Stream API

Introduzione agli Stream

Gli **Stream** in Java, introdotti in **Java 8**, sono strumenti potenti per elaborare e manipolare collezioni di dati in modo **dichiarativo e funzionale**. Non modificano le collezioni originali ma generano nuovi flussi di dati trasformati secondo le operazioni applicate.

Caratteristiche principali degli Stream:

- **Lazy Evaluation**: le operazioni vengono eseguite solo quando è necessaria un'operazione terminale.
- **Pipeline**: gli stream possono concatenare operazioni come filtraggio, mappatura e riduzione.
- **Parallellismo**: è possibile eseguire operazioni in parallelo con `.parallelStream()`.

Tabella dei Metodi degli Stream

Metodo	Descrizione
`limit(n)`	Prende solo i primi `n` elementi dello stream
`filter(Predicate<T>)`	Filtra gli elementi in base a una condizione
`forEach(Consumer<T>)`	Esegue un'azione per ogni elemento dello stream
`map(Function<T, R>)`	Trasforma gli elementi dello stream in altri valori
`flatMap(Function<T, Stream<R>>)`	Unisce più stream annidati in un unico stream
`anyMatch(Predicate<T>)`	Verifica se almeno un elemento soddisfa una condizione

Metodo	Descrizione
`allMatch(Predicate<T>)`	Verifica se tutti gli elementi soddisfano una condizione
`collect(Collector<T, A, R>)`	Raccoglie gli elementi in una collezione
`asList()`	Converte un array in una lista immutabile
`Collectors.joining(CharSequence)`	Concatena gli elementi in una stringa

Classe Base per gli Esempi

Tutti gli esempi utilizzeranno le seguenti classi:

```
class Autore {
    private String nome;

    public Autore(String nome) {
        this.nome = nome;
    }

    public String getNome() {
        return nome;
    }
}

class CasaEditrice {
    private String nome;

    public CasaEditrice(String nome) {
        this.nome = nome;
    }
```

```
    public String getNome() {
        return nome;
    }
}

class Libro {
    private String titolo;
    private Autore autore;
    private CasaEditrice casaEditrice;

    public Libro(String titolo, Autore autore, CasaEditrice casaEditrice) {
        this.titolo = titolo;
        this.autore = autore;
        this.casaEditrice = casaEditrice;
    }

    public String getTitolo() {
        return titolo;
    }

    public Autore getAutore() {
        return autore;
    }

    public CasaEditrice getCasaEditrice() {
        return casaEditrice;
    }
}
```

1. `limit(n)` - Prendi solo i primi N libri

```
List<Libro> primiDue = libri.stream()
    .limit(2)
    .collect(Collectors.toList());

primiDue.forEach(l -> System.out.println(l.getTitolo()));
```

2. `filter(Predicate<T>)` - Filtra i libri di un autore specifico

```
List<Libro> libriDiAutoreX = libri.stream()
    .filter(l -> l.getAutore().getNome().equals("Autore X"))
    .collect(Collectors.toList());

libriDiAutoreX.forEach(l -> System.out.println(l.getTitolo()));
```

3. `forEach(Consumer<T>)` - Stampa tutti i titoli

```
libri.stream()
    .forEach(l -> System.out.println(l.getTitolo()));
```

4. `map(Function<T, R>)` - Estrai i titoli dei libri

```
List<String> titoli = libri.stream()
    .map(Libro::getTitolo)
```

```
    .collect(Collectors.toList());

System.out.println(titoli);
```

5. `flatMap(Function<T, Stream<R>>)` - Unisci liste di libri in un unico stream

```
List<List<Libro>> librerie = List.of(libreria1, libreria2);

List<Libro> tuttiLibri = librerie.stream()
    .flatMap(List::stream)
    .collect(Collectors.toList());
```

6. `anyMatch(Predicate<T>)` - Verifica se almeno un libro è di un certo autore

```
boolean esisteLibroAutoreX = libri.stream()
    .anyMatch(l -> l.getAutore().getNome().equals("Autore X"));

System.out.println(esisteLibroAutoreX);
```

7. `allMatch(Predicate<T>)` - Verifica se tutti i libri sono della stessa casa editrice

```
boolean tuttiStessaCasaEditrice = libri.stream()
    .allMatch(l -> l.getCasaEditrice().getNome().equals("Editore X"));
```

```
System.out.println(tuttiStessaCasaEditrice);
```

8. `collect(Collector<T, A, R>)` - Raccogli i titoli dei libri in una lista

```
List<String> titoli = libri.stream()
    .map(Libro::getTitolo)
    .collect(Collectors.toList());

System.out.println(titoli);
```

9. `asList()` - Converti un array in una lista

```
String[] libriArray = {"Libro Uno", "Libro Due", "Libro Tre"};
List<String> libriList = Arrays.asList(libriArray);

System.out.println(libriList);
```

10. `Collectors.joining(CharSequence)` - Concatena i titoli in una stringa

```
String listaTitoli = libri.stream()
    .map(Libro::getTitolo)
    .collect(Collectors.joining(", "));
```

```
System.out.println(listaTitoli);
```