30/7/2020

Functional Interface → Interface with only one abstract method

Lambda → Object without an Identity.

BiFunction<T, U>

Interface BinaryOperator<T> extends BiFunction<T, T, T>

Predicate<String> id = Predicate.isEqual(target);

Map/filter/reduce.

1st step: mapping.

List<Person> returns List<Integer>

Size of both lists → same.

2nd: Filtering

↳ Some Elements have been filtered out in process

Reduce → average

↳ reduction step.

typed Interface:

public Interface Stream<T> extends BaseStream<T, Stream<T>>

↳ An object that doesnot hold any data

Stream<Person> filtered =

Stream.filter(person → person.getAge() > 20)

Stream API details
Intermediary operations
forEach (Consumer) (Great Learning)
Peek (Consumer) (")
Filter (Predicate) (")

3 categories:
forEach & Peep.

Integer sum = stream.reduce(0, (age1, age2) -> age1 + age2);
1st Argument - Identity Element of reduction operation
2nd Argument - reduction operation, type of BinaryOperator<T>

Reductions:
↳ max(), min() & count()
Reduction is aggregation
→ no limit on amount of data being processed
→ Stream cannot be reused
Duration → amount of time between two Instant
Instant → Immutable
Methods:
↳ many cases where date is not an Instant.
LocalDate now = LocalDate.now();
LocalDate date = LocalDate.of(Year, Month.July, 04);
Period → Amount of time between two LocalDate
DateAdjuster → Useful to add/subtract amount of time to an Instant/LocalDate
LocalTime → time of day
ZonedTime → Time Zones all over Earth
Set<String> allZoneIds = ZoneId.getAvailableZoneIds();

Format a Date:
New Methods on Collection API:
→ stream() & parallelStream().