■ Java Streams & Functional Programming Cheat Sheet (Detailed)

1. Collectors (java.util.stream.Collectors)

Collector	Description	Example
toList()	Collect elements into a List	stream.collect(Collectors.toList())
toSet()	Collect elements into a Set	stream.collect(Collectors.toSet())
toMap	Collect into a Map	stream.collect(Collectors.toMap(String::length, s -> s))
joining()	Concatenate into a String	stream.collect(Collectors.joining(", "))
counting()	Count elements	stream.collect(Collectors.counting())
summingInt	Sum of int values	stream.collect(Collectors.summingInt(String::length))
averagingInt	Average of ints	stream.collect(Collectors.averagingInt(String::length))
maxBy	Maximum element	stream.collect(Collectors.maxBy(Comparator.naturalOrder()))
minBy	Minimum element	stream.collect(Collectors.minBy(Comparator.naturalOrder()))
groupingBy	Group into Map<K, List<T>>	stream.collect(Collectors.groupingBy(String::length))
partitioningBy	Split into 2 groups (true/false)	stream.collect(Collectors.partitioningBy(n -> n % 2 == 0))
mapping	Transform before collecting	stream.collect(Collectors.mapping(String::length, Collectors.toSet()))
reducing	Reduce while collecting	stream.collect(Collectors.reducing(0, String::length, Integer::sum))
collectingAndThen	Post-process after collecting	stream.collect(Collectors.collectingAndThen(Collectors.toList(), Colle

2. Comparator Helpers (java.util.Comparator)

Method	Description	Example
naturalOrder()	Sort ascending	sorted(Comparator.naturalOrder())
reverseOrder()	Sort descending	sorted(Comparator.reverseOrder())
comparing	Compare by property	sorted(Comparator.comparing(String::length))
comparingInt	Compare by int property	sorted(Comparator.comparingInt(String::length))
comparingLong	Compare by long property	sorted(Comparator.comparingLong(Person::getAge))
comparingDouble	Compare by double property	sorted(Comparator.comparingDouble(Product::getPrice))
thenComparing	Tie-breaker comparator	Comparator.comparingInt(String::length).thenComparing(Comparato
reversed()	Reverse comparator	Comparator.comparingInt(String::length).reversed()
nullsFirst	Nulls first in order	Comparator.nullsFirst(Comparator.naturalOrder())
nullsLast	Nulls last in order	Comparator.nullsLast(Comparator.naturalOrder())

3. Method References

Type	Syntax	Example

Static method	ClassName::methodName	Character::getNumericValue
Instance method (arbitrary)	ClassName::instanceMethod	String::toUpperCase
Instance method (particular object)	instance::methodName	System.out::println
Constructor	ClassName::new	ArrayList::new

4. Stream Intermediate Operations (lazy, return Stream)

Operation	Description	Example
map	Transform each element	list.stream().map(String::toUpperCase)
filter	Keep elements matching predicate	list.stream().filter(s -> s.startsWith("A"))
flatMap	Flatten nested streams	nested.stream().flatMap(List::stream)
distinct	Remove duplicates	list.stream().distinct()
sorted	Sort elements	list.stream().sorted()
peek	Debug/inspect elements	list.stream().peek(System.out::println)
limit	Take first N elements	list.stream().limit(3)
skip	Skip first N elements	list.stream().skip(2)

5. Stream Terminal Operations (eager, consume Stream)

Operation	Description	Example
forEach	Perform action for each element	list.stream().forEach(System.out::println)
collect	Collect into List/Set/Map	list.stream().collect(Collectors.toList())
reduce	Reduce to single value	list.stream().reduce(0, Integer::sum)
count	Count elements	list.stream().count()
min	Find min by comparator	list.stream().min(Comparator.naturalOrder())
max	Find max by comparator	list.stream().max(Comparator.naturalOrder())
anyMatch	Check if any element matches	list.stream().anyMatch(x -> x > 10)
allMatch	Check if all match	list.stream().allMatch(x -> x > 0)
noneMatch	Check if none match	list.stream().noneMatch(String::isEmpty)
findFirst	Return first element	list.stream().findFirst()
findAny	Return any element	list.stream().findAny()

6. Handy Examples

■ Reverse sort by string length: words.stream().sorted(Comparator.comparingInt(String::length).reversed()).toList();

■ Group by first letter: words.stream().collect(Collectors.groupingBy(s -> s.charAt(0)));

■ Partition numbers into even/odd: numbers.stream().collect(Collectors.partitioningBy(n -> n % 2 == 0));

■ Flatten nested lists: nested.stream().flatMap(List::stream).toList();

- Debug with peek: numbers.stream().peek(System.out::println).map(n -> n*2).toList();
- Find max length string: words.stream().max(Comparator.comparingInt(String::length));