Java 8 : Functional Programming

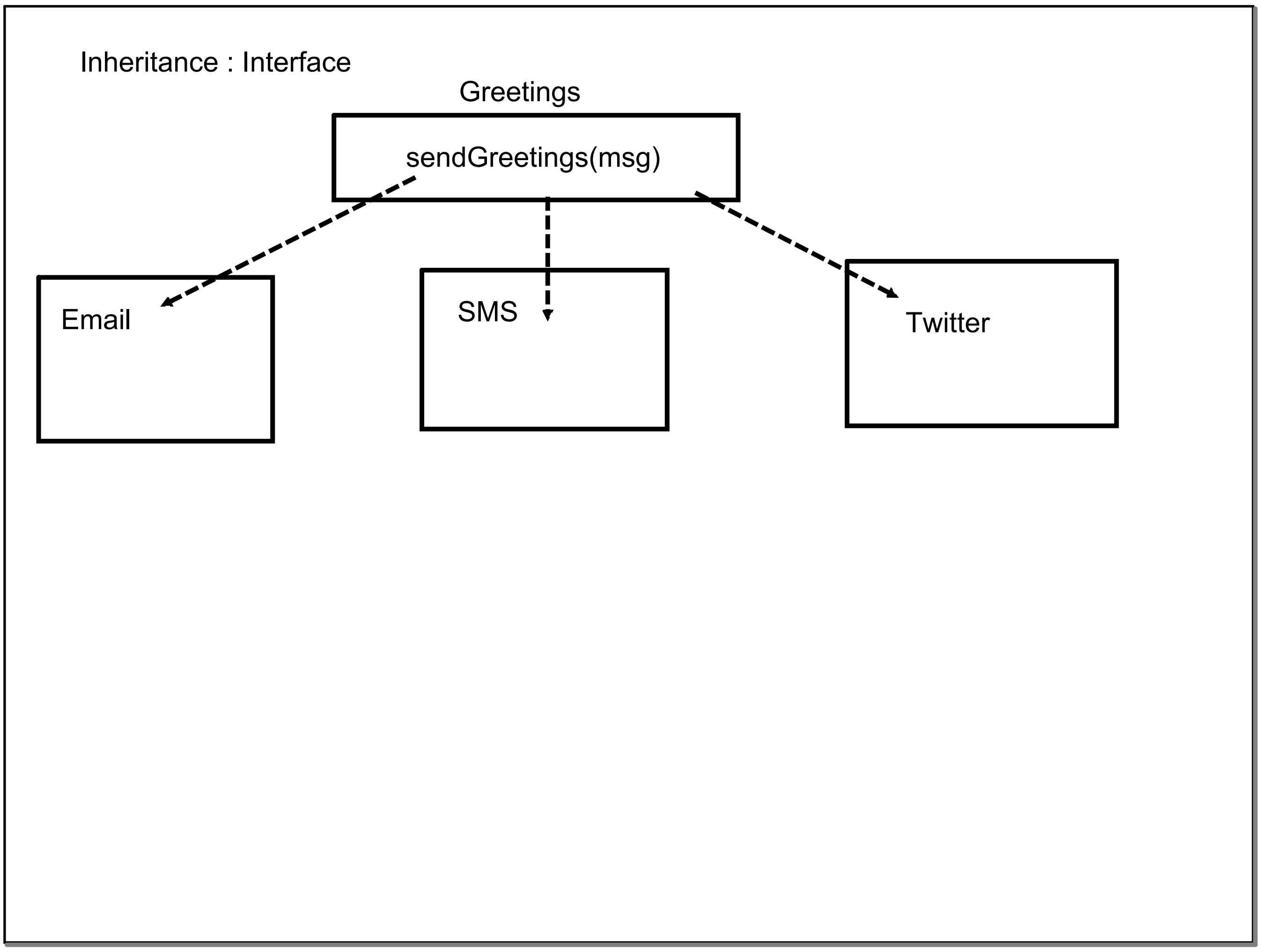
Procedure Oriented

Object Oriented

Functional Oriented :

Imperative style : Classic OOPs
- how to perform operation
- Object mutability

Declarative style : Functional
- focus on function/methods
- concise & readable
- Embraces the immutable objects


Inheritance : Interface
Greetings
sendGreetings(msg)
Email
SMS
Twitter

Lambda expressions

single argument

```
(arg)->{
}
arg->{
}
```

multiple arguments

```
(arg1,arg2)->{
}
```

no arg

```
()->{
}
```

single instruction

```
(arg)-> single instruction
```

multiple instruction

```
(arg)->{
        stmt1
        stmt2
}
```

returning values

```
(arg)-> instruction  // by default associated with return


(n1,n2)-> n1 + n2;
(n1,n2)-> {
     return n1 + n2;
}
```

Functional Interface

1. Lambda expression can be assigned only to Functional Interface

    Functional Interface : only one abstract method

2. Lambda expression/ any method signature must match with the only abstract method of interface

API support for functional programming

functionalAPI

FunctionalInterface :

Assign an instance of derived class

Assign an lambda

Assign any other method if signature matches

Common Method Signature <Generics>

Consumer	:void accept(<T>)
Predicate	: boolean test(<T>)
Function	: <T> apply(<T>)
Supplier	: <T> get()

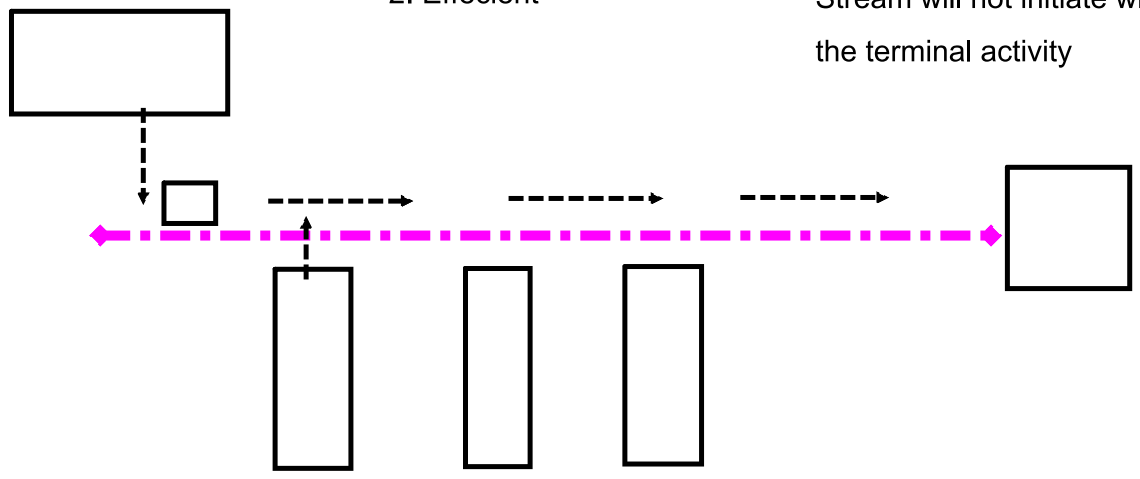
Optional

Stream ~ Conveyer belt

1. immutable (Thread Safe)
2. Effecient

1. Fetch the stream
2. actions on stream
3. Terminal Activity (must)

Stream will not initiate without the terminal activity



1. Collection is sequential, and need to be processed in same sequence
2. taking use of external service/data which is mutable (Not Thread Safe)
3. activity is inherentlyComplex : parallel processing is in-effecient