

==>Java 8-11

==>Reactive Programming (Spring)

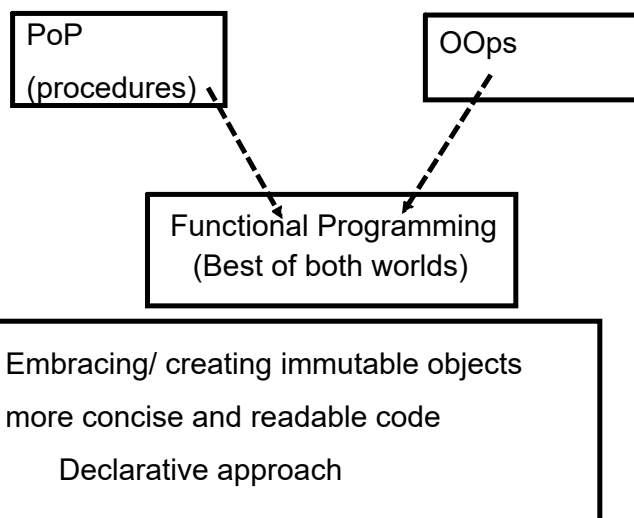
java 1.5 Functional Programming

java 1.8

==> Technological advancement : mobile/laptops/system

Java new feature simplifies concurrency operations

Functional Programming:



Traditional : Imperative

=> Focus on how to perform

=> Object mutability

Declarative Style:

=>Focus on what result we want

=>Object immutability

=>Analogous to SQL (use of already existing part of library to achieve an objective)

Improvisation on interface:

default method

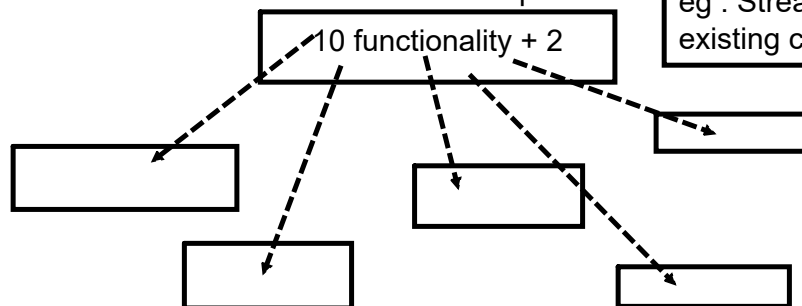
static method

Demand from developer community :

#provision to add new functionality at the top without breaking existing API

interface exists at top level

eg : Stream feature req to be added in existing collection api



Functional Interface : Lambda

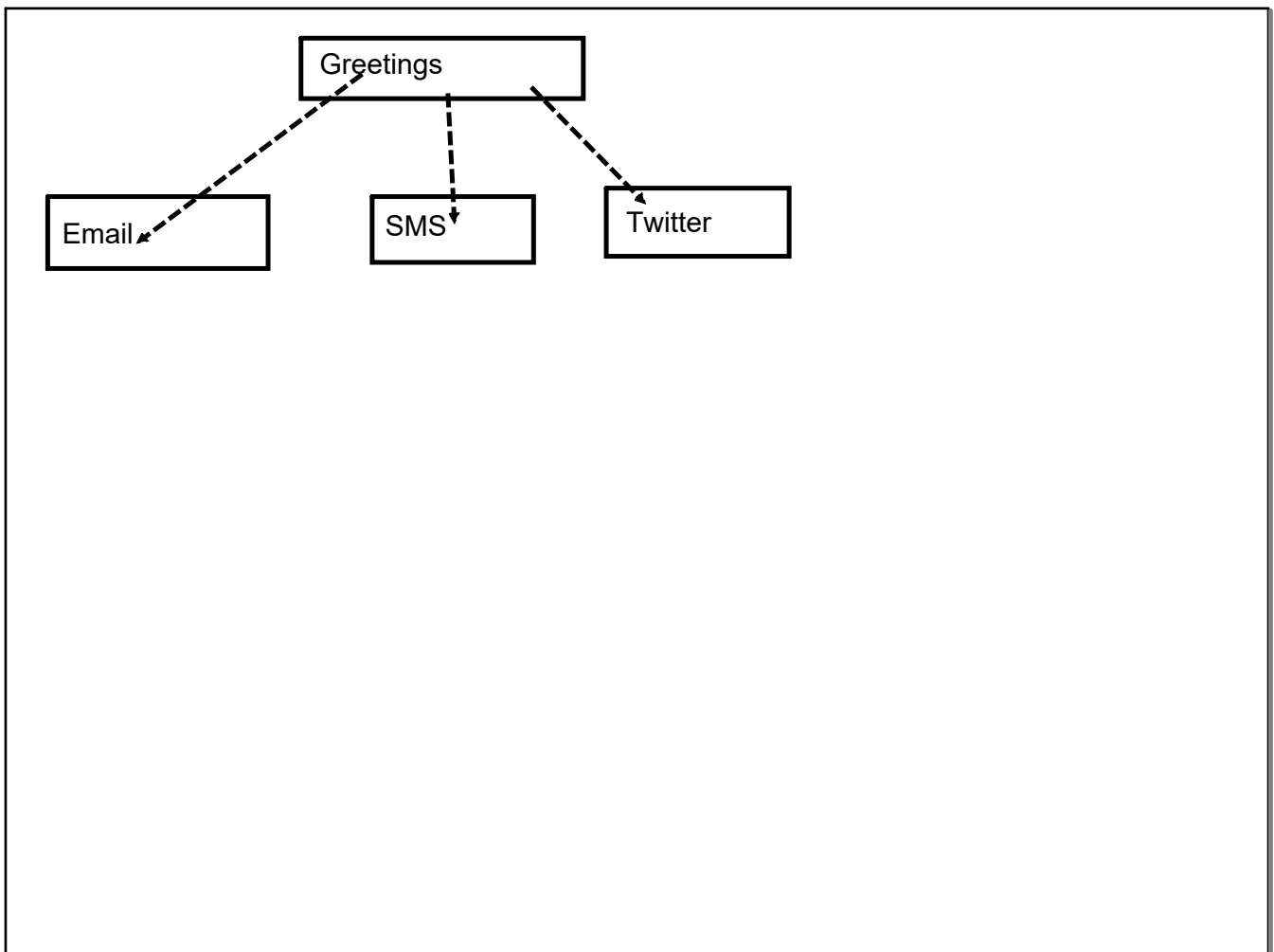
Contains only one abstract method

might have static method, default method (in any count)

jdk1.8 : special annotation

@FunctionalInterface (Compile time)/ optional

would restricts addition of any more abstract method



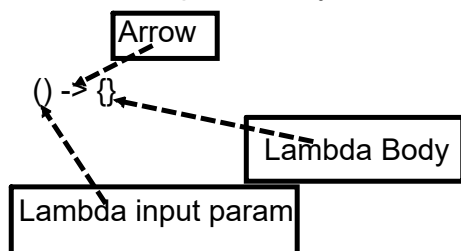
Interface can now act as a <Function type>

1. Functional Interface
2. Function definition that its reference is going to hold must match the prototype only abstract method inside it

Lambda expression:

1. anonymous function
2. method param, method body, return spec
3. Not encapsulated under any class
4. can be assigned to a variable, can be passed around

Lambda expression syntax : Compact



no class implementation

no object management

saved lots of runtime overheads

(msg) -> {}

msg -> {} // if single param no need to bind in para

(msg, other) -> {} // multiple, it is necessary

() -> {} // if no param

msg -> single instruction // no need to bind in braces

(a,b) -> a+b // return a+b

// if no braces then single stmt is by default associated with return

Functional Interface : (SAM)

=>Runnable

=>Comparator

```
interface Runnable{  
    void run()  
}
```

```
interface Comparator<T>{  
    int compare(T o1, T o2);  
}
```

New Functional Interfaces of Java 8

Lambdas connect themselves with specific signature

Java 8 has been introduced with group of
functional interfaces containing some very common prototype method

Usage of them has been updated in existing APIs

`java.util.function`

Consumer : BiConsumer

Predicate : BiPredicate

Function : BiFunction, UnaryOperator, BinaryOperator

Supplier

Consumer interface :

Single abstract method

void accept(<T>)