Java-8

=> Lambdas

Functional Programming

# those feature that define functional programming

# streams

# Executor (Future)

# Concurrency Collection

Style :

Traditional : Imperative

(HOW)

#exposing the steps how to perform an operation

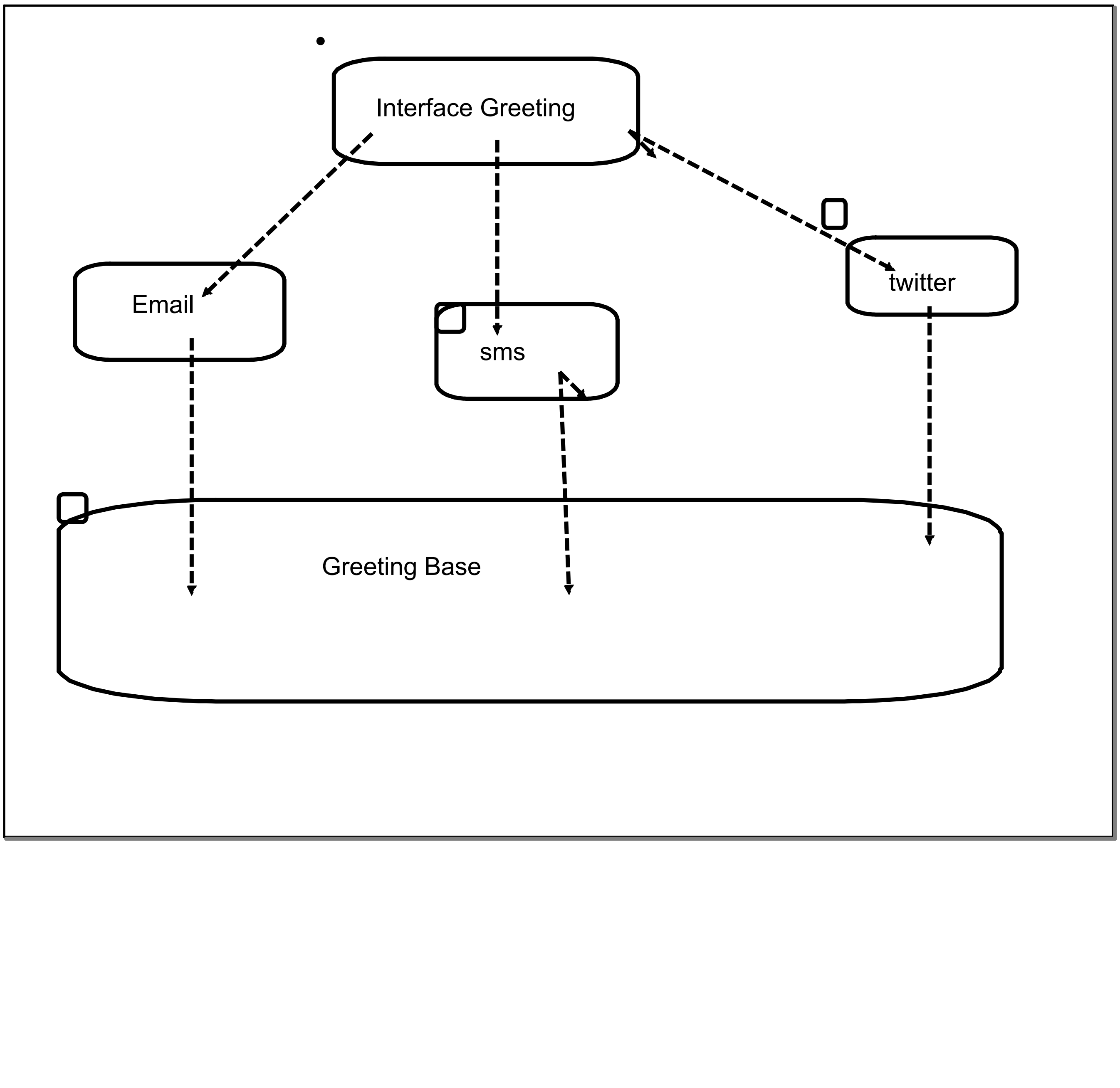
# embrace object mutability (not in sync with concurrency)

Functional : Declarative

(What) : result

immutability

Analogous SQL

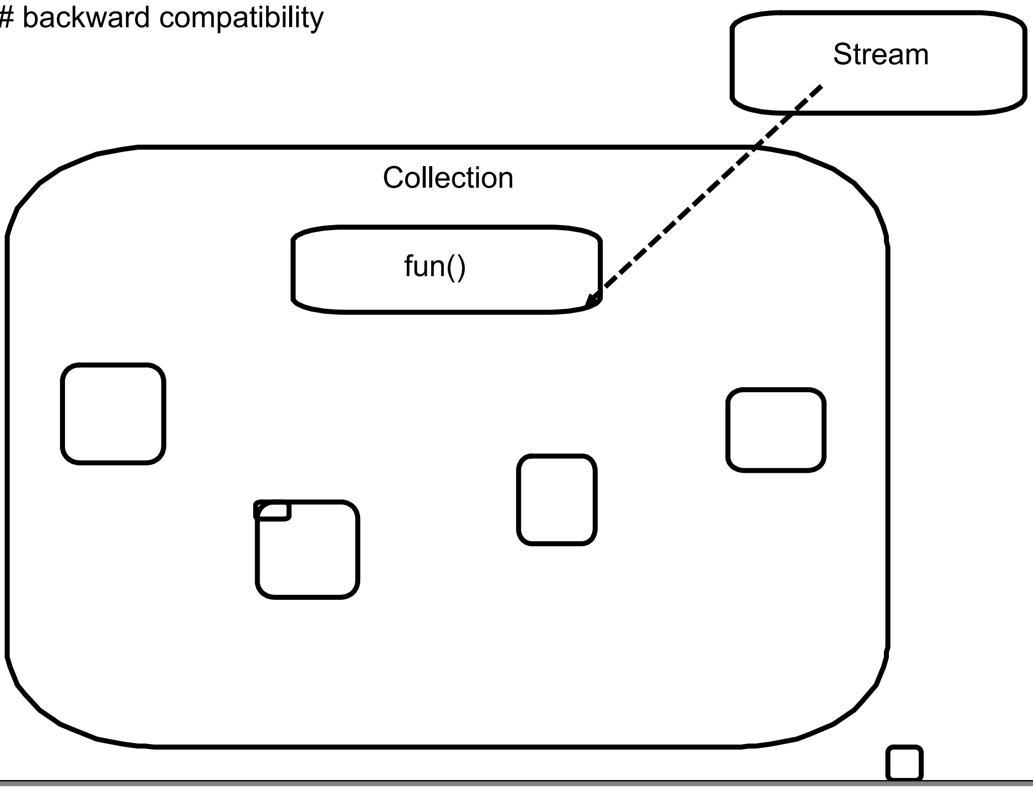

Interface Greeting
Email
sms
twitter
Greeting Base

Interface :

# default method (definition)

# multiple implementations can be inheritance

# backward compatibility



Escape from OOPs

independent Functions (not wrapped inside an object)

Relationship between interface and function

1. interface must have only one abstract method (any number of default/static) :

Functional Interface : Annotation @FunctionalInterface

2. single method signature must match with function implementation

Lambda expression

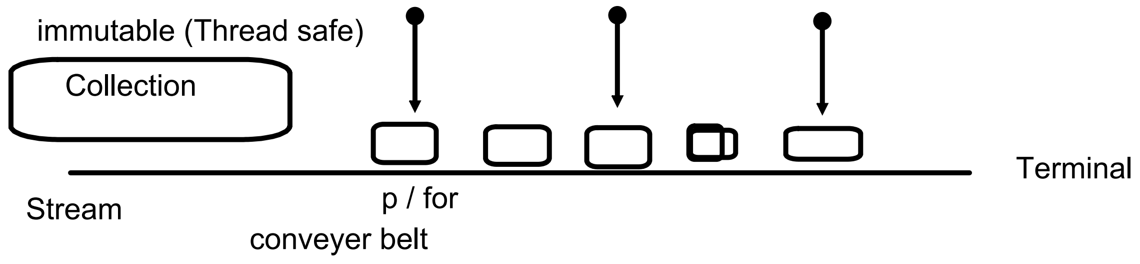
```
    (<arg1>,<arg2>) -> {
}


    arg1 -> {
    }


    () -> {
    }


    (<arg1>) -> <return> <single instruction>


    (a,b) -> <return>a+b;


    (a,b) -> {
        return a+b;
    }
```

```
Pre defined functional interfaces
    => Runnable
    => Comparator
Explicit Functional Interface
    # Consumer
        void accept(<>);
        DoubleConsumer() // specialized implementations on primitive
        BiConsumer
            void accept(<>,<>);
    # Predicate (test)
        boolean test(<>)
    # Supplier
        <> get()
    # Function
        <> apply(<>)
```

Stream :

not a data structure

immutable (Thread safe)



Stream : LAZY Processes

Stream can have 2 type of activities

intermidiate activities (filter/map/flat map)

Terminal activity : terminate/close the stream

forEach()

collect()

IF terminal activity is not present : stream will not initiate groupingBy(<return> Function(student))

return value : would become a group

Transforms

y map(x)

flatmap() : Collection into stream

map:

["",""]

["","",""]

["",""]

flat map

["","","","","","","",""]

return type fixed : stream of data passed as argument

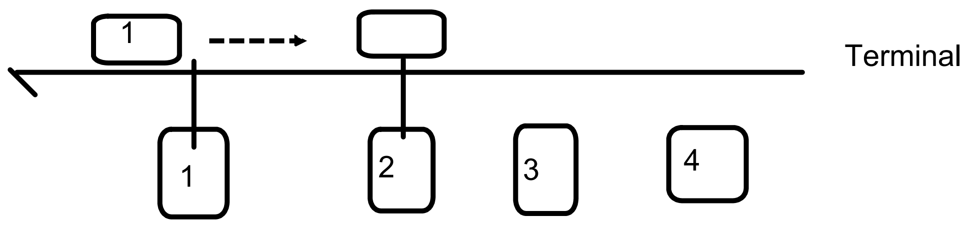
(Stream of) Multiple collection

into (Stream of)single collection

Stream :

# Sequential Stream

# Parallel Stream



# Parallel Streaming not commended if working on external mutable data (not thread safe)

# Activities that are inherently complex

Binary Operator : variant Function

y Function(x) : x and y can be of different type

z BinaryOperator(x,y) : x,y,z : must be of same type