



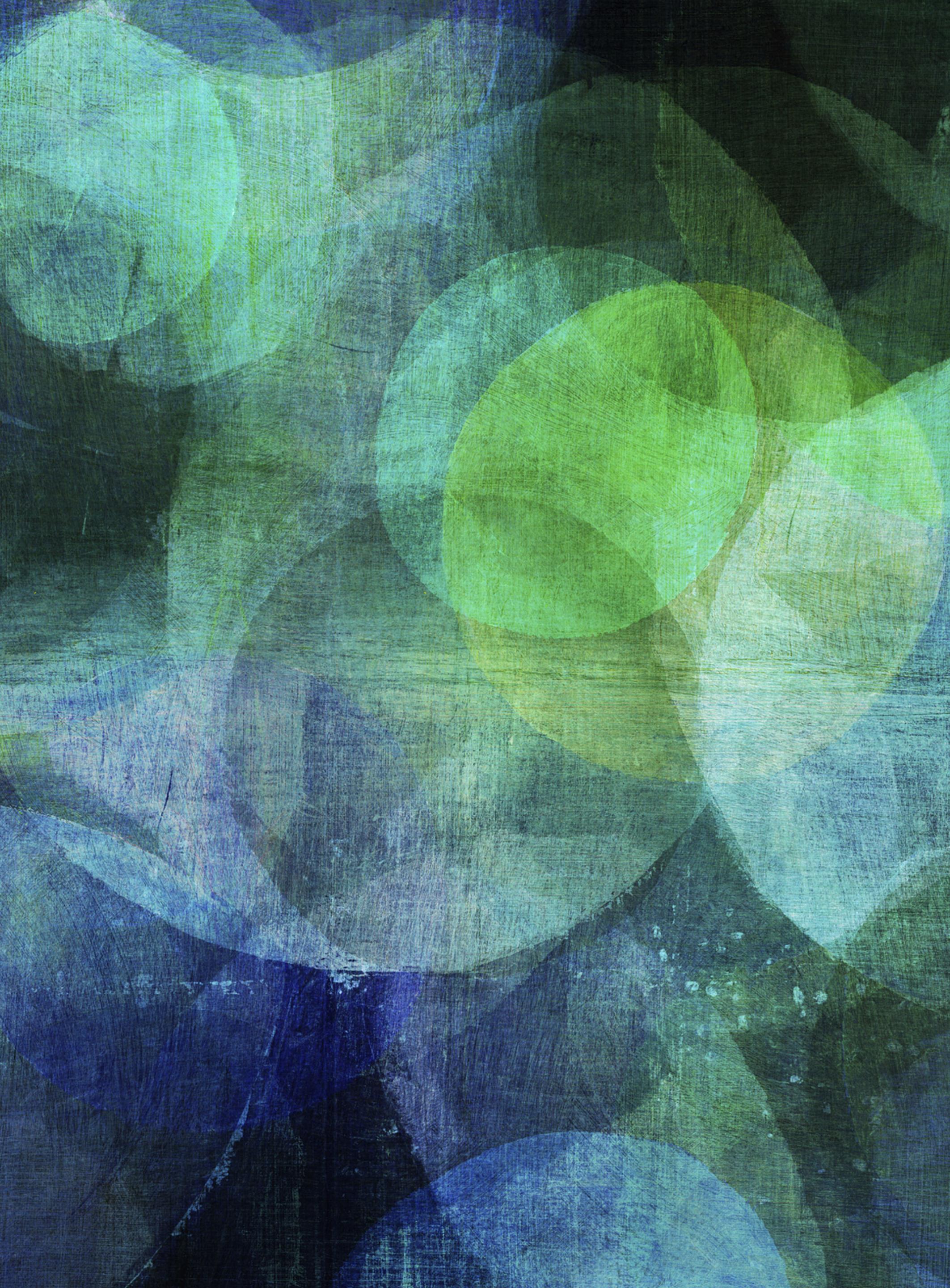
Java™

SPEEDRUN COURSE

By Amine Fattas

I - BASICS

Coding with Java



Hello World

Variables

String class

Data Structures

Flow Control

Loops

Functions

Hello World

- main method
- print in console
- import

Hello World

The screenshot shows a Java code editor with a dark theme. On the left is the code editor pane containing a Java file named `HelloWorld.java`. The code is as follows:

```
no usages
class HelloWorld {
    no usages
    public static void main(String[] args) {
        System.out.println("Hello World");
        // ----- END -----
        System.out.println("\n");
        System.out.println("----- END -----");
    }
}
```

To the right of the code editor is the terminal or run output pane. It displays the following text:

```
Hello World
-----
Process finished with exit
```

The terminal window has a title bar that is partially visible at the top.

Variables

- Declaration, value types, reference types, constants
- Primitives
- Casting (Transtypage)
- Basic Math
- Math Library
- Wrappers

Primitives

```
boolean bool = true;  
  
char character = 'a';  
  
// Numbers  
byte byteVar = 1;  
short shortVar = 1;  
int intVar = 1;  
long longVar = 1l;  
  
float floatVar = 1.0f;  
double doubleVar = 1.0;
```

Casting : byte < short < int < long < float < double

```
// Implicit Casting  
// Valid casting  
short shortResult = byteVar;  
float result = longVar;
```

```
// Invalid Casting  
byte byteResult = shortVar;  
float floatResult = doubleVar;
```

The screenshot shows an IDE interface with two tabs: 'Casting' (selected) and 'Output'. The 'Casting' tab contains Java code demonstrating both valid and invalid casting. The 'Output' tab shows the resulting console output.

```
// Explicit Casting  
byte byteResult = (byte) shortVar;  
float floatResult = (float) doubleVar;  
  
System.out.println("result: " + result);
```

Casting ×

```
result: -24
```

Basic Math

```
class MathLibrary {  
    no usages  
    public static void main(String[] args) {  
  
        // Basic Operations  
        System.out.println("10 + 4 = " + (10 + 4));  
        System.out.println("10 - 4 = " + (10 - 4));  
        System.out.println("10 * 4 = " + (10 * 4));  
        System.out.println("10 / 4 = " + (10 / 4));  
        System.out.println("10 % 4 = " + (10 % 4));  
  
        System.out.println("10 / 4 = " + (10.0 / 4.0));  
  
        int a = 10;  
        a = a + 5;  
        System.out.println("incremented a = " + a);  
        a += 5;  
        System.out.println("incremented a = " + a);  
  
        // Incrementation  
        int counter = 0;  
        System.out.println("counter = " + counter++);  
        System.out.println("counter = " + ++counter);  
    }  
}
```

10 + 4 = 14

10 - 4 = 6

10 * 4 = 40

10 / 4 = 2

10 % 4 = 2

10 / 4 = 2.5

incremented a = 15

incremented a = 20

counter = 0

counter = 2

Math Lib

```
[  
.....  
void main(String[] args) {  
  
    ..println( "Math.abs(-4.1456) = " + Math.abs(-4.1456) );  
    ..println( "Math.ceil(4.1456) = " + Math.ceil(4.1456) );  
    ..println( "Math.floor(4.1456) = " + Math.floor(4.1456) );  
    ..println( "Math.round(4.1456) = " + Math.round(4.1456) );  
    ..println( "Math.max(5, 10) = " + Math.max(5, 10) );  
    ..println( "Math.min(5, 10) = " + Math.min(5, 10) );  
    ..println( "Math.exp(1) = " + Math.exp(1) );  
    ..println( "Math.pow(10, 4) = " + Math.pow(10, 4) );  
    ..println( "Math.sqrt(9) = " + Math.sqrt(9) );  
    ..println( "Math.PI = π = " + Math.PI );  
    ..println( "Math.sin(π/2) = " + Math.sin(Math.PI/2));  
    ..println( "Math.sin(π/2) = " + Math.cos(Math.PI/2));  
    ..println( "Math.tan(π/2) = " + Math.tan(Math.PI/2));  
  
    ..println( "Random float in [0,1] = " + Math.random());  
    ..println( "Random float in [0,20] = " + Math.random() * 20);  
    ..println( "Random int in [0,20] = " + (int) (Math.random() * 20));  
  
. END -----  
.println("\n");  
.println("----- END -----");  
.....]  
Math.abs(-4.1456) = 4.1456  
Math.ceil(4.1456) = 5.0  
Math.floor(4.1456) = 4.0  
Math.round(4.1456) = 4  
Math.max(5, 10) = 10  
Math.min(5, 10) = 5  
Math.exp(1) = 2.7182818284590455  
Math.pow(10, 4) = 10000.0  
Math.sqrt(9) = 3.0  
Math.PI = π = 3.141592653589793  
Math.sin(π/2) = 1.0  
Math.sin(π/2) = 6.123233995736766E-17  
Math.tan(π/2) = 1.633123935319537E16  
Random float in [0,1] = 0.4022363050729939  
Random float in [0,20] = 12.571126723239722  
Random int in [0,20] = 11
```

Wrappers

```
int primitiveInt = 1; // value type
Integer objectInt = 1;    // reference type

primitiveInt = null; // Primitives cannot be null
objectInt = null;

// Other Wrappers
// Boolean, Character, Byte, Short, Long, Float, Double ...
```

“

The scope of a local variable is the block It was
declared in

String class

- Declaration
- Concatenation
- Parsing
- important methods
- StringBuilder & StringBuffer

String Concatenation & Parsing

The screenshot shows a Java code editor with a dark theme. On the left, the code is written in Java:

```
String str = "Hello ";
String name = "String";

//Concatenation
System.out.println(str + name);

// Parsing
Integer number = 46;
String stringNum = "1234";
System.out.println(46);                                // primitive to String
System.out.println(number.toString());                 // Reference Type to String
System.out.println(Integer.parseInt(stringNum));      // String to Int
System.out.println(Double.parseDouble(stringNum));    // String to Double
```

On the right, the output of the code is displayed in the terminal window:

```
/Users/aminefattas/Library/Mobile Documents/com~apple~CloudDocs/Java/IDEAProjects/JavaBasics/src/main/java/com/aminefattas/basicjava/StringManipulation.java:2: error: illegal start of expression
String str = "Hello ";
          ^
1 error
```

The terminal then shows the concatenated string and the parsed values:

```
Hello String
46
46
1234
1234.0
```

Important methods

```
String str = " Hello World ";
String str2 = new String(original: " Hello World ");

System.out.println("charAt 7: " + str.charAt(7));
System.out.println("indexOf W: " + str.indexOf('W'));
System.out.println("contains: " + str.contains("Hello"));
System.out.println("length: " + str.length());

System.out.println("== : " + (str == str2));
System.out.println("equals: " + str.equals(str2));

System.out.println("replace: " + str.replace(
    target: "World",
    replacement: "Universe"
));

System.out.println("substring: " + str.substring(5,10));

for(String word: str.split(regex: ""))
    System.out.println("split: " + word + ",");

System.out.println("toUpperCase: " + str.toUpperCase());
System.out.println("trim: " + str.trim());
```

charAt 7:
indexOf W: 8
contains: true
length: 15
== : false
equals: true
replace: Hello Universe
substring: lo Wo
split: ,
split: ,
split: Hello,
split: World,
toUpperCase: HELLO WORLD
trim: Hello World

StringBuilder & StringBuffer

```
String str = "Hello ";

StringBuilder stringBuilder = new StringBuilder(str);

stringBuilder.append("World1 ");
stringBuilder.append("World2 ");
stringBuilder.append("World3 ");
stringBuilder.append("World4 ");
System.out.println(stringBuilder);

stringBuilder.insert( offset: 0, str: "Big " );
System.out.println(stringBuilder);

stringBuilder.delete(0,2);
System.out.println(stringBuilder);
```

/Users/aminefattas/Library/Java/JavaView.jar

Hello World1 World2 World3 World4

Big Hello World1 World2 World3 World4

g Hello World1 World2 World3 World4

----- END -----

Process finished with exit code 0

Data Structures

- Arrays
- ArrayLists
- Stacks
- Hashmaps

Arrays

The screenshot shows an IDE interface with a dark theme. On the left, there is a code editor containing Java code. On the right, there is a terminal window showing the output of the code. The code itself is as follows:

```
int[] numbers1 = {1, 15, 42, 7, 4, 5};  
System.out.println(numbers1[3]);  
  
int numbers2[] = {1, 15, 42, 7, 4, 5};  
System.out.println(numbers1[2]);  
  
int[] numbers3 = new int[6];  
numbers3[5] = 10;  
System.out.println(numbers3[5]);  
System.out.println(numbers3[4]);  
  
System.out.println(numbers3.length);  
  
int[][] matrix = {  
    { 43, 54, 234 },  
    { 656, 34, 98 }  
};  
System.out.println(matrix[1][2]);  
  
int copyOfNumbers[] = Arrays.copyOf(numbers1, newLength: 6);  
System.out.println(copyOfNumbers.equals(numbers1));  
  
Arrays.sort(numbers1);  
System.out.println(Arrays.toString(numbers1));
```

The terminal output is:

```
7  
42  
10  
0  
6  
98  
false  
[1, 4, 5, 7, 15, 42]  
----- END -----
```

ArrayList

```
ArrayList<String> arrayList = new ArrayList<String>();  
  
ArrayList<String> arrayListWithCapacity = new ArrayList<String>( initialCapacity: 10);  
  
ArrayList<String> arrayListWithInitValues =  
    new ArrayList<String>(Arrays.asList("Monday", "Tuesday", "Wednesday"));  
  
arrayList.add("My name");  
arrayListWithInitValues.add("Thursday");  
arrayListWithInitValues.remove( o: "Tuesday");  
  
arrayListWithInitValues.set(0, "Sunday");  
System.out.println(arrayListWithInitValues.toString());  
System.out.println(arrayListWithInitValues.get(2));  
  
// ----- END -----  
System.out.println("\n");  
System.out.println("----- END -----");
```

[Sunday, Wednesday, Thursday]
Thursday
----- END -----
Process finished with exit code 0

Stacks

```
Stack<String> stack = new Stack<>();  
  
stack.add("Monday");  
stack.add("Tuesday");  
stack.add("Wednesday");  
stack.add("Thursday");  
  
System.out.println(stack.toString());
```

```
System.out.println(stack.pop());  
System.out.println(stack.toString());
```

/Users/aminefattas/Library/Java/JavaVi
[Monday, Tuesday, Wednesday, Thursday]

Thursday

[Monday, Tuesday, Wednesday]

----- END -----

HashMap

```
HashMap<Integer, String> months = new HashMap<>();  
  
months.put(1, "January");  
months.put(2, "February");  
months.put(3, "March");  
months.put(4, "April");  
months.put(5, "May");  
  
System.out.println(months.get(3));  
  
----- END -----  
  
Process finished with exit code 0  
  
HashMap<String, String> days = new HashMap<>();  
  
days.put("Mo", "Monday");  
days.put("Tu", "Tuesday");  
days.put("We", "Wednesday");  
days.put("Th", "Thursday");  
  
System.out.println(days.get("Tu"));
```

Flow Control

- if else
- Ternary operator
- switch, case, break, default
- Exception handling, try catch finally

Flow Control:

The screenshot shows a Java code editor with the following code:

```
class FlowControl {
    public static void main(String[] args) {
        // if else
        int age = 25;

        if (age > 18) {
            System.out.println("Adult");
        } else {
            System.out.println("Minor");
        }

        // Ternary Operator
        String message = (age > 18) ? "Adult" : "Minor";
        System.out.println(message);

        // switch case
        String today = "Th";

        switch (today) {
            case "Mo":
                System.out.println("It's Monday");
                break;
            case "Tu":
                System.out.println("It's Tuesday");
                break;
            case "We":
                System.out.println("It's Wednesday");
                break;
            case "Th":
                System.out.println("It's Thursday");
                break;
            default:
                System.out.println("It's Weekend");
                break;
        }
    }
}
```

The code uses an if-else statement to print "Adult" or "Minor" based on the age. It also demonstrates the ternary operator to assign a string value. Finally, it uses a switch case statement to determine the day of the week and print the corresponding message.

The output window shows the results of the execution:

```
Process finished with exit code 0
```

Output:

```
Adult
Adult
It's Thursday
```

END

Exception Handling: try, catch, finally

The screenshot shows a Java code editor with a dark theme. On the left, there is a code editor window containing the following Java code:

```
try {
    System.out.println( 10/0 );
}
catch (Exception exception){
    System.out.println( exception.toString() );
}

finally {
    System.out.println( "Calcul finished" );
}
```

The line `System.out.println(10/0);` is highlighted in yellow. The line `exception.toString()` is also highlighted in yellow. The code editor has a vertical scrollbar on the right.

To the right of the code editor is a terminal window showing the execution results:

```
/Users/aminefattas/Library/Java/JavaVirtualMachine
java.lang.ArithmetricException: / by zero
Calcul finished
----- END -----
Process finished with exit code 0
```

The terminal window has a dark background and light-colored text. The path `/Users/aminefattas/Library/Java/JavaVirtualMachine` is at the top. The error message `java.lang.ArithmetricException: / by zero` is followed by the output of the `finally` block: `Calcul finished`. A horizontal dashed line separates the output from the exit status: `----- END -----`. The exit code `0` is at the bottom.



In the catch block argument, It is not recommended to define your exception as vague
Exception type

```
catch (ArithmetcException exception){ // Type definition Recommended
    System.out.println( exception.toString() );
}

catch (Exception exception){ // Not recommended
    System.out.println( exception.toString() );
}
```

Exception Handling: throw

```
try {
    int denominator = 0;
    System.out.println( 10/denominator );
    if (denominator == 0)
        // Manually throw an exception
        throw new ArithmeticException("Cannot divide by 0");
}
catch (ArithmeticException exception){ // Type definition Recommended
    System.out.println( exception.toString() );
}
```

Loops

- for
- Enhanced for
- while
- do while

For loop

The screenshot shows a Java code editor with two sections of code and their corresponding outputs.

Normal For Loop:

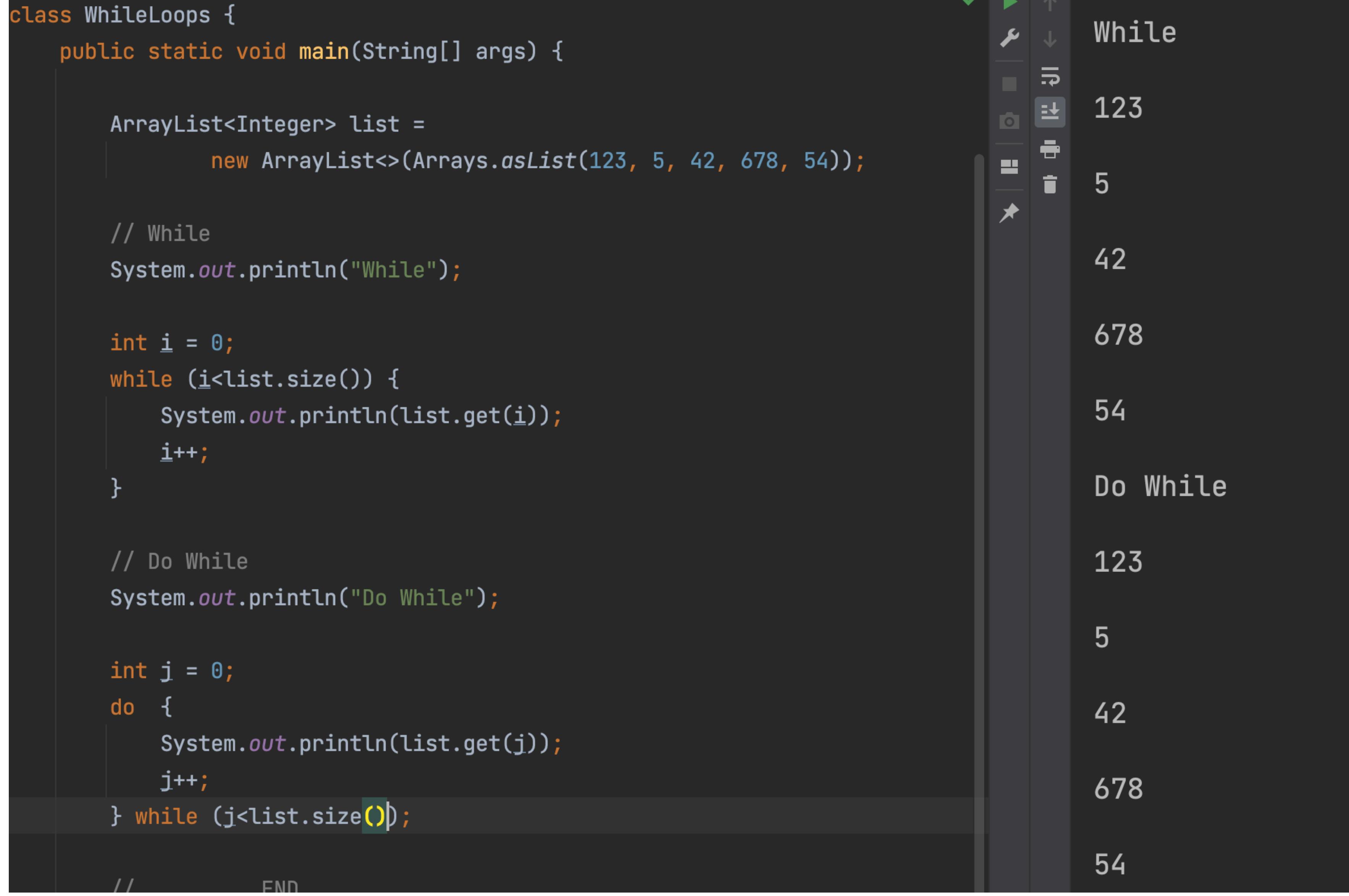
```
ArrayList<Integer> list =  
    new ArrayList<>(Arrays.asList(123, 5, 42, 678, 54));  
  
// Normal For Loop  
System.out.println("Normal For Loop");  
  
for (int i = 0; i < list.size() - 1; i++) {  
    System.out.println(list.get(i));  
}  
  
// Enhanced For Loop  
System.out.println("Enhanced For Loop");  
  
for (int num : list) {  
    System.out.println(num);  
}
```

Output:

Normal For Loop	Enhanced For Loop
123	123
5	5
42	42
678	678
	Enhanced For Loop
	123
	5
	42
	678
	54

While loops

```
class WhileLoops {  
    public static void main(String[] args) {  
  
        ArrayList<Integer> list =  
            new ArrayList<>(Arrays.asList(123, 5, 42, 678, 54));  
  
        // While  
        System.out.println("While");  
  
        int i = 0;  
        while (i < list.size()) {  
            System.out.println(list.get(i));  
            i++;  
        }  
  
        // Do While  
        System.out.println("Do While");  
  
        int j = 0;  
        do {  
            System.out.println(list.get(j));  
            j++;  
        } while (j < list.size());  
  
        // END
```



The screenshot shows the execution of a Java program named `WhileLoops`. The code contains two loops: a `while` loop and a `do` loop, both printing elements from an `ArrayList` of integers. The output window displays the following results:

- While loop output:
 - 123
 - 5
 - 42
 - 678
 - 54
- Do While loop output:
 - 123
 - 5
 - 42
 - 678
 - 54

Functions

- Syntaxe
- Multiple Arguments (args ...)
- Recursive functions
- Throwing Functions

Syntax: returnType + functionName + Arguments + CodeBlock

The screenshot shows a Java code editor and a terminal window. The code in the editor is:

```
1 usage
static void printItemsOf(ArrayList<Integer> list) {
    for (int i = 0; i < list.size() - 1; i++) {
        System.out.println(list.get(i));
    }
}

public static void main(String[] args) {
    ArrayList<Integer> list =
        new ArrayList<>(Arrays.asList(123, 5, 42, 678, 54));
    // We extracted the printing logic to a function
    Functions.printItemsOf(list);

    // ----- END -----
    System.out.println("\n");
    System.out.println("----- END -----");
}
```

The terminal window to the right displays the output of the program:

```
123
5
42
678
----- END -----
Process finished with exit code 0
```

Multiple Arguments

The screenshot shows a Java code editor with a dark theme. On the left, the code defines a static method `sum` that takes multiple integer arguments and returns their sum. The main method calls this sum function with five arguments (12, 34, 63, 234, 678) and prints the result. The output window on the right shows the printed output: "Somme = 1021".

```
usage: new *
static int sum(int ... numbers) {
    int sum = 0;
    for (int number: numbers) {
        sum += number;
    }
    return sum;
}
new *
public static void main(String[] args) {

    System.out.println(
        "Somme = " + MultipleArguments.sum( ...numbers: 12, 34, 63, 234, 678)
    );
}
```

/Users/aminefattas/Li
Somme = 1021
----- END -----
Process finished with

Recursive functions

The screenshot shows an IDE interface with a dark theme. On the left, there is a code editor containing Java code. On the right, there is a terminal window showing the execution results.

Code Editor Content:

```
2 usages
static void printItemsOf(ArrayList<Integer> list, int i) {
    if (i < list.size()) {
        System.out.println(list.get(i));
        RecusiveFunction.printItemsOf(list, i+1);
    }
}

public static void main(String[] args) {
    ArrayList<Integer> list =
        new ArrayList<>(Arrays.asList(123, 5, 42, 678, 54));

    RecusiveFunction.printItemsOf(list, i: 0);

    // ----- END -----
    System.out.println("\n");
    System.out.println("----- END -----");
}
```

Terminal Output:

```
/Users/amineattas/Library/Mobile Documents/com~apple~CloudDocs/Java/Recursion/RecusiveFunction.java:1: error: cannot find symbol
static void printItemsOf(ArrayList<Integer> list, int i) {
                                     ^
  symbol:   class ArrayList
  location: class RecusiveFunction
1 error

Process finished with exit code 1
```

The terminal output shows an error message indicating that the symbol 'ArrayList' could not be found. The code itself is a recursive function named `printItemsOf` that prints the elements of a list. It is called from the `main` method with an initial index of 0. The expected output is visible in the terminal, showing the numbers 123, 5, 42, 678, and 54, each on a new line, followed by a blank line and the text "----- END -----".

Throwing functions: throws

```
package com.example;

public class ThrowingFunctions {
    static void divideByZero() throws ArithmeticException {
        System.out.println( 10/0 );
    }

    public static void main(String[] args) {
        try {
            ThrowingFunctions.divideByZero();
        }
        catch (ArithmeticException exception){ // Type definition Recommended
            System.out.println( exception.toString() );
        }
    }
}
```

LET'S PRACTICE !

Exercise 1

Moroccan national football team stops at Mamounia Hostel rooms:

101	102	104	202	205	211	303	304	310	404
Hakimi	Saiss	Ziyech	Ounahi	Nesyri	Bono	Yamiq	Boufal	Regragui	Amrabat

- A. Represent the data in an appropriate collection.
- B. Print All the players sorted by their room numbers.
- C. Get the player name is in room 202.
- D. "Regragui" is not a player ! Replace it by "Sabiri" F
- E. There was an input error ! "Amrabat" is actually in room 405. Fix it.
- F. Get the player(s) whose name(s) start with 'B'.

II - INTRO TO OOP

Object Oriented Programming



Classes

Encapsulation

Inheritance

Polymorphism

Abstraction

Interfaces

Generics

Classes

- Properties
- Methods
- Constructor
- Garbage Collector
- `toString()`
- Instance properties/methods vs Class properties/methods
- Naming Conventions

Naming Conventions

Constant	SCREAMING_SNAKE_CASE
Variable Method	camelCase
Class Interface Enum	PascalCase

Exercise 2

Moroccan national football team stops at Mamounia Hostel rooms:

101	102	104	202	205	211	303	304	310	404
Hakimi	Saiss	Ziyech	Ounahi	Nesyri	Bono	Yamiq	Boufal	Regragui	Amrabat

- A. Represent the data as **Objects**.
- B. Print All the players sorted by their room numbers.
- C. Get the player name is in room 202.
- D. "Regragui" is not a player ! Replace it by "Sabiri" F
- E. There was an input error ! "Amrabat" is actually in room 405. Fix it.
- F. Get the player(s) whose name(s) start with 'B'.

Encapsulation

- Access Control
- Getters & Setters

“

A public class must have the same name of its
file

Inheritance

- Syntaxe and application

Polymorphism

- Method Overriding (Redéfinition)
- Method Overloading (Surcharge)

Abstraction

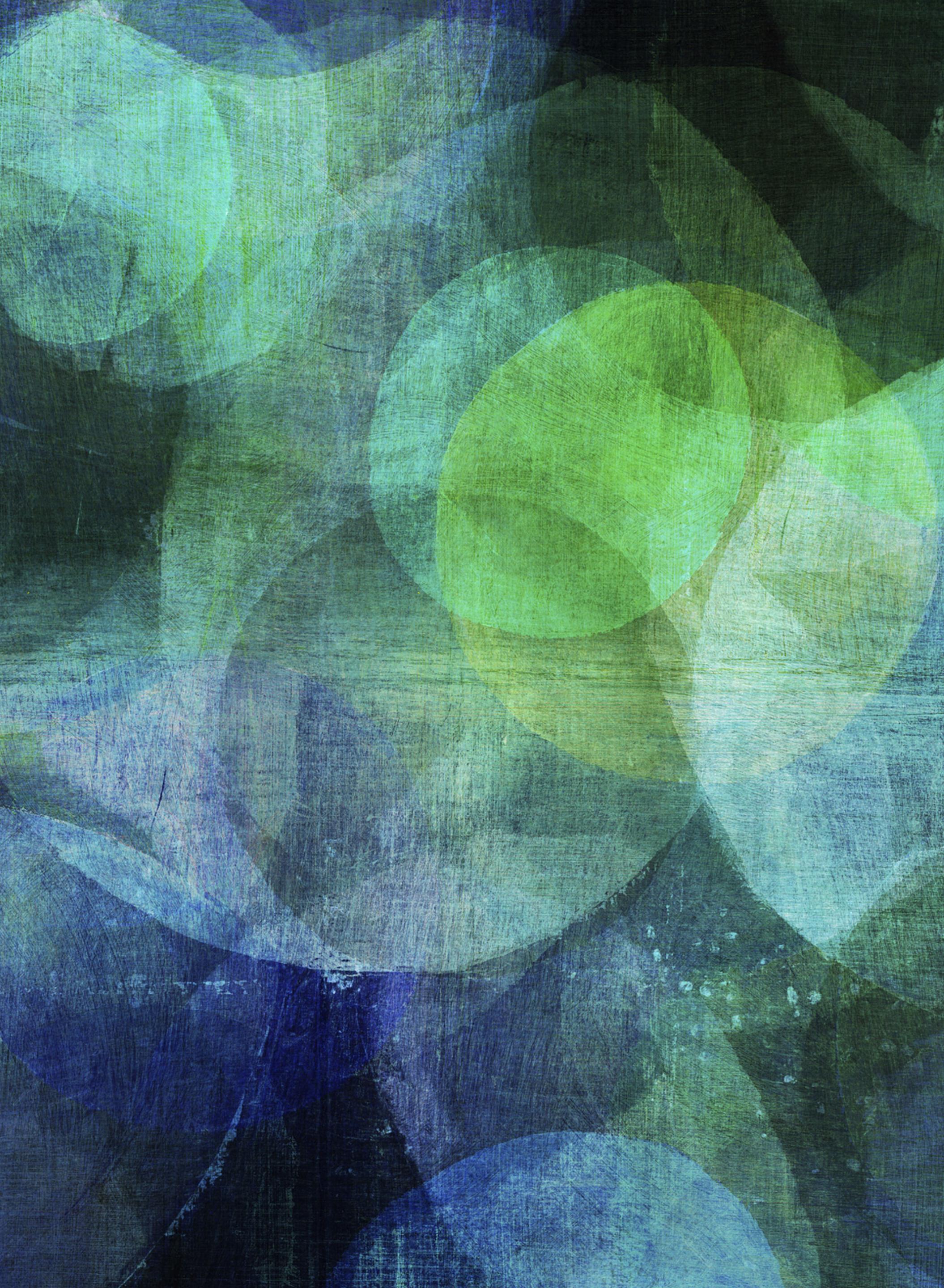
- Abstraction example with polymorphism
- abstract Class

Interfaces

- As a Contract
- Abstraction
- Delegation

III - NEXT STEPS

It's Google time !



Some common topics often found in Real Projects:

Lambda Expressions

URL Format

Dates

Regex

Design Patterns (Singleton, MVC, ...)

Maven

Threads



**THANK YOU
BEST OF LUCK**