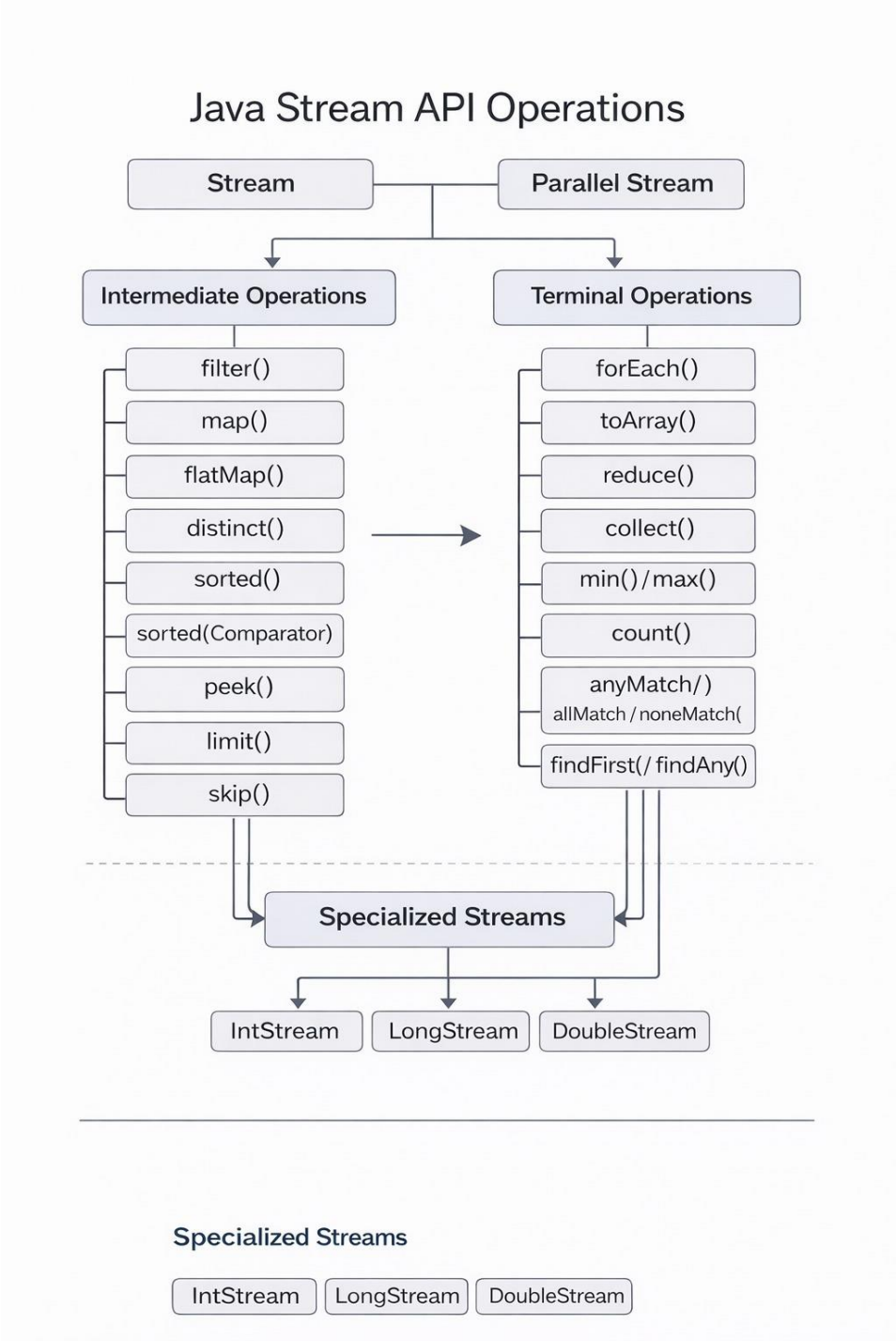
Explanation of the Stream API Diagram

The diagram visually organizes Java Stream API operations into three categories:



1. Intermediate Operations

These operations transform a stream into another stream. They are **lazy**, meaning they don't run until a terminal operation is called

Common Intermediate Operations

- **filter()** — Keeps elements that match a condition.
- **map()** — Transforms each element (e.g., String → length).
- **flatMap()** — Flattens nested structures (e.g., list of lists → single list).
- **distinct()** — Removes duplicates.
- **sorted()** — Sorts elements naturally.
- **sorted(Comparator)** — Sorts using a custom rule.
- **peek()** — Debugging; looks at elements without modifying them.
- **limit()** — Takes the first N elements.
- **skip()** — Skips the first N elements.

These operations can be chained together.

---

2. Terminal Operations

These operations **end the stream** and return a result (value, collection, or side-effect).

Common Terminal Operations

- **forEach()** — Performs an action for each element.
- **toArray()** — Converts the stream to an array.
- **reduce()** — Reduces the stream to a single value.
- **collect()** — Converts to Set, List, Map, etc.
- **min() / max()** — Finds smallest/largest element.
- **count()** — Counts elements.
- **anyMatch() / allMatch() / noneMatch()** — Boolean checks.
- **findFirst() / findAny()** — Retrieves elements.

Once a terminal operation runs, the stream is consumed and cannot be reused.

---

3. Specialized Streams

These are optimized for handling primitives:

- **IntStream**
- **LongStream**
- **DoubleStream**

They avoid boxing (Integer, Long, Double) → better performance.
