Java Stream API Advanced

Definizione delle Classi di Base per esempi

Classe `CasaEditrice`

```
public class CasaEditrice {
    private String nome;
    private String sede;

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

    public String getNome() { return nome; }
    public void setNome(String nome) { this.nome = nome; }
    public String getSede() { return sede; }
    public void setSede(String sede) { this.sede = sede; }

    @Override
    public String toString() {
        return "CasaEditrice{" + "nome='" + nome + '\'' + ",
        sede='" + sede + '\'' + '}';
    }
}
```

Classe `Autore`

```
public class Autore {
    private String nome;
    private String cognome;

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

    public String getNome() { return nome; }
    public void setNome(String nome) { this.nome = nome; }
    public String getCognome() { return cognome; }
    public void setCognome(String cognome) { this.cognome = cognome; }

    @Override
    public String toString() {
        return "Autore{" + "nome='" + nome +
```

```
        '\'' + ", cognome='" + cognome + '\'' + '}';
    }
}
```

Classe `Libro`

```
public class Libro {
    private String titolo;
    private Autore autore;
    private CasaEditrice casaEditrice;
    private double prezzo;
    private int pagine;

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

    public String getTitolo() { return titolo; }
    public void setTitolo(String titolo) { this.titolo = titolo; }
    public Autore getAutore() { return autore; }
    public void setAutore(Autore autore) { this.autore = autore; }
    public CasaEditrice getCasaEditrice() { return casaEditrice; }
    public void setCasaEditrice(CasaEditrice casaEditrice) {
    this.casaEditrice = casaEditrice; }
    public double getPrezzo() { return prezzo; }
    public void setPrezzo(double prezzo) { this.prezzo = prezzo; }
    public int getPagine() { return pagine; }
    public void setPagine(int pagine) { this.pagine = pagine; }

    @Override
    public String toString() {
        return "Libro{" + "titolo='" +
        titolo + '\'' + ", autore=" + autore + ",
        casaEditrice=" + casaEditrice + ",
        prezzo=" + prezzo + ", pagine=" + pagine + '}';
    }
}
```

Metodi dello Stream API con Esempi

1. `Collectors.toList()`

Descrizione: Converte uno stream in una lista.

```
List<Libro> listaLibri = libri.stream().collect(Collectors.toList());
```

2. `Collectors.toSet()`

Descrizione: Converte uno stream in un set (senza duplicati).

```
Set<Libro> setLibri = libri.stream()
.collect(Collectors.toSet());
```

3. `Collectors.toMap()`

Descrizione: Crea una mappa a partire da una chiave e un valore.

```
Map<String, Libro> libriMap = libri.stream()
.collect(Collectors.toMap(Libro::getTitolo, libro -> libro));
```

4. `Collectors.groupingBy()`

Descrizione: Raggruppa i libri in base a un criterio.

```
Map<String, List<Libro>> libriPerAutore = libri.stream()
.collect(Collectors.groupingBy(libro -> libro.getAutore().getNome()));
```

5. `Collectors.summingInt()`

Descrizione: Somma un valore intero di un attributo.

```
int totalePagine = libri.stream()
.collect(Collectors.summingInt(Libro::getPagine));
```

6. `Collectors.summingDouble()`

Descrizione: Somma un valore double.

```
double totalePrezzo = libri.stream()
.collect(Collectors.summingDouble(Libro::getPrezzo));
```

7. `Collectors.averagingInt()`

Descrizione: Calcola la media di un valore intero.

```
double mediaPagine = libri.stream()
.collect(Collectors.averagingInt(Libro::getPagine));
```

8. `Collectors.joining()`

Descrizione: Unisce stringhe in un'unica stringa.

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

9. `Collectors.partitioningBy()`

Descrizione: Divide in due gruppi basati su un predicato.

```
Map<Boolean, List<Libro>> libriCostosi = libri.stream()
.collect(Collectors.partitioningBy(libro -> libro.getPrezzo() > 20));
```

10. `Comparator su String`

Descrizione: Ordina libri per titolo.

```
List<Libro> libriOrdinati = libri.stream()
.sorted(Comparator.comparing(Libro::getTitolo))
.collect(Collectors.toList());
```

11. `limit()`

Descrizione: Limita il numero di elementi dello stream.

```
List<Libro> primiTre = libri.stream()
.limit(3)
.collect(Collectors.toList());
```

12. `mapToDouble()`

Descrizione: Mappa un valore numerico.

```
DoubleSummaryStatistics stats = libri.stream()
```

```
.mapToDouble(Libro::getPrezzo).summaryStatistics();
```

13. `summaryStatistics()`

Descrizione: Raccoglie statistiche di un valore numerico.

```
IntSummaryStatistics statistiche = libri.stream()
.mapToInt(Libro::getPagine)
.summaryStatistics();

System.out.println("Min: " + statistiche.getMin());
System.out.println("Max: " + statistiche.getMax());
System.out.println("Media: " + statistiche.getAverage());
System.out.println("Somma: " + statistiche.getSum());
System.out.println("Conteggio: " + statistiche.getCount());
```