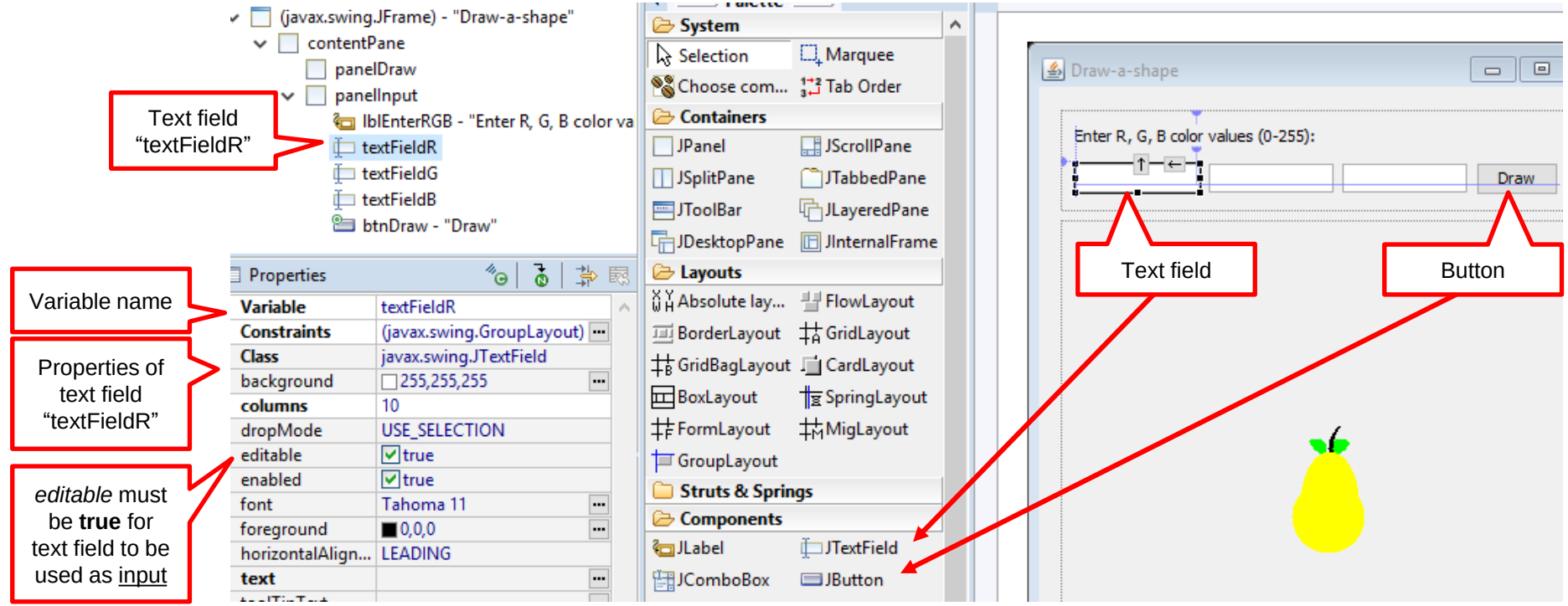



```
btnDraw.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {

      // handle button press

  }
});
```

* Or right-click:

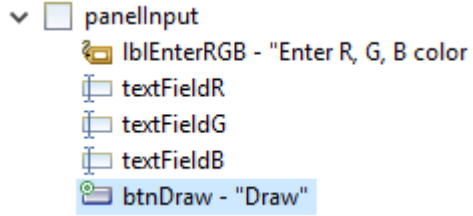


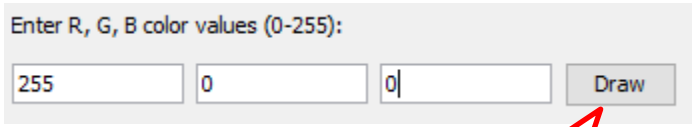
USER INTERFACE COMPONENTS



INPUT OF NUMBERS

IN A TEXT FIELD





Event handler 'actionPerformed' will be run:

If user clicks on button

```
btnDraw.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    String r = textFieldR.getText(); // read input from text field textFieldR
    // convert String r to value (integer):
    int rValue = Integer.parseInt(r);
  }
});
```

Variable of type 'String' can contain a string of characters

Convert a String (*r*) to an Integer (*rValue*)

VARIABLES

x 50 y 25

```
int x, y;
x = 20;
y = 25;
x = y * 2;
```

Declare two new variables x and y. From now, they exist.

Store the value ‘20’ in x

Store the value ‘25’ in y

The result of the expression y * 2 will now be stored in x

VARIABLES
TYPES INT AND DOUBLE

i 3 d 3.333333333333

```
int i; double d;

i = 3;
d = 3.141592653;

i = 10;
d = 10;

i = i / 3;
d = d / 3;
```

Declare two new variables `i` and `d`. From now, they exist.

Store values in `i` and `d`

Store new values in `i` and `d`

Result expressions will now be stored in `x` and `y`

EXPRESSIONS

- Expression = piece of code that delivers a value

```
double C, r = 15;

C = 2 * 3.141592653 * r;
```

Expression

A circle's circumference:

$$C = 2 \times \pi \times r$$

Operators:
+ add
- subtract
* multiply
/ divide
% modulo (remainder of division)

Evaluation is from left to right.
Priorities work the same as in Math.
You may also use brackets:
2 * (x+100)

For example 5%2 will return 1 because if you divide 5 with 2, the remainder will be 1.

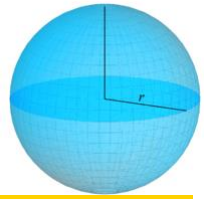
ANOTHER EXPRESSION

MATH LIBRARY

Surface area [edit]

The surface area of a sphere is:

$$A = 4\pi r^2.$$



en.wikipedia.org/wiki/Sphere

We use PI constant from Math library

pow() method from Math library

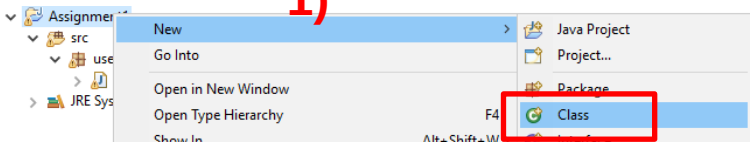
pow(x,y): x to-the-power-of y

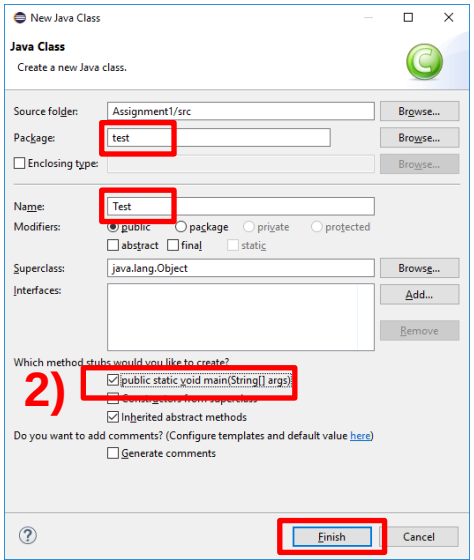
```
double A, r = 10;
A = 4 * Math.PI * Math.pow(r,2);
System.out.println( "A=" + A );
```

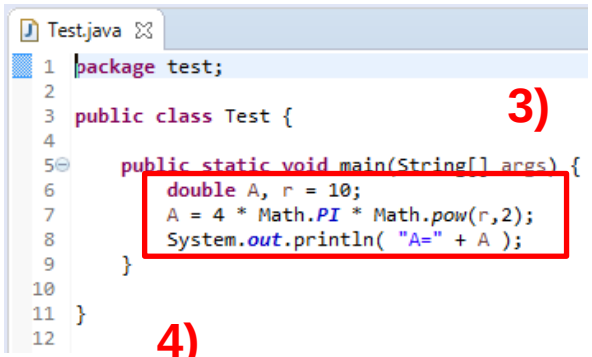
Try in Eclipse:

1. Add class to project
2. Check option “public static void main(…)“
3. Copy code inside main() method
4. Run ▶ and check Console for result

1)







UNIVERSITY OF TWENTE.

CODE OF USER INTERFACE

GENERATED BY WINDOW BUILDER

```
public class Assignment2GUI extends JFrame {

/**
 * Launch the application.
 */
  public static void main(String[] args) {
    // ...
  }

/**
 * Create the frame.
 */
  public Assignment2GUI() {

    // user interface components are created here

    JButton btnDraw = new JButton("Draw");
                                                    ...
    DrawingPanel panelDraw = new DrawingPanel();

    // ...

  }

}
```

main() method

End of **main()** method

Method (constructor) **Assignment2GUI()**

Object **btnDraw** is created here

Objects (and variables) are valid (can be used) after their creation.

Solution: move `panelDraw` up: make it a class variable (WindowBuilder calls this a 'field')

New code (eg. an eventhandler) gets inserted here… what if that code 'needs' the `panelDraw`?

CODE OF USER INTERFACE

GENERATED BY WINDOW BUILDER

panelDraw can be used in the whole class: it's scope is **global**.

Scope: region in code where a variable (or object) is valid

```
public class Assignment2GUI extends JFrame {
 DrawingPanel panelDraw;

/**
 * Launch the application.
 */
  public static void main(String[] args) {
    // ...
  }

/**
 * Create the frame.
 */
  public Assignment2GUI() {

    // user interface components are created here

    JButton btnDraw = new JButton("Draw");
                                                ...
    panelDraw = new DrawingPanel();

    // ...

  }

}
```

Object **btnDraw** has **local** scope: it can be used only inside the method (from the point where it is created)

New code (eg. an eventhandler) gets inserted here… what if that code 'needs' the panelDraw?

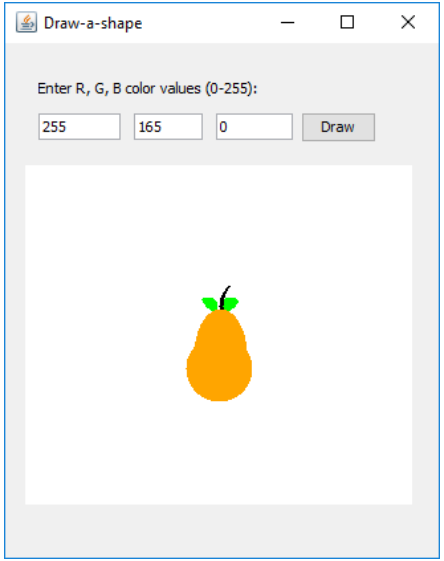
Solution: move panelDraw up: make it a class variable

ASSIGNMENT #2

Deadline of assignment 1 is today!

Deadline of each assignment is the next lecture day:
so you can get help with assignment #2 today and the next lecture day

- “Create an application that can draw one or more shapes in a user-defined color”
- Get help via chat, or outside of lecture hours, via forum
- Try examples/self-study: “Learn more” @ end of assignment



13:45h: practical session

Slides, assignments etc @ vanslooten.com/appdev