# Functional Interface

Single Abstract Method

1. It used to create objects by lambda expression.
2. @FunctionalInterface allows only one abstract method.
3. The annotation can be omitted if there is only one un-implemented method.
4. It can have 0~N default and static methods.

# Java8 functional interfaces:

Function

Consumer

Predicate

Supplier

Comparator

**default** Comparator<T> thenComparing(Comparator<? **super** T> other)

**default** <U> Comparator<T> thenComparing(

Function<? **super** T, ? **extends** U> keyExtractor,

Comparator<? **super** U> keyComparator)

**default** <U **extends** Comparable<? **super** U>> Comparator<T> thenComparing(

Function<? **super** T, ? **extends** U> keyExtractor)

**public** **static** <T> Comparator<T> nullsFirst(Comparator<? **super** T> comparator) {

**return** **new** Comparators.NullComparator<>(**true**, comparator);

}

**public** **static** <T> Comparator<T> nullsLast(Comparator<? **super** T> comparator) {

**return** **new** Comparators.NullComparator<>(**false**, comparator);

}

**public** **static** <T, U> Comparator<T> comparing(

Function<? **super** T, ? **extends** U> keyExtractor,

Comparator<? **super** U> keyComparator)

{

Objects.*requireNonNull*(keyExtractor);

Objects.*requireNonNull*(keyComparator);

**return** (Comparator<T> & Serializable)

(c1, c2) -> keyComparator.compare(keyExtractor.apply(c1),

keyExtractor.apply(c2));

}

BiFunction

BiConsumer

BiPredicate

IntFunction

ToIntFunction

ToIntBiFunction

Runnable

Callable

# Default method

Virtual extension methods

Why? What’s the purpose?

Extends, compile

Can it replace the abstract class?

Protect, polymorphism

What is polymorphism? How does it work in Java?

Family, hide

How to avoid instance a class?

Java9

Private

How do they co-exist?

Citi employee

Work

Eat

Dealtrax member

Client guy

Server guy

# Lambda

Stream

Intermediate operation

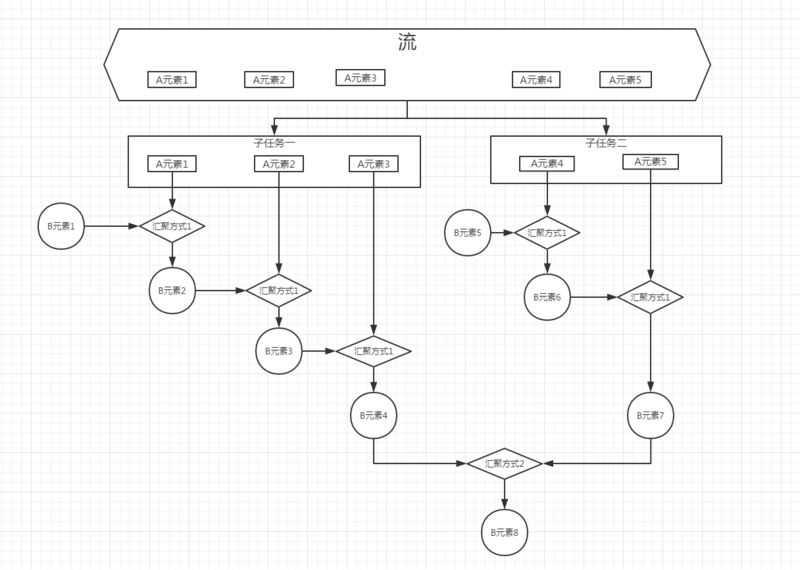
Terminal operation

Reduce

Accumulator

greatly reduced risk of data races.

combiner.apply(u, accumulator.apply(identity, t)) == accumulator.apply(u, t)



Optional

Map

Returns a stream consisting of the results of applying the given function to the elements of this stream.

<R> Stream<R> map(Function<? **super** T, ? **extends** R> mapper);

Collector

Result container

BiConsumer has no result

# Collections

Iterable

Collection

Queue

Deque

Double ended queue

ArrayDeque

// Double capacity if small; else grow by 50%, line 145

**int** jump = (oldCapacity < 64) ? (oldCapacity + 2) : (oldCapacity >> 1); >>>

relocate head

LinkedList

PriorityQueue

complete binary tree

<https://blog.csdn.net/u013309870/article/details/71189189>

removeAt, I don’t know why?

PriorityBlockingQueue

Vector

Stack

ArrayList

**public** **class** Vector<E>

**extends** AbstractList<E>

**implements** List<E>, RandomAccess, Cloneable, java.io.Serializable

**public** **class** ArrayList<E> **extends** AbstractList<E>

**implements** List<E>, RandomAccess, Cloneable, java.io.Serializable

**public** **class** HashMap<K,V> **extends** AbstractMap<K,V>

**implements** Map<K,V>, Cloneable, Serializable

**public** **class** HashSet<E>

**extends** AbstractSet<E>

**implements** Set<E>, Cloneable, java.io.Serializable

why?

# Specialized Interface

RandomAccess

ArrayList, LinkedList

Collections.binarySearch()

Cloneable

Serializable

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

Externalizable

PropertyChange

Listener

# HashCode & equals

Used to determine if two objects are same.

Efficient

Equals == true then hashCode == true.

hashCode == true then equals may not true.

HashSet, HashMap, HashTable

# Java Memory Model

JVM

Stack

Primitive

Instance

Frame when loading method

Heap

Object

Constant

Literal

Symbolic References

Wrap

Class/Interface

Method

Field

Static

-128~127

String

Benefit of constant pool?

Save space

Save time

# Primitive

Byte 1byte 8bit -27 ~ 27 - 1

Short 2 16 -215 ~ 215 - 1

Int 4 32 -231 ~ 231 - 1

Long 8 64 -263 ~ 263 - 1

Float 4 32 2-149 ~ (2-2-23)·2127

Double 8 64 2-1074 ~ (2-2-52)·21023

Char 2 16 0 ~ 216 – 1 65534?

Boolean 1 8

Chinese character

American Standard Code for Information Interchange

<https://www.cnblogs.com/lslk89/p/6898526.html>

# JVM

PermGen

MetaSpace

# Concurrent

CAS

Spin lock

ReentrantLock