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

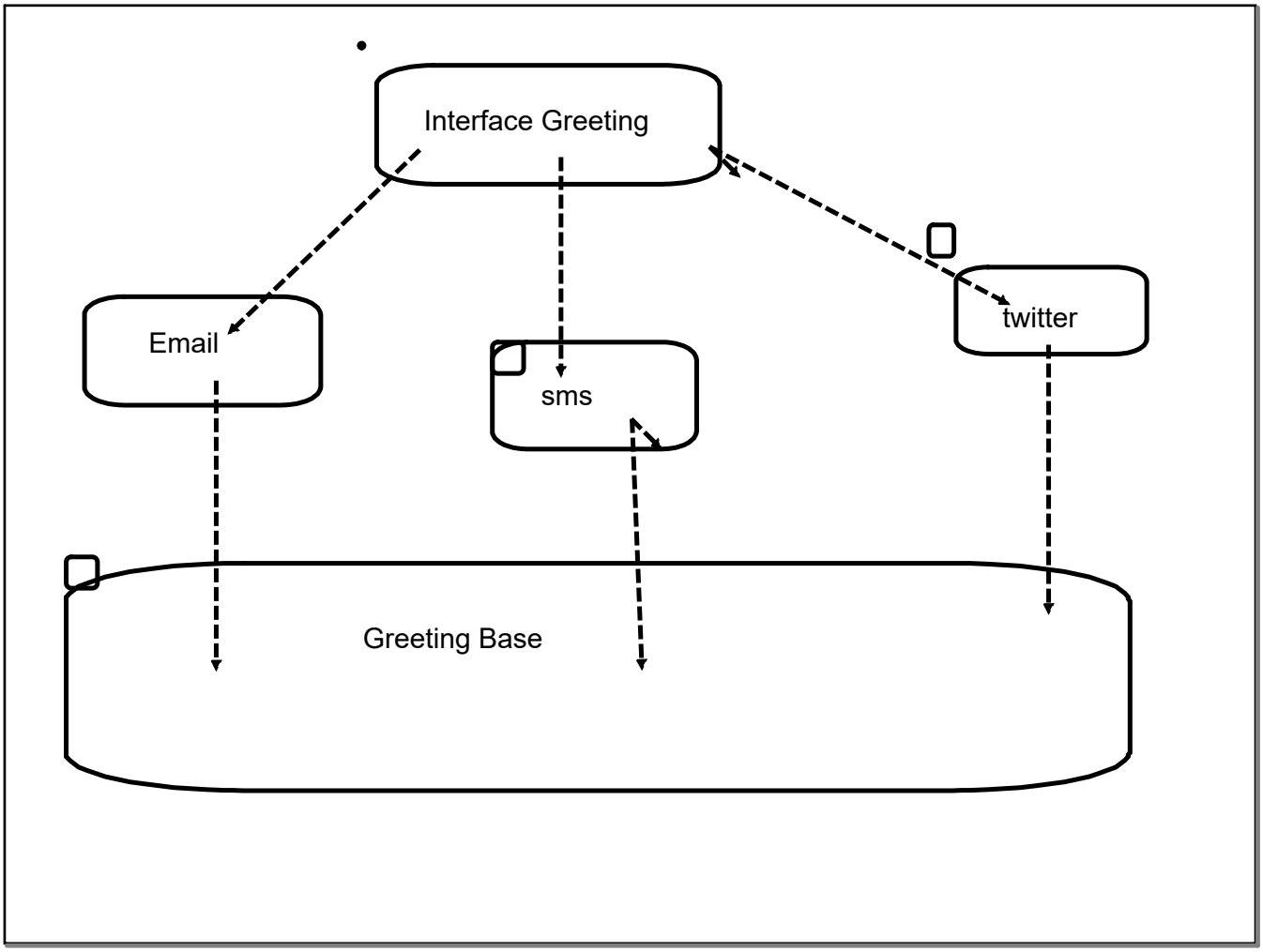
embrace object mutability (not in sync with concurrency)

Functional : Declarative

(What) : result

immutability

Analogous SQL

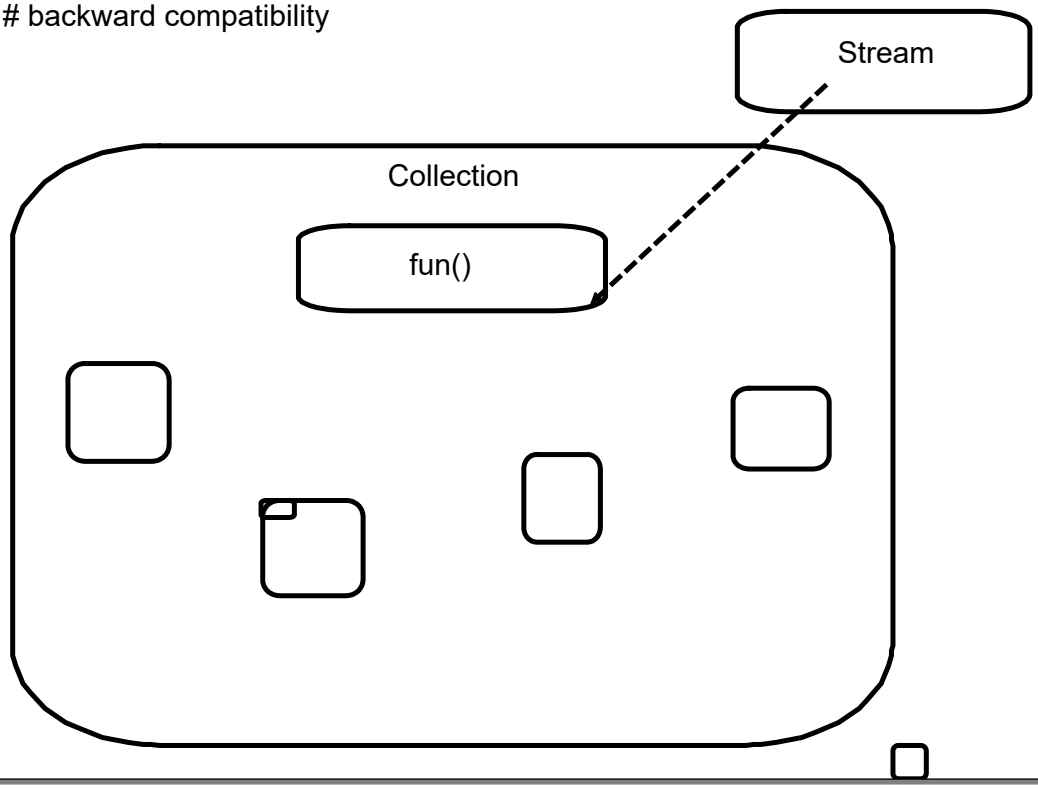

Interface Greeting
Email
sms
twitter
Greeting Base

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()

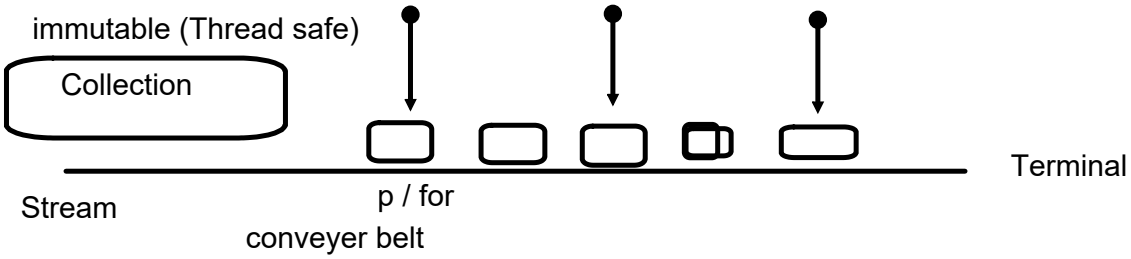
Function

<> 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

(Stream of) Multiple collection into (Stream of)single collection

Transforms

y map(x)

flatmap() : Collection into stream

map:

["",""]

["","",""]

["",""]

flat map

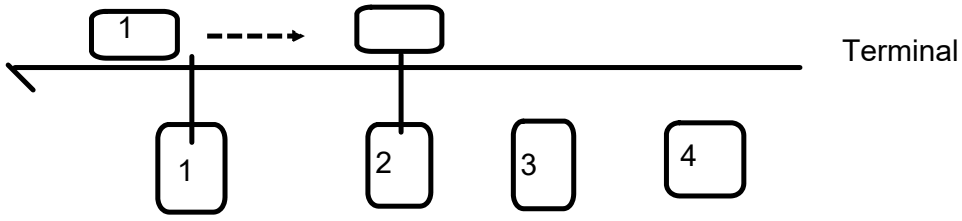
["","","","","","","","",""]

return type fixed : stream of data passed as argument

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

Multithreading :

interleaved (Threaded Multitasking)

1. Multiple activities waiting for I/O : that time can be used by tasks
2. Multi-core architecture of micro-processor

Base Interface :

Runnable (run)

Implementation:

Core Functionality of Multithreading (Thread)

inheriting Runnable

inheriting Thread

Traditional approach:

create, execute, manage

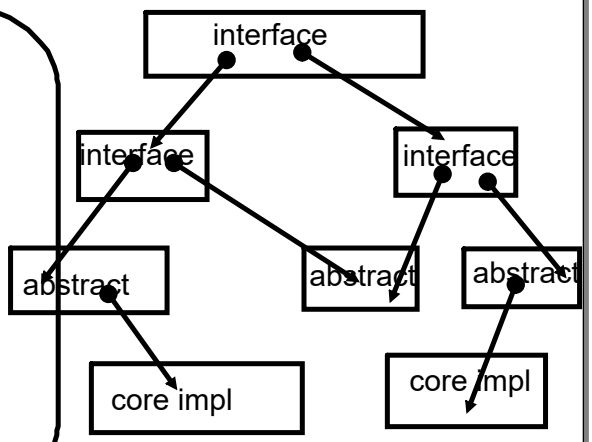
Thread per task

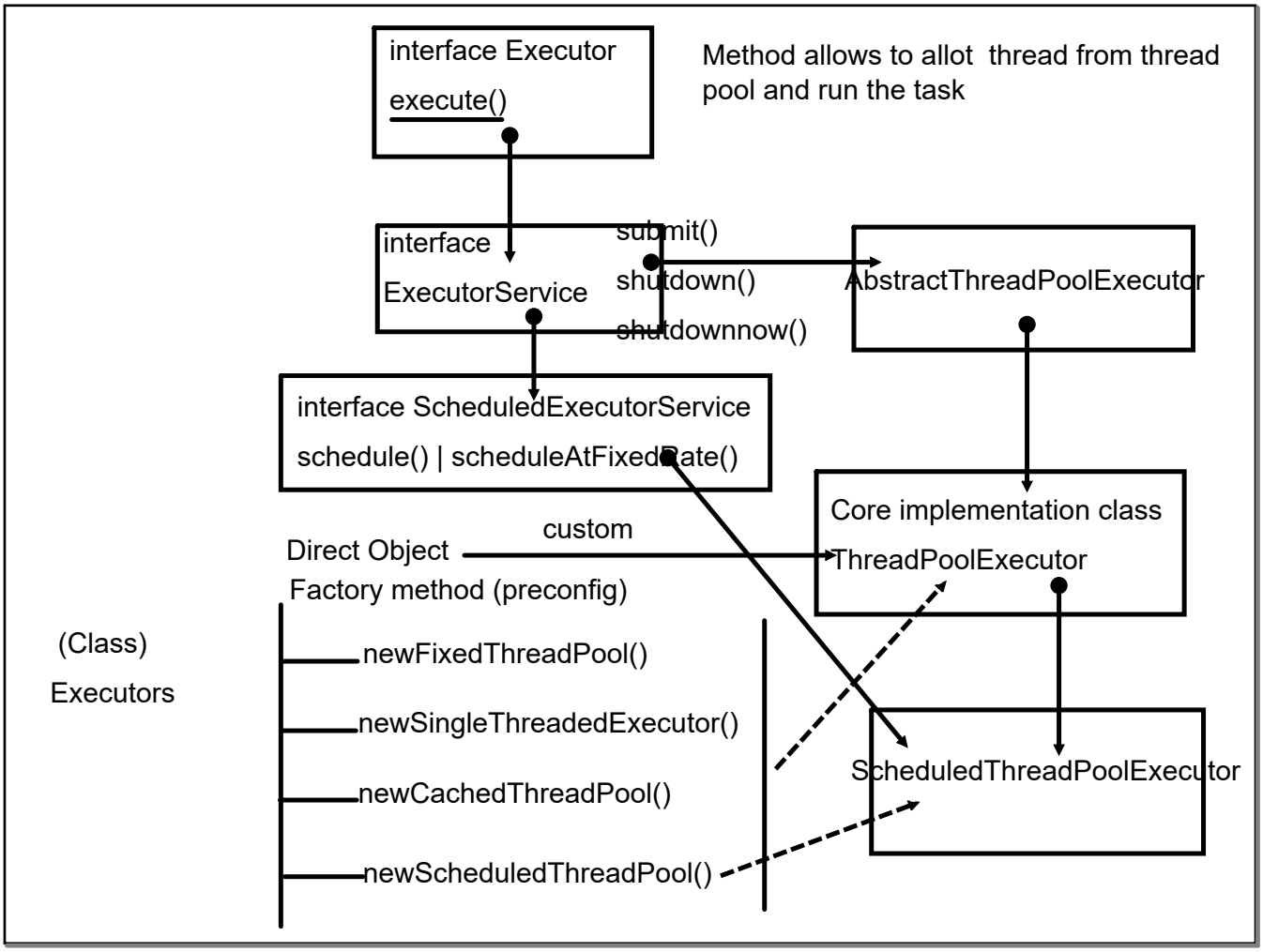
Not possible to keep a track of number of thread

Executor framework :

do all management + improve in performance

ThreadPool : create a predefined pool of thread




interface Executor
execute()
Method allows to allot thread from thread pool and run the task
interface
ExecutorService
submit()
shutdown()
shutdownnow()
AbstractThreadPoolExecutor
interface ScheduledExecutorService
schedule() | scheduleAtFixedRate()
Core implementation class
ThreadPoolExecutor
custom
Direct Object
Factory method (preconfig)
(Class)
Executors
newFixedThreadPool()
newSingleThreadedExecutor()
newCachedThreadPool()
newScheduledThreadPool()
ScheduledThreadPoolExecutor

Need to create instance of ThreadPoolExecutor

FixedThreadPool (number of thread are predefined(extra task alloted will added to queue)

CustomThreadPoolExecutor

<corePoolSize> : number of threads to always keep even if they are idle (2)

<maxPoolSize>: max no of thread (5)

<keepAliveTime> : time to wait before idle thread gets removed/released from thread pool

<TimeUnit> :

<queue capacity>: capacity of queue

<RejectedHAndler> : what to do if a task is rejected from queue

SingleThreadExecutor()

FixedThreadExecutor(1)

can change the thread capacity

CachedThreadPool() : Unbounded ThreadPool : Max Integer Val

if demand decreases : can tear down thread

default keep alive time : 1 min

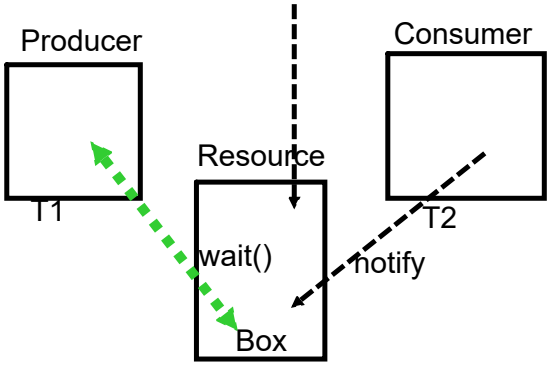
ScheduleThreadPool()

Thread returning some value

Special Interface Callable (call())

ThreadPool can work on Callable

wait/notify/notifyAll() : Object



ExecutorCompletionService

: will going to get results in order of completion of task

Future : blocking

CompletableFuture <callback : logic to follow when task is done>

Functional interfaces

Runnable

Callable

=> Supplier

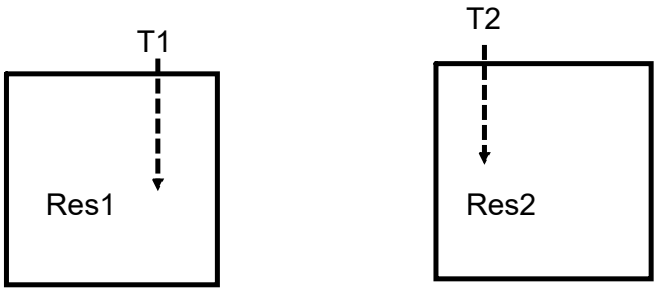
the method to associate a callback function

1. thenApply(Function); // transform
2. thenAccept(Consumer); // consuming and using

CompleatableFuture by default uses the inbuilt thread pool

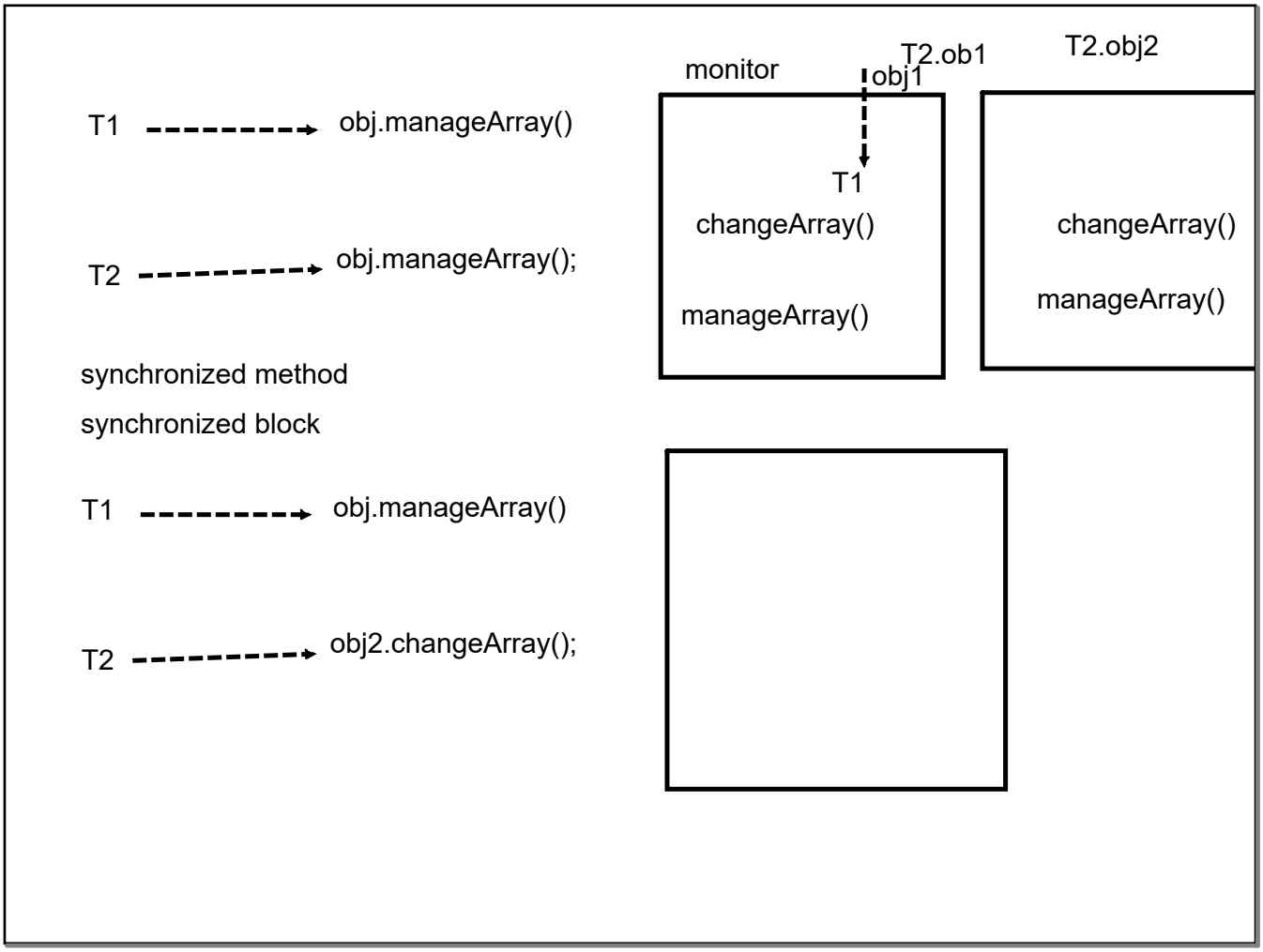
ForkJoinPool.commonPool();

Executor ThreadPool



Common Resource Shared among multiple threads (Thread safe)

Resolve Data inconsistency


T1
obj.manageArray()
T2
obj.manageArray();
synchronized method
synchronized block
T1
obj.manageArray()
T2
obj2.changeArray();
monitor
T2.ob1
obj1
T1
changeArray()
manageArray()
T2.obj2
changeArray()
manageArray()

locking :

=>wide spectrum locking : (synchronized...)

=>granular locking

java.util.concurrent.

API : Granular locking on resources

Collection API

1 .Traditional : 2

1. HashTable

2. Vector

2. To get a Thread safe variant of those class

Collections.concurrentList();

all methods are sync

Atomic operation : single CPU instruction

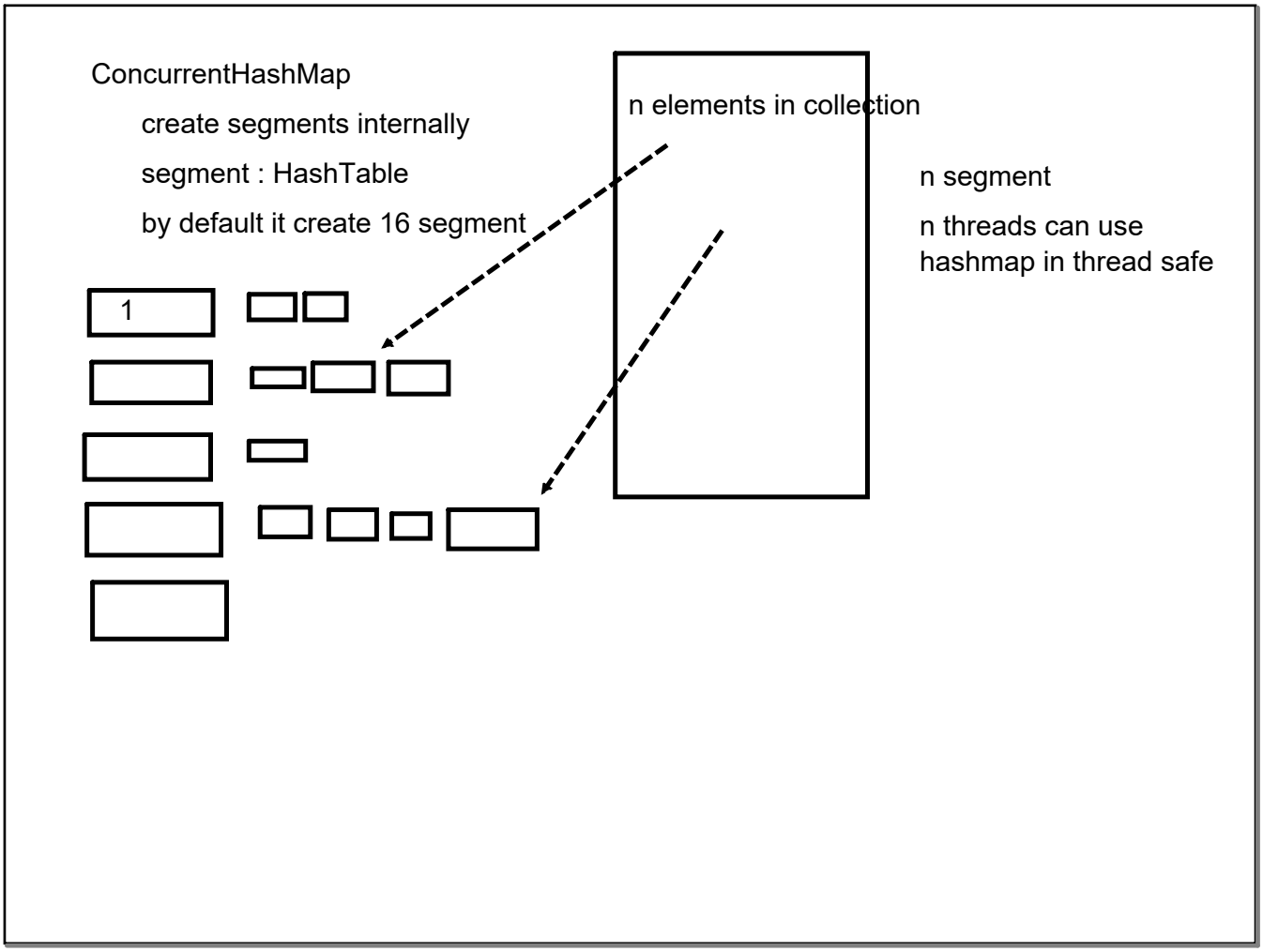
n=10; // Thread safe operations

assignment long/double are non-atomic

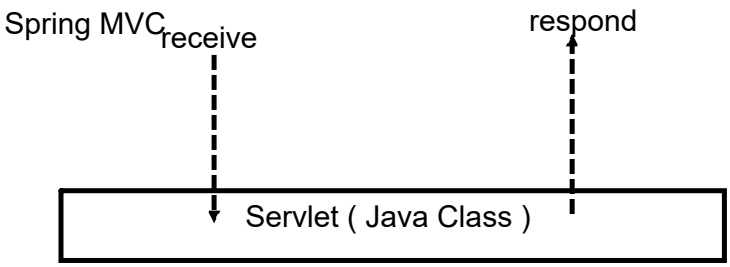
Concurent API : Focus on granular locking

Provides Atomic Variant of type : allow to convert non-atomic activities into atomic

multiple approach for ThreadSafety along with high level of concurrency


ConcurrentHashMap
create segments internally
segment : HashTable
by default it create 16 segment
1
n elements in collection
n segment
n threads can use
hashmap in thread safe

Servlet Technology



How to define java class as Servlet

Extends

HttpServlet/GenericServlet

GenericServlet : does not classifies between various HTTP Verbs

HttpServlet : can identify

GET/POST/PUT/DELETE/PATCH

Life Cycle

=> init

=> generic service() / http (doPost()/doGet()/doDelete()

=>destroy

Servlet Technology

(Controlled By web.xml)

lst request

Multiple request simultaneously

instance of servlet class is created

init phase gets activated

service() (Responsible for receive request/and respond)

Web Servers tends to cache the servlet object

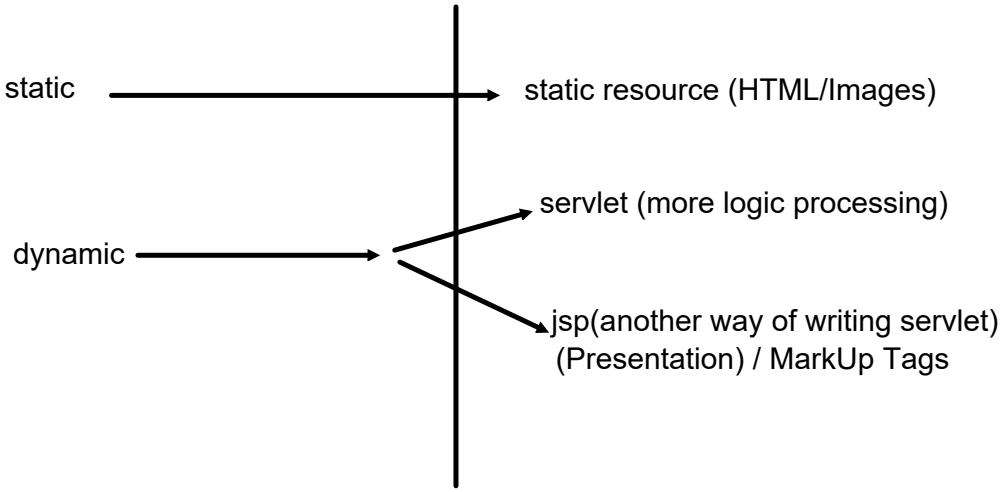
destroy

Object is released

service method and variant

have an access over HttpRequest/HttpResponse object

JSP : another way of writing servlet



Spring uses Servlet Technology:

But provides a high level abstraction over complexities/ boilerplate req / config and enhances the seperation of concerns