Phase 1 : Java

Phase 2 : Spring Framework

Phase 3 : DevOps Tools

Java - 8 : IoT

Lambda Expression

base64 API

Streams

Functional Interface

default

method reference

Optional

DateTime API

Concurrent API enhanced

Nashorn Engine (JS engine)

Functional Programming

functions as first class citizens + OOPs

Imperative style of programming

Classical/Traditional

Focus : how to perform operation

write steps on how to achieve an objective

Object mutability

Declarative style of programming

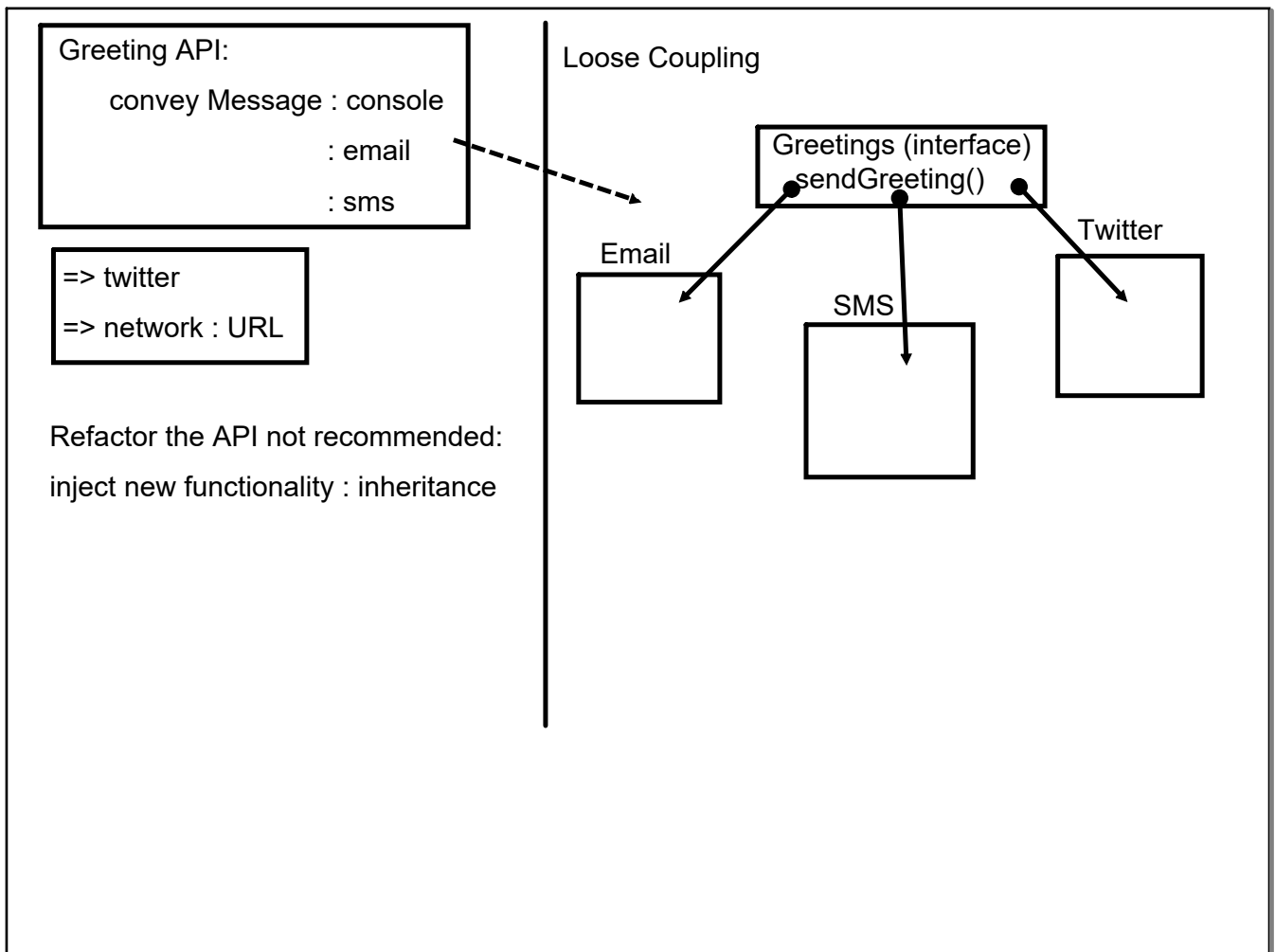# Focus : what is the result

Object immutability

SQL style

Collection of numbers

fetch the unique numbers

wf.com

com.wf

<reverse domain of org><training>

Greeting API:

     convey Message : console

                : email

                : sms

=> twitter

=> network : URL

Refactor the API not recommended:

inject new functionality : inheritance

Loose Coupling

Greetings (interface)
sendGreeting()

Email

SMS

Twitter

Declarative:

Inject only functionality (pure function), not wrapped inside an object

Java should expose a datatype : Function

New Datatype : would not be backward compatible

interface : Function datatype

Syntax :

1. not have any access modifier

2. Anonymous function (no name)

3. return type is not mentioned

4. no param types

5. <praram name> -> {<definition>}

```
void fun(String str1,String str2)
{
}
(str1,str2) -> {
}
```

```
void fun(String str){
      ..only single instruction
}
 str-> <single instruction>
```

```
void fun(String str1){
}
str1->{
}
```

```
void fun(){
}
()->{
}
```

```
void fun(String str){
---------
---------
}
str -> {
-------------
--------------
}
```
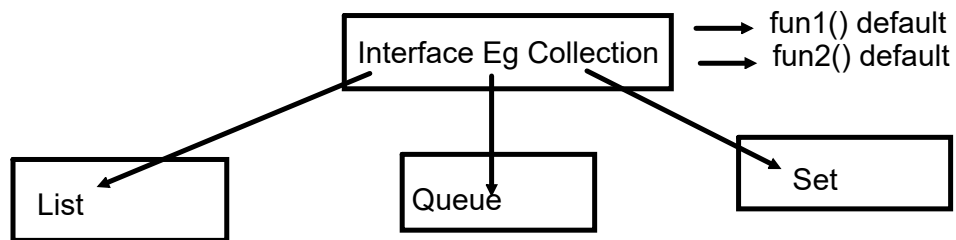
```
int add(int a,int b){

    return a+b;

}
(a,b) -> a+b;    // single instruction return is by default associated
(a,b) -> {

    return a+b;

}
```

interface :

 1. default functions : interfaces can have functions with definition

```
                          ┌──────────────────────┐  ──────▶  fun1() default
                          │ Interface Eg Collection │  ──────▶  fun2() default
                          └──────────────────────┘
                     ╱              │              ╲
         ┌──────────┐       ┌──────────┐       ┌──────────┐
         │   List   │       │  Queue   │       │   Set    │
         └──────────┘       └──────────┘       └──────────┘
```

Functional Interface :

    An interface containing only a single abstract method

    it may have multiple default and static method

Lambdas/Method Reference can be assigned to only functional interface reference

Lambda Expression/ Method Reference signature must match with the only abstract method
of FI

An reference of functional interface can refer to any method as long as its signature matches with the only abstract method (other than lambdas also)

More Practical Usage....

Streams

Existing interface

    # Runnable

    # Comparator

    # Comparable