Programming Basics I

Think like a programmer, Java Syntax, Keywords, Program Flow

03.10.2016

iA

treyer@arch.ethz.ch
zuend@arch.ethz.ch
chirkin@arch.ethz.ch

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

DARCH

iA Chair of Information Architecture

Today's Structure

What's It Like To Be A Programmer?

Java Syntax: The Grammar Part

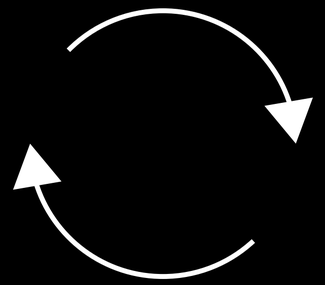
Keywords, Data Types, Operators,
Program Flow, Functions/Methods

Exercise: First Steps With Processing

Typical Workflows

Workflow In Architecture

(Define Requirements)
Brainstorming
References
Sketching
Drafting
Modeling
Discussion / Critique

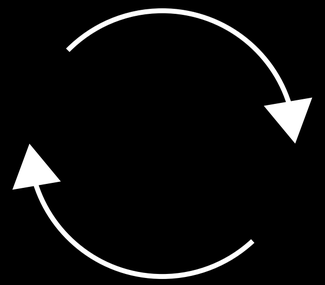


Workflow In Programming

Requirements Elicitation
Design
- Brainstorming
- Modeling (Sketching)

Implementation
- Coding
- Debugging

Validation



Peter Müller, Chair of Programming Methodology, ETH Zurich, Spring Semester 2016

https://www.ethz.ch/content/dam/ethz/special-interest/infk/chair-program-method/pm/documents/Education/Courses/SS2016/SAE/02%20-%20requirements%20elicitation.pdf

Domain Knowledge

Architecture		**Programming**
Drawing/ Modeling/ Rendering	Technique	Write Code / Handle Editor
History		Binary Operations
Ergonomics	Both disciplines require	Graphical Pipeline
Building Regulations	**domain knowledge**	Frameworks
Energy/Water Transp.	in order to design.	Object Oriented Programming
Indoor Climate / Insulation		Operating Systems
Structural Engineering		File / Database Systems
Building Process / Materials		Memory Management
....		....

Domain Knowledge

Architecture		**Programming**
Drawing/ Modeling/ Rendering	Technique	**Write Code / Handle Editor**
History Ergonomics Building Regulations Energy/Water Transp. Indoor Climate / Insulation Structural Engineering Building Process / Materials	Both disciplines require **<u>domain knowledge</u>** in order to design.	Binary Operations Graphical Pipeline **Frameworks** **Object Oriented Programming** Operating Systems File / Database Systems Memory Management ....

Java Syntax

The Grammar Part

Architecture		**Programming**
Drawing/ Modeling/ Rendering	Technique	**Write Code / Handle Editor**
History		Binary Operations
Ergonomics	Both disciplines require	Graphical Pipeline
Building Regulations	**domain knowledge**	**Frameworks**
Energy/Water Transp.	in order to design.	**Object Oriented Programming**
Indoor Climate / Insulation		Operating Systems
Structural Engineering		File / Database Systems
Building Process / Materials		Memory Management
....		....

Java Syntax

Learning Process:
The Table Metaphor

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

DARCH

iA Chair of Information Architecture

Lukas Treyer | 03.10.2016

https://chasinghollyfeld.files.wordpress.com/2012/09/bad-carpentry.jpg

Java Syntax ☞

Try it yourself

all slides having a ☞ at the upper left corner

are meant to be tried out directly in Eclipse;

Startup Eclipse and open the "Hello World" file and type in the examples as they appear.

Ideally, after this section, you should understand every word of the "Hello World" example

Java Syntax ☞

Keywords

HelloWorld.java

```
1
2 public class HelloWorld {
3     public static void main(String[] args){
4         // This is a comment
5         int a = 7;
6         System.out.println("a * 3 = "+a*3);
7     }
8 }
9
```

keywords:

`public, class, static, void, int`

code is concise, usually every word matters

even every character!

Java Syntax

Comments

Hel Open Type (⇧⌘T)

```
1
2  public class HelloWorld {
3      public static void main(String[] args){
4          // This is a comment; here I can explain something
5          // The double slashes comment out only one line
6          /* This is also a comment
7             The comment between /* and * / can spawn over
8             multiple
9             lines
10          */
11         int a = 7;
12         System.out.println("a * 3 = "+a*3);
13     }
14 }
15
```

Line: 15

comments:

```
// single line

/* multiline comments */
```

Java Syntax

Language Elements

HelloWorld.java

```java

public class HelloWorld {
    public static void main(String[] args){
        // like in any grammar theory of a language
        // we have names for the elements of the language
        int A;      // variable declaration
        int a = 7; // variable initialization
        a = 5;      // assignment operation
        a = 5 + 2; // assignment and addition operations

        // function (method) call with parameters
        System.out.println("a * 3 = "+a*3);
    }
}

```

how things are called in programming
(basically in all programming languages)

one **statement** per line
in Java every statement has to be followed by **;**
(theoretically multiple statements per line, but this looks chaotic)
program flow: line by line

a program block starts with **{** and ends with **}**

note: in java capitalisation matters. **A** is not **a**.

Java Syntax

Language Elements

HelloWorld.java

```java

public class HelloWorld {
    public static void main(String[] args){
        // like in any grammar theory of a language
        // we have names for the elements of the language
        int A;      // variable declaration
        int a = 7; // variable initialization
        a = 5;      // assignment operation
        a = 5 + 2; // assignment and addition operations

        // variable initialization; Strings are "blue"
        String calculation = "a * 3 = ";

        // function (method) call with parameters
        System.out.println(calculation+a*3);
    }
}
```

a programming language has *data types*

`int`, `float`, `double`, `String`

and *operators*

`= += -= *= /*`	*assignment*
`+ - * / ^ %`	*calculation*
`< > == != && ||`	*logical*

shortcut: using + with strings = concatenation

Java Syntax

Data Types

HelloWorld.java

```java

public class HelloWorld {
    public static void main(String[] args){
        // like in any grammar theory of a language
        // we have names for the elements of the language
        Integer A;       // variable declaration
        Integer a = 7;   // variable initialization
        a = 5;           // assignment operation
        a = 5 + 2;       // assignment and addition operations

        // variable initialization; Strings are "blue"
        String calculation = "a * 3 = ";

        // function (method) call with parameters
        System.out.println(calculation+a*3);
    }
}
```

`int` is just short for **`Integer`**

historical reasons

Java Syntax

Data Types

HelloWorld.java

```java

public class HelloWorld {
    public void sayHello(){
        System.out.println("Hello");
    }

    public static void main(String[] args){
        int a = 7;
        String calculation = "a * 3 = ";
        HelloWorld onlyDeclared;
        HelloWorld declaredAndInitialized = new HelloWorld();
        declaredAndInitialized.sayHello();
        System.out.println(calculation+a*3);

    }
}

```

define your own data types in order to reflect the structure of your data

that's what keywords **`class`** and **`new`** are used for

--> introduction later in the course

Java Syntax

Data Types

```java
public class HelloWorld {
    public static void main(String[] args){
        int a = 2;

        // boolean data type
        boolean isNot = false;
        boolean is = true;
        boolean smallerThanFive = a < 5;

        if (smallerThanFive) {
            System.out.println("a is smaller than 5");
        }
        if (true){
            System.out.println("true");
        }
        if (is){
            System.out.println("false");
        }
    }
```

`boolean` data type

stores **`true`** or **`false`**

Java Syntax ☞

Program Flow Control

HelloWorld.java

```
1
2  public class HelloWorld {
3      // function (method) call with parameters
4      public static void main(String[] args){
5          // block
6          int a = 3;
7          if (a < 5) {
8              System.out.println("a is smaller than five");
9          }
10     }
11 }
12
```

comparison operators

program flow control structures:

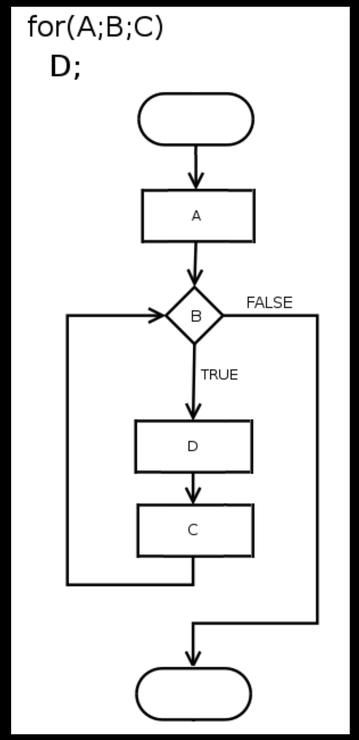
`if`, `while`, `for`

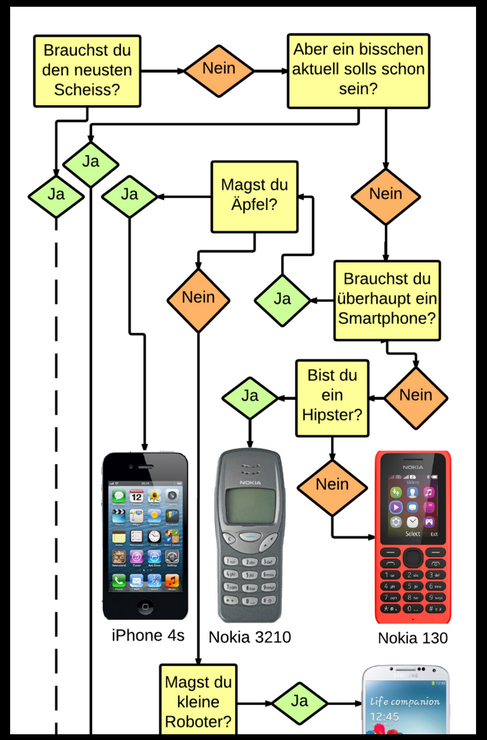
Java Syntax

Program Flow Control

HelloWorld.java

```java
public class HelloWorld {
    // function (method) call with parameters
    public static void main(String[] args){
        // block
        int a = 3;
        if (a < 5) {
            System.out.println("a is smaller than five");
        }
    }
}
```





http://www.watson.ch/Digital/Best%20of%20watson/208034098-Du-weisst-immer-noch-nicht--ob-du-das-neue-iPhone-oder-ein-Android-Handy-kaufen-sollst--Dann-lass-dir-von-uns-helfen

Java Syntax

Program Flow Control

HelloWorld.java

```java
public class HelloWorld {
    // function (method) call with parameters
    public static void main(String[] args){
        // block
        int a = 3;
        if (a < 5) {
            System.out.println("a is smaller than five");
        }
    }
}
```

control flow in other languages (Wikipedia):

Pascal:	Ada:	C:	Shell script:	Python:	Lisp:

`if a > 0 then`
`writeln("yes")`
`else`
`writeln("no");`
	`if a > 0 then`
	`Put_Line("yes");`
	`else`
	`Put_Line("no");`
	`end if;`
		`if (a > 0) {`
		`printf("yes");`
		`} else {`
		`printf("no");`
		`}`
			`if [ $a -gt 0 ]; then`
			`echo "yes"`
			`else`
			`echo "no"`
			`fi`
				`if a > 0:`
				`print "yes"`
				`else:`
				`print "no"`
					`(princ (if (plusp a) "yes" "no"))`

Java Syntax

Program Flow Control & Data Types

*HelloWorld.java

```
1
2  public class HelloWorld {
3      // function (method) call with parameters
4      public static void main(String[] args){
5          //block
6          int a = 3.5;
7          if (a < 5) {
8              // block
9              System.out.println("a is smaller than five");
10         }
11     }
12 }
13
```

data types in practise:

Integer type can hold only integral numbers

Eclipse indicates an error

Java Syntax

Program Flow Control & Data Types

*HelloWorld.java

```java

public class HelloWorld {
    // function (method) call with parameters
    public static void main(String[] args){
        //block
        double a = 3.5;
        if (a < 5) {
            // block
            System.out.println("a is smaller than five");
        }
    }
}

```

data types in practise:

in order to initialise a variable with a decimal value the variable type must be float or double

`float` = floating point number (32bit)
`double` = floating point number (64bit)

example: 3.8514E-15
0.0000000000000385140

Java Syntax

Program Flow Control & Data Types

HelloWorld.java

```java

public class HelloWorld {
    // function (method) with arguments
    public static void main(String[] args){
        // block
        double a = 3.5;
        while( a < 5) {
            // block
            System.out.println(a + " is smaller than five");
            a += 0.5;
        }
    }
}

```

repetition: **while**

Problems | Javadoc | Declaration | Console

```
<terminated> HelloWorld [Java Application] /Library/Java/JavaVi
3.5 is smaller than five
4.0 is smaller than five
4.5 is smaller than five
```

Java Syntax ☞

Program Flow Control & Data Types

*HelloWorld.java

```java

public class HelloWorld {
    // function (method) with arguments
    public static void main(String[] args){
        // block
        for(double a = 3.5; a < 5; a += 0.5) {
            // block
            System.out.println(a + " is smaller than five");
            a += 0.5;
        }
    }
}

```

repetition: `for`

Problems | @ Javadoc | Declaration | Console

```
<terminated> HelloWorld [Java Application] /Library/Java/JavaVi
3.5 is smaller than five
4.0 is smaller than five
4.5 is smaller than five
```

Java Syntax

Program Flow Control & Data Types

HelloWorld.java

```java

public class HelloWorld {
    // function (method) with arguments
    public static void main(String[] args){
        // block
        double a = 3.5;
        while( a < 5) {
            // block
            System.out.println(a + " is smaller than five");
            a += 0.5;
        }
    }
}

```

while

*HelloWorld.java

```java

public class HelloWorld {
    // function (method) with arguments
    public static void main(String[] args){
        // block
        for(double a = 3.5; a < 5; a += 0.5) {
            // block
            System.out.println(a + " is smaller than five");
            a += 0.5;
        }
    }
}

```

for

Java Syntax

Calculation Operators

*HelloWorld.java

```
1
2  public class HelloWorld {
3      public static void main(String[] args){
4          int a = 2;
5          int b = 32;
6          System.out.println("A 32bit number "
7                  + "has a maximum value of "+(a^b));
8      }
9  }
10
```

power operator ^

$2^{32} = ?$

Java Syntax ☞

Calculation Operators

HelloWorld.java

```java

public class HelloWorld {
    public static void main(String[] args){
        // by convention indices are called "i" or "j"
        // or any other single letter
        for (int i = 0; i < 13; i += 1){
            if (i % 2 == 0) {
                System.out.println(i+" is an even number");
            }
        }
    }
}

```

module operator %

n° 1 trick to get a list of even numbers

Java Syntax

Increment Operators

HelloWorld.java

```java

public class HelloWorld {
    public static void main(String[] args){
        // by convention indices are called "i" or "j"
        // or any other single letter
        for (int i = 0, j = 10; i < 13 && j > 5; i++, j--){
            System.out.println(i+j);
            System.out.println(i+" "+j);
            System.out.println("");
        }
    }
}

```

increment operator ++
decrement operator --

Java Syntax

Logical Operators

```
1
2  public class HelloWorld {
3      public static void main(String[] args){
4          // logical operators
5          if (true && true){
6              System.out.println("true && true");
7          }
8  Dead code.f (true && false) {
9              System.out.println("true && false");
10         }
11         if (true || false) {
12             System.out.println("true || false");
13         }
14         if (false || false) {
15             System.out.println("false || false");
16         }
17     }
18 }
```

combining conditional expressions

`&&` logical AND

`||` logical OR

Java Syntax

Logical Operators

```java
public class HelloWorld {
    public static void main(String[] args){
        int a = 2;
        int b = 4;

        if (a < b){
            System.out.println("a is smaller than b");
        }
        if (a > b){
            System.out.println("a is larger than b");
        }
        if (a == b){
            System.out.println("a is equal to b");
        }
        if (a != b){
            System.out.println("a is not equal to b");
        }
    }
}
```

operators for comparison

< smaller than

> larger than

== equal to

!= not equal to

Java Syntax

Logical Operators

*HelloWorld.java

HelloWorld/src/HelloWorld.java

```
2  public class HelloWorld {
3      public static void main(String[] args){
4          if (!false){
5              System.out.println("well, at least it not false");
6          }
7      }
8  }
9
```

the logical NOT operator

!

Java Syntax

Logical Operators

*HelloWorld.java

HelloWorld/src/HelloWorld.java

```
2  public class HelloWorld {
3      public static void main(String[] args){
4          if (!false){
5              System.out.println("well, at least it not false");
6          }
7      }
8  }
9
```

the logical NOT operator

!

Java Syntax

Logical Operators

HelloWorld.java

```java

public class HelloWorld {
    public static void main(String[] args){
        int a = 6;
        if (a < 5){
            System.out.println(a+" is smaller than 5");
        } else {
            System.out.println(a+" is either equal or larger than 5");
        }
    }
}

```

`else`

only valid when there is an **`if`** above

Java Syntax

Two more things to go for today

*HelloWorld.java

HelloWorld/src/HelloWorld.java

```
2  public class HelloWorld {
3      public static void main(String[] args){
4          if (!false){
5              System.out.println("well, at least it not false");
6          }
7      }
8  }
9
```

static

[]

Java Syntax

Calling a code block

- introduction to `class` later
- used in today's exercise

HelloWorld.java

```java

public class HelloWorld {
    public static void sayHelloKevin(){
        System.out.println("Hello Kevin");
    }

    public static void main(String[] args){
        double a = 3.5;
        while (a < 5) {
            sayHelloKevin();
            a += 0.5;
        }
    }
}

```

static

HelloWorld.java

```java

public class HelloWorld {
    public void sayHello(){
        System.out.println("Hello");
    }

    public static void main(String[] args){
        int a = 7;
        String calculation = "a * 3 = ";
        HelloWorld onlyDeclared;
        HelloWorld declaredAndInitialized = new HelloWorld();
        declaredAndInitialized.sayHello();
        System.out.println(calculation+a*3);

    }
}

```

"non static"

Java Syntax

Calling a code block

HelloWorld.java

```java

public class HelloWorld {
    public static void sayHelloKevin(){
        System.out.println("Hello Kevin");
    }

    public static void main(String[] args){
        double a = 3.5;
        while (a < 5) {
            sayHelloKevin();
            a += 0.5;
        }
    }
}

```

note that the program flow is not line by line anymore! (only within one code block it is line by line)

In Java every program starts with the main method. That's the program entry point - even if it is at the very bottom of your code.

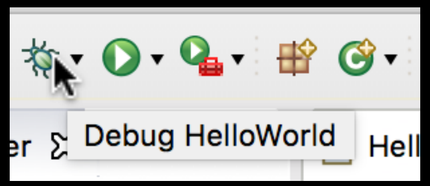
Java Syntax ☞

Eclipse Debugging Mode

HelloWorld.java

```
1
2  public class HelloWorld {
3      public static void sayHelloKevin(){
4          System.out.println("Hello Kevin");
5      }
6
7      public static void main(String[] args){
8          double a = 3.5;
9          while (a < 5) {
10             sayHelloKevin();
11             a += 0.5;
12         }
13     }
14 }
15
```

1. set a break point
2. start the program in debugging mode



3. Follow the program flow

Java Syntax

Return

HelloWorld.java

```java
public class HelloWorld {
    public static String say(){
        return "I said I'm hungry!";
    }

    public static void main(String[] args){
        double a = 3.5;
        while(a < 5) {
            String whatSheSays = say();
            System.out.println(whatSheSays);
            a += 0.5;
        }
    }
}
```

many languages have a return statement

it terminates your function block and returns a value at the place where it was called

```
<terminated> HelloWorld [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_1
I said I'm hungry!
I said I'm hungry!
I said I'm hungry!
```

Java Syntax

Eclipse: Toggle Comment

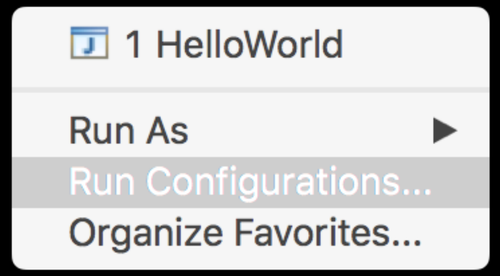
```
    public void draw() {
        a();
        b();
        c();
        d();
        e();
        f();
        g();
        h();
        demo();
    public void draw() {
//      a();
//      b();
//      c();
//      d();
//      e();
//      f();
//      g();
//      h();
//      demo();
    }
```

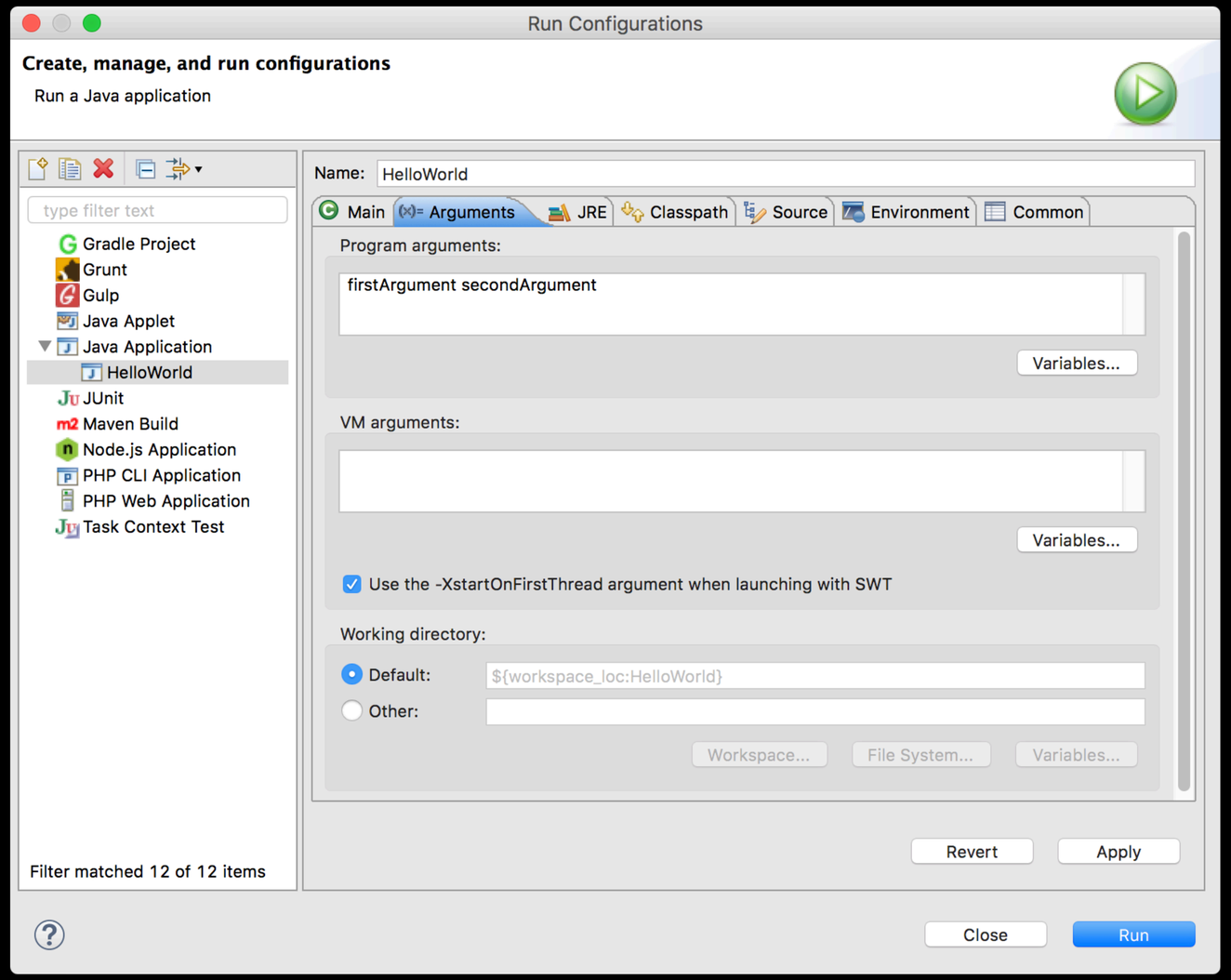
⌘ + /
⌘ + ⇧ + C

Ctrl + /
Ctrl + ⇧ + C

Java Syntax

Arrays: set main arguments





Java Syntax

Arrays: get main arguments

HelloWorld.java

```
1
2  public class HelloWorld {
3      public static void main(String[] args){
4          int numArgs = args.length;
5          for (int i = 0; i < numArgs; i++){
6              System.out.println("arg at position "
7                      +i+" is: "+args[i]);
8          }
9      }
10 }
11
```

Problems @ Javadoc Declaration Console

```
<terminated> HelloWorld [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk
arg at position 0 is: firstArgument
arg at position 1 is: secondArgument
```

arrays are a list containing all the same items

arrays have a length

every entry has an index;
counting starts at zero

type `[]` after the variable to access its members

like `args[0]` to get the first member of `args`

Java Syntax

Arrays: get main arguments

HelloWorld.java

```java

public class HelloWorld {
    public static void main(String[] args){
        int[] numbers = new int[2];
        for (int i = 0; i < args.length; i++){
            numbers[i] = args[i].length();
        }
        for (int i = 0; i < numbers.length; i++){
            System.out.println(args[i]
                    +" has a length of "+numbers[i]);
        }
    }
```

Problems @ Javadoc Declaration Console

```
<terminated> HelloWorld [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk
firstArgument has a length of 13
secondArgument has a length of 14
```

every type can also be an array when you append `[]` after the type declaration

initialisation with `new` keyword:

```java
int[] number = new int[2];
```

Arrays cannot change their `length`! It must be given at initialisation. Therefore the **2** in new `int[2];`

Java Syntax

Arrays: get main arguments

HelloWorld.java

```java

public class HelloWorld {
    public static void main(String[] args){
        String[] myStrings = new String[]{"one","two","three"};
        for (int i = 0; i < myStrings.length; i++){
            System.out.println(i+" = "+myStrings[i]);
        }
    }
}

```

Problems @ Javadoc Declaration Console

```
<terminated> HelloWorld [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/C
0 = one
1 = two
2 = three
```

for initialisation you can even append values in curly brackets **{ }**

Java Syntax ☞

Arrays: get main arguments

HelloWorld.java

```java

public class HelloWorld {
    public static void main(String[] args){
        String[] myStrings = new String[]
                {"zero","one","two","three"};
        for (int i = 0; i < myStrings.length; i++){
            System.out.println(i+" = "+myStrings[i]);
        }
    }
}
```

Problems | Javadoc | Declaration | Console

```
<terminated> HelloWorld [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/C
0 = zero
1 = one
2 = two
3 = three
```

again: indices start at 0 !

Processing

First Steps

Architecture		**Programming**
Drawing/ Modeling/ Rendering	Technique	**Write Code** / **Handle Editor**
History Ergonomics Building Regulations Energy/Water Transp. Indoor Climate / Insulation Structural Engineering Building Process / Materials	Both disciplines require **domain knowledge** in order to design.	Binary Operations Graphical Pipeline **Frameworks** **Object Oriented Programming** Operating Systems File / Database Systems Memory Management ....

Processing

First Steps

Processing Reference

https://processing.org/reference/

file:///Applications/Processing.app/Contents/Java/modes/java/reference/index.html

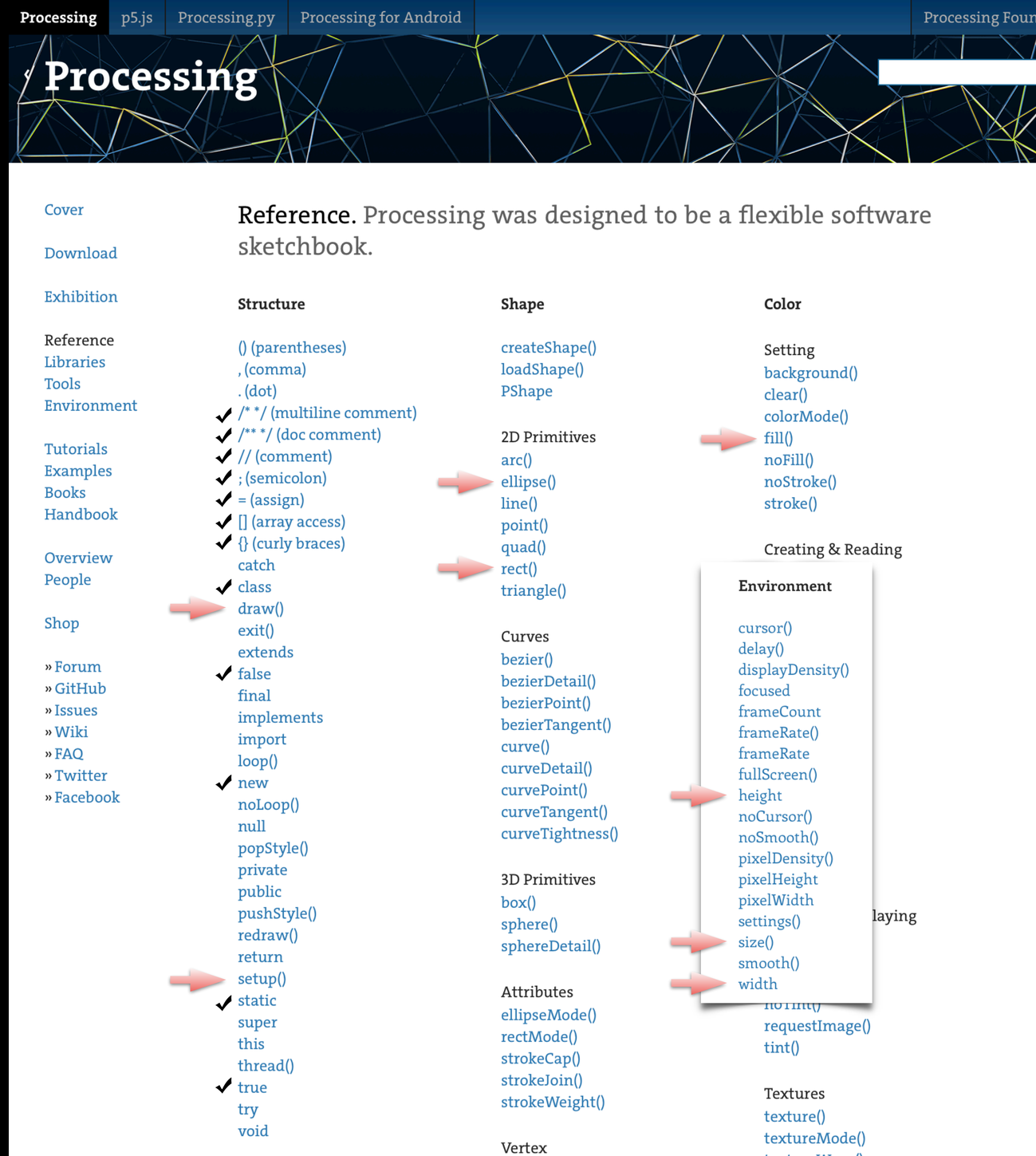
listing all the commands that
Processing adds to standard Java
plus some standard Java you now know already

look at those marked with → for the exercise

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

DARCH

iA Chair of Information Architecture



Thinking Like A Programmer

In our cases there is **NO** such a thing as ~~***"try again, maybe it works"***~~.

A computer has no temper. You may have.

The machine reacts **ALWAYS THE SAME**.

http://www.keepcalm-o-matic.co.uk/p/stay-calm-and-stick-to-the-rules/

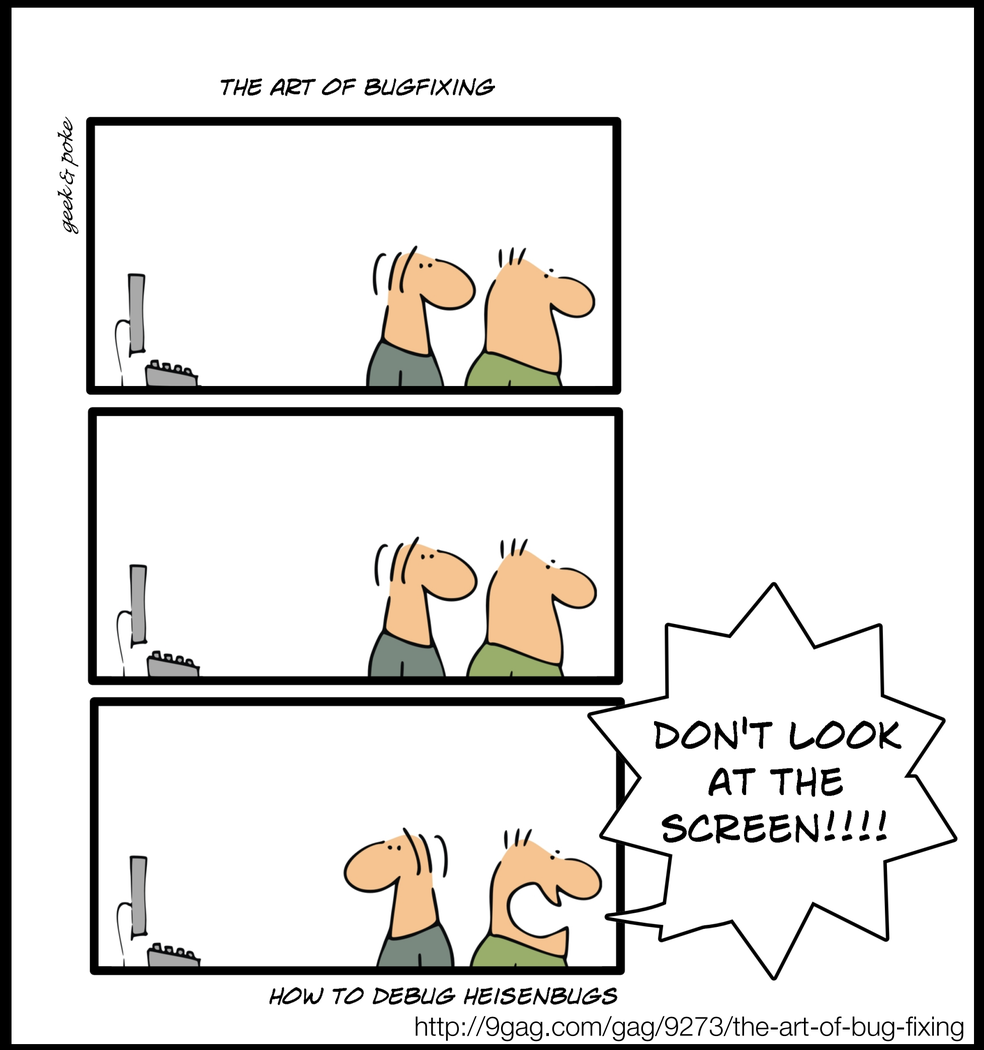
Thinking Like A Programmer

In our cases there is **NO** such a thing as ~~***"try again, maybe it works"***~~.

A computer has no temper. You may have.

The machine reacts **ALWAYS THE SAME**.

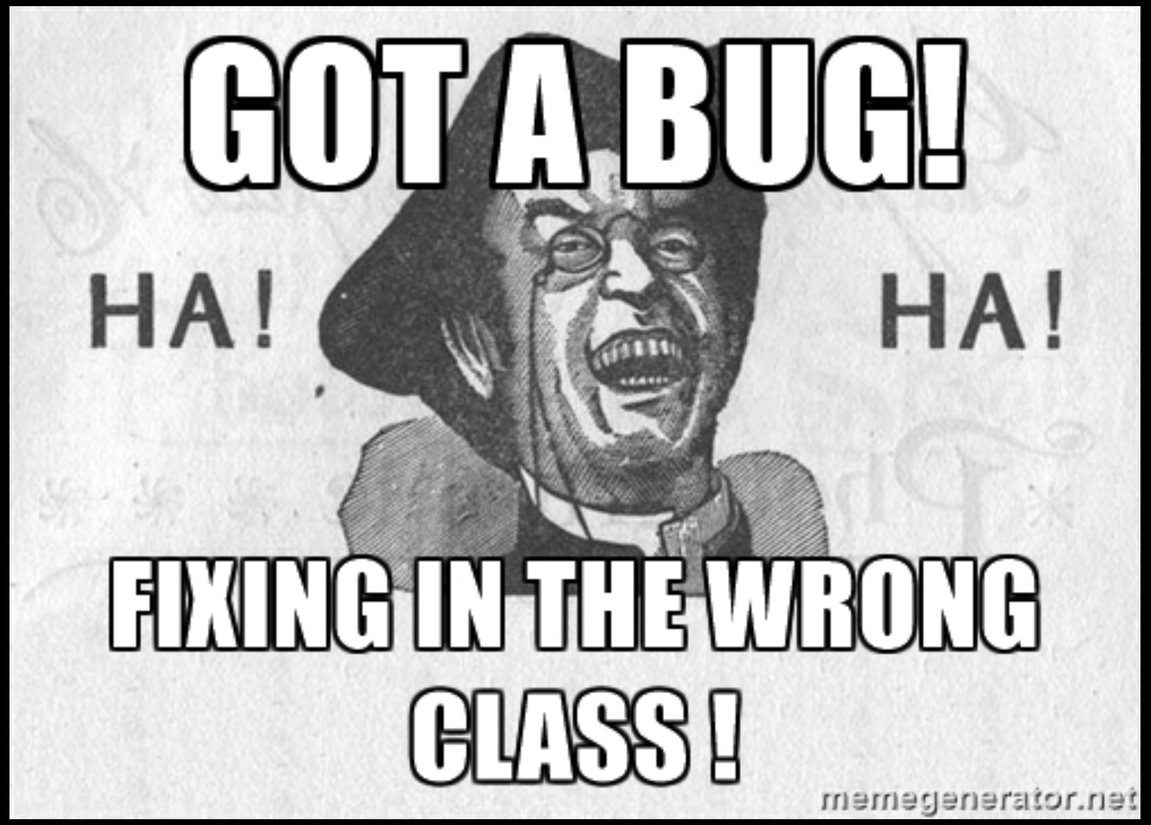
It has no personality!



http://9gag.com/gag/9273/the-art-of-bug-fixing

Thinking Like A Programmer

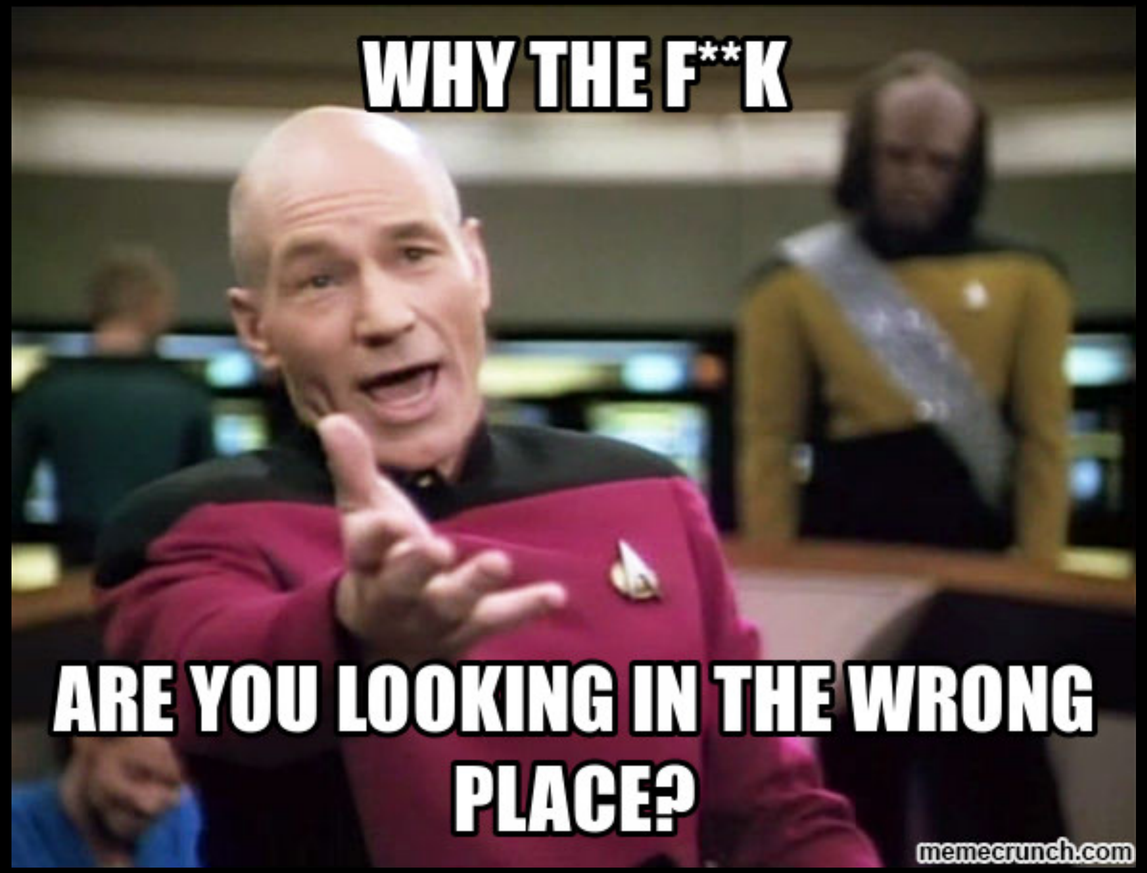
1. you have a bug 🐞
2. you think you know where it is
3. while adding more and more weird stuff the bug is still there
4. you start to think you're just too stupid [but your original code was perfectly fine]
5. hours later you realise the error is caused by something completely different [ask for HELP!]



https://cdn.meme.am/instances/500x/37574915.jpg

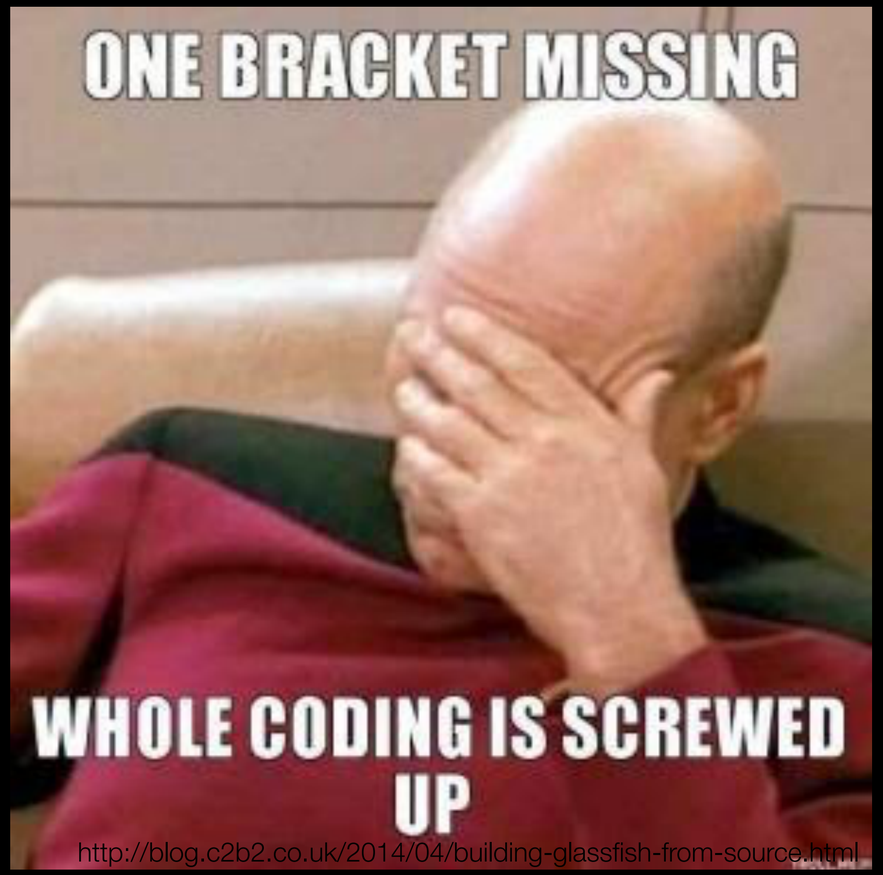
Thinking Like A Programmer

1. you have a bug 🐞
2. you think you know where it is
3. while adding more and more weird stuff the bug is still there
4. you start to think you're just too stupid [but your original code was perfectly fine]
5. hours later you realise the error is caused by something completely different [ask for HELP!]



Thinking Like A Programmer

Syntax / Language Grammar



http://blog.c2b2.co.uk/2014/04/building-glassfish-from-source.html

Thinking Like A Programmer

Syntax / Language Grammar

KEEP
CALM
AND
JUST KEEP
SWIMMING

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

DARCH

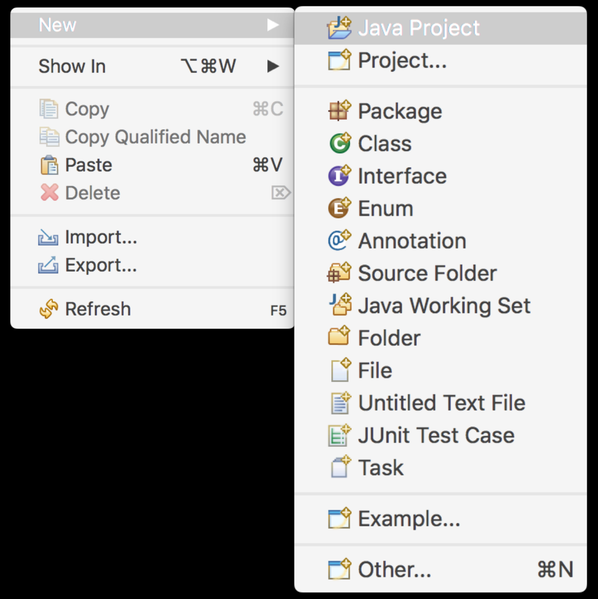
iA Chair of Information Architecture

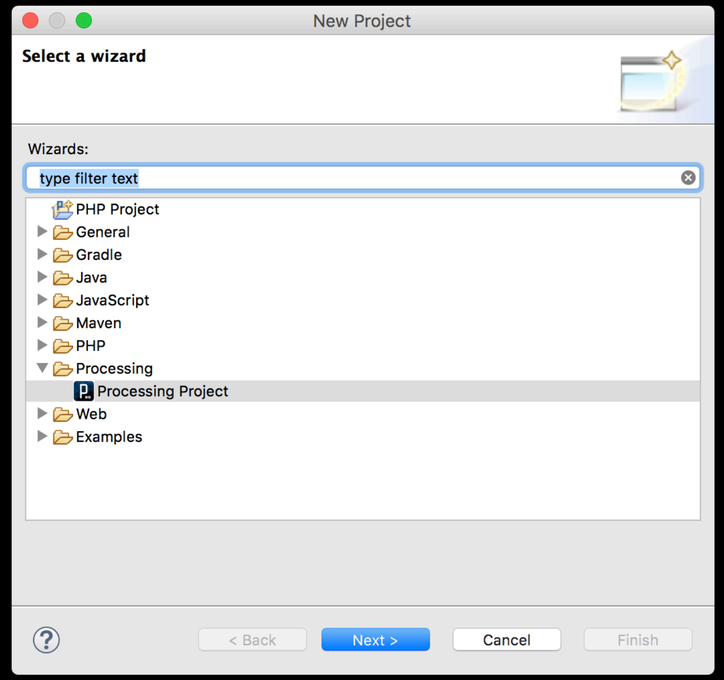
Lukas Treyer | 03.10.2016

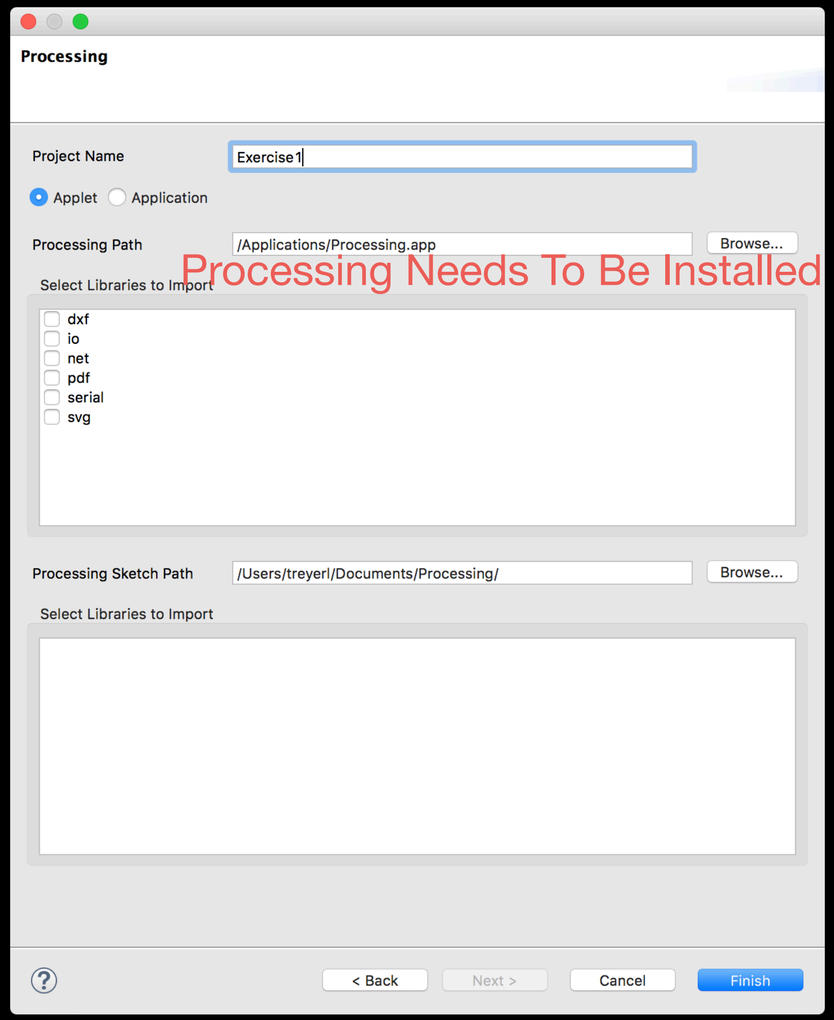
http://gallery.yopriceville.com/Free-Clipart-Pictures/Cartoons-PNG/Finding_Dory_Dory_Transparent_PNG_Clip_Art_Image#.V-5W0KNh2Hs

Processing

First Steps





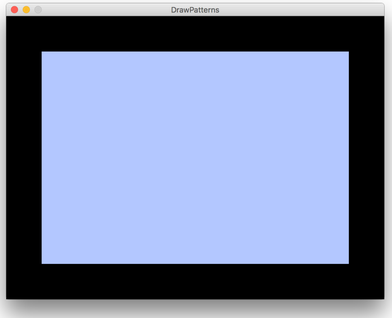


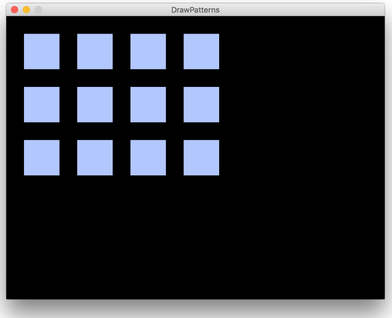
Processing

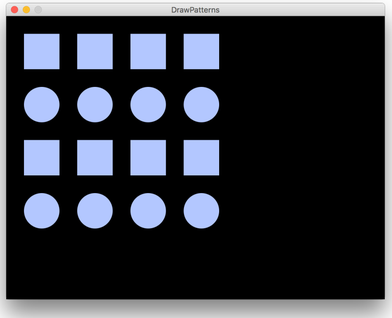
First Exercise

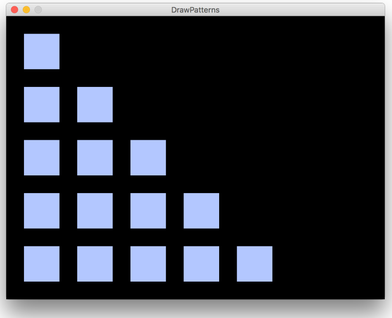
1)
copy the first example from the setup() reference page into your newly created project and try to solve all errors

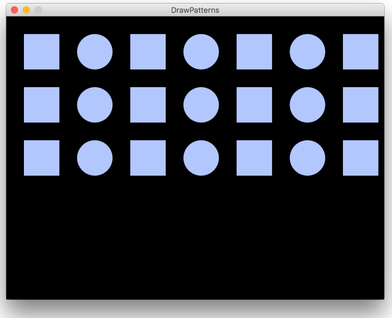
2)
draw the examples shown below using the commands marked with → on the previous slide

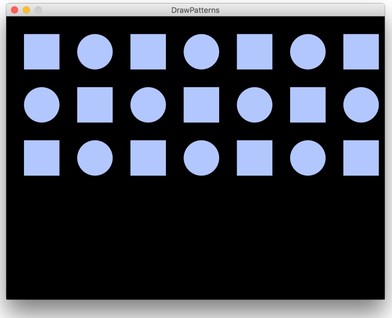
a)


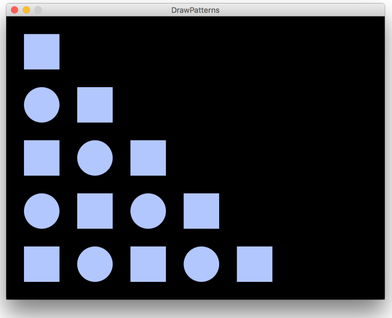
b)


c)


d)


e)


f)


g)


h)