Progetto Java - Stream

Creazione del Progetto

1. **Aprire IntelliJ IDEA** e selezionare **New Project**.
2. Selezionare **Java** come linguaggio e **Maven** come sistema di build.
3. Premere **Create**.

Struttura del Progetto

```
progetto-libri
├── src
│   └── main
│       └── java
│           └── it
│               └── epicode
│                   └── esercizi
│                       ├── Libro.java
│                       ├── FilterExample.java
│                       ├── MapExample.java
│                       ├── SortedExample.java
│                       ├── CompleteStreamExample.java
│                       └── StreamExamples.java
├── pom.xml
```

- `src/main/java` : Codice sorgente principale.
- `pom.xml` : Configurazione Maven.

Istruzioni per creare nuovi file Java

- Fare clic destro sulla cartella `it.epicode.esercizi` > **New** > **Java Class**
- Inserire il nome della classe (es. `FilterExample`, `MapExample`, `SortedExample`, ecc.)

Classe Libro

```java
package it.epicode;

public class Libro {
    private String titolo;
    private String autore;
    private String casaEditrice;
    private String categoria;

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

    public String getTitolo() {
        return titolo;
    }

    public void setTitolo(String titolo) {
        this.titolo = titolo;
    }

    public String getAutore() {
        return autore;
    }
```

```
    public void setAutore(String autore) {
        this.autore = autore;
    }

    public String getCasaEditrice() {
        return casaEditrice;
    }

    public void setCasaEditrice(String casaEditrice) {
        this.casaEditrice = casaEditrice;
    }

    public String getCategoria() {
        return categoria;
    }

    public void setCategoria(String categoria) {
        this.categoria = categoria;
    }
}
```

Esempi Stream

FilterExample.java

```
package it.epicode.esercizi;

import java.util.*;
import java.util.stream.*;

public class FilterExample {
    public static void main(String[] args) {
```

```
        List<Libro> libri = Arrays.asList(
            new Libro("Java Fundamentals", "Mario Rossi", "Epicode Press", "Informatica"),
            new Libro("Fairy Tales", "Carla Verdi", "Kids Books", "Bambini")
        );

        // filter: seleziona libri categoria "Informatica"
        libri.stream()
            .filter(libro -> libro.getCategoria().equals("Informatica"))
            .forEach(libro -> System.out.println(libro.getTitolo()));
    }
}
```

MapExample.java

```
package it.epicode.esercizi;

import java.util.*;

public class MapExample {
    public static void main(String[] args) {
        List<Libro> libri = Arrays.asList(
            new Libro("Java Fundamentals", "Mario Rossi", "Epicode Press", "Informatica"),
            new Libro("Fairy Tales", "Carla Verdi", "Kids Books", "Bambini")
        );

        // map: trasforma oggetto Libro nel suo titolo (String)
        libri.stream()
            .map(Libro::getTitolo)
            .forEach(System.out::println); // stampa ogni titolo
    }
}
```

SortedExample.java

```java
package it.epicode.esercizi;

import java.util.*;

public class SortedExample {
    public static void main(String[] args) {
        List<Libro> libri = Arrays.asList(
            new Libro("Java Fundamentals", "Mario Rossi", "Epicode Press", "Informatica"),
            new Libro("Fairy Tales", "Carla Verdi", "Kids Books", "Bambini")
        );

        // sorted: ordina i libri per autore in ordine alfabetico
        libri.stream()
            .sorted(Comparator.comparing(Libro::getAutore))
            .forEach(libro -> System.out.println(libro.getAutore())); // stampa ogni autore
    }
}
```

CompleteStreamExample.java

```java
package it.epicode.esercizi;

import java.util.*;

public class CompleteStreamExample {
    public static void main(String[] args) {
        List<Libro> libri = Arrays.asList(
            new Libro("Java Fundamentals", "Mario Rossi", "Epicode Press", "Informatica"),
            new Libro("Fairy Tales", "Carla Verdi", "Kids Books", "Bambini"),
            new Libro("World History", "Anna Bianchi", "History Press", "Storia")
        );

        // Combinazione completa di operazioni stream
```

```
        libri.stream()
            .filter(libro -> libro.getCategoria().equals("Bambini"))
            .map(Libro::getTitolo)
            .sorted()
            .forEach(System.out::println); // stampa titoli ordinati di libri categoria "Bambini"
    }
}
```

Spiegazione `.forEach(System.out::println)`

- Il metodo `.forEach` permette di eseguire una operazione per ogni elemento dello stream.
- `System.out::println` utilizza il "method reference" (operatore `::`) per riferirsi direttamente al metodo `println`. Questo rende il codice più pulito rispetto a una lambda come `elemento -> System.out.println(elemento)`.
- L'operatore `::` indica quindi il riferimento diretto a un metodo, in questo caso, il metodo di stampa standard `println`.

StreamExamples.java

```
package it.epicode.esercizi;

import java.util.*;
import java.util.stream.*;

public class StreamExamples {
    public static void main(String[] args) {
        List<Libro> libri = Arrays.asList(
            new Libro("Java Fundamentals", "Mario Rossi", "Epicode Press", "Informatica"),
            new Libro("Fairy Tales", "Carla Verdi", "Kids Books", "Bambini"),
            new Libro("World History", "Anna Bianchi", "History Press", "Storia")
        );

        // Esempio combinato di filter + map + sorted + forEach
        libri.stream()
```

```
            .filter(libro -> !libro.getCategoria().equals("Informatica"))
            .map(Libro::getTitolo)
            .sorted()
            .forEach(System.out::println);
+
    }
}
```

Spiegazione `.forEach(System.out::println)`

- Il metodo `.forEach` permette di eseguire una operazione per ogni elemento dello stream.
- `System.out::println` utilizza il "method reference" (operatore `::`) per riferirsi direttamente al metodo `println`. Questo rende il codice più pulito rispetto a una lambda come `elemento -> System.out.println(elemento)`.
- L'operatore `::` indica quindi il riferimento diretto a un metodo, in questo caso, il metodo di stampa standard `println`.