1. `stream()`:
    - Parametri: Nessuno
    - Ritorno: `Stream<T>`
    - Descrizione: Restituisce uno stream sequenziale con questo insieme come sua fonte.
2. `parallelStream()`:
    - Parametri: Nessuno
    - Ritorno: `Stream<T>`
    - Descrizione: Restituisce uno stream parallelo con questo insieme come sua fonte.
3. `filter(Predicate<? super T> predicate)`:
    - Parametri: `Predicate<? super T> predicate`
    - Ritorno: `Stream<T>`
    - Descrizione: Restituisce uno stream composto dagli elementi di questo stream che corrispondono al predicato dato.
4. `map(Function<? super T,? extends R> mapper)`:
    - Parametri: `Function<? super T,? extends R> mapper`
    - Ritorno: `Stream<R>`
    - Descrizione: Restituisce uno stream composto dai risultati dell'applicazione della funzione data agli elementi di questo stream.
5. `flatMap(Function<? super T,? extends Stream<? extends R>> mapper)`:
    - Parametri: `Function<? super T,? extends Stream<? extends R>> mapper`
    - Ritorno: `Stream<R>`
    - Descrizione: Restituisce uno stream composto dai risultati dell'applicazione della funzione data agli elementi di questo stream.
6. `reduce(BinaryOperator<T> accumulator)`:
    - Parametri: `BinaryOperator<T> accumulator`
    - Ritorno: `Optional<T>`
    - Descrizione: Esegue un'operazione di riduzione su gli elementi di questo stream utilizzando un operatore di accumulazione.
7. `collect(Collector<? super T,A,R> collector)`:
    - Parametri: `Collector<? super T,A,R> collector`
    - Ritorno: `R`
    - Descrizione: Esegue un'operazione di riduzione mutabile sui gli elementi di questo stream utilizzando un `Collector`.
8. `forEach(Consumer<? super T> action)`:
    - Parametri: `Consumer<? super T> action`
    - Ritorno: void
    - Descrizione: Esegue un'azione per ciascun elemento di questo stream.
9. `sorted()`:
    - Parametri: Nessuno
    - Ritorno: `Stream<T>`
    - Descrizione: Restituisce uno stream composto dagli elementi di questo stream, ordinati secondo l'ordine naturale.
10. `sorted(Comparator<? super T> comparator)`:
    - Parametri: `Comparator<? super T> comparator`
    - Ritorno: `Stream<T>`
    - Descrizione: Restituisce uno stream composto dagli elementi di questo stream, ordinati secondo il comparatore fornito.
11. `limit(long maxSize)`:
    - Parametri: `long maxSize`
    - Ritorno: `Stream<T>`
    - Descrizione: Restituisce uno stream composto dai primi n elementi di questo stream.
12. `count()`:
    - Parametri: Nessuno
    - Ritorno: long
    - Descrizione: Restituisce il conteggio degli elementi in questo stream.
13. `allMatch(Predicate<? super T> predicate)`:
    - Parametri: `Predicate<? super T> predicate`
    - Ritorno: boolean
    - Descrizione: Restituisce se tutti gli elementi di questo stream corrispondono al predicato fornito.

2. `anyMatch(Predicate<? super T> predicate)`:
    - Parametri: `Predicate<? super T> predicate`
    - Ritorno: boolean
    - Descrizione: Restituisce se qualsiasi elemento di questo stream corrisponde al predicato fornito.
3. `noneMatch(Predicate<? super T> predicate)`:
    - Parametri: `Predicate<? super T> predicate`
    - Ritorno: boolean
    - Descrizione: Restituisce se nessun elemento di questo stream corrisponde al predicato fornito.
4. `findFirst()`:
    - Parametri: Nessuno
    - Ritorno: `Optional<T>`
    - Descrizione: Restituisce un `Optional` che descrive il primo elemento di questo stream, o un `Optional` vuoto se lo stream è vuoto.
5. `findAny()`:
    - Parametri: Nessuno
    - Ritorno: `Optional<T>`
    - Descrizione: Restituisce un `Optional` che descrive un elemento qualsiasi di questo stream, o un `Optional` vuoto se lo stream è vuoto.
6. `max(Comparator<? super T> comparator)`:
    - Parametri: `Comparator<? super T> comparator`
    - Ritorno: `Optional<T>`
    - Descrizione: Restituisce il massimo elemento di questo stream secondo l'ordine naturale.
7. `min(Comparator<? super T> comparator)`:
    - Parametri: `Comparator<? super T> comparator`
    - Ritorno: `Optional<T>`
    - Descrizione: Restituisce il minimo elemento di questo stream secondo l'ordine naturale.
8. `distinct()`:
    - Parametri: Nessuno
    - Ritorno: `Stream<T>`
    - Descrizione: Restituisce uno stream composto dagli elementi distinti (secondo il metodo equals(Object)) di questo stream.
9. `peek(Consumer<? super T> action)`:
    - Parametri: `Consumer<? super T> action`
    - Ritorno: `Stream<T>`
    - Descrizione: Restituisce uno stream composto dagli elementi di questo stream, inoltre esegue un'azione su ciascun elemento mentre vengono consumati dallo stream pipeline.
10. `skip(long n)`:
    - Parametri: `long n`
    - Ritorno: `Stream<T>`
    - Descrizione: Restituisce uno stream composto dagli elementi rimanenti di questo stream dopo aver scartato i primi n elementi del flusso.
11. `toArray()`:
    - Parametri: Nessuno
    - Ritorno: Object[]
    - Descrizione: Restituisce un array contenente gli elementi di questo stream.
12. `toArray(IntFunction<A[]> generator)`:
    - Parametri: `IntFunction<A[]> generator`
    - Ritorno: A[]
    - Descrizione: Restituisce un array contenente gli elementi di questo stream, utilizzando il generatore fornito per allocare l'array restituito.
13. `asDoubleStream()`:
    - Parametri: Nessuno
    - Ritorno: `DoubleStream`
    - Descrizione: Restituisce un DoubleStream costituito dagli elementi di questo stream, convertiti in double.
14. `average()`:
    - Parametri: Nessuno
    - Ritorno: `OptionalDouble`
    - Descrizione: Restituisce un OptionalDouble che descrive la media aritmetica degli elementi di questo stream, o un optional vuoto se questo stream è vuoto.

15. `boxed()`:
    - Parametri: Nessuno
    - Ritorno: `Stream<Long>`
    - Descrizione: Restituisce uno Stream, converte uno stream di tipi primitivi in uno stream di oggetti del loro Wrapper.
16. `mapToInt(Function mapper)`:
    - Parametri: `Function mapper`
    - Ritorno: `IntStream`
    - Descrizione: Trasforma lo Stream in un IntStream applicando la funzione di mappatura fornita a ciascun elemento.
17. `mapToLong(Function mapper)`:
    - Parametri: `Function mapper`
    - Ritorno: `LongStream`
    - Descrizione: Trasforma lo Stream in un LongStream applicando la funzione di mappatura fornita a ciascun elemento.
18. `mapToDouble(Function mapper)`:
    - Parametri: `Function mapper`
    - Ritorno: `DoubleStream`
    - Descrizione: Trasforma lo Stream in un DoubleStream applicando la funzione di mappatura fornita a ciascun elemento.

Interfacce funzionali principali

1. *Predicate< T>*, che ha il metodo *boolean test(T t)*, è un'interfaccia perfetta per fare i test.
2. *Consumer< T>*, che ha il metodo *void accept(T t)*, è utilizzata per aggiornare lo stato di un oggetto.
3. *Supplier< T>*, che ha il metodo *T get()*,si adatta bene per gestire una factory e restituisce un'istanza del tipo parametro dichiarato, quindi è pensato per creare e restituire un oggetto.
4. *Function< T, R>*, che ha il metodo *R apply(T t)*, permette di astrarre il concetto classico di funzione dove c'è un input t e un output r.
5. *UnaryOperator< T>* è un'estensione di Function e ha il metodo T apply(T t) è un'operazione che viene fatta su un singolo parametro che è di I-O. Si adatta bene quando abbiamo bisogno di trasformare un oggetto, ad esempio una stringa che entra in un modo e esce in un altro.