

```

package com.example;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import java.time.Year;
import java.util.Objects;

public class Book {

    @JsonCreator
    public Book(
        @JsonProperty("author") String author,
        @JsonProperty("title") String title,
        @JsonProperty("publishYear") Year publishYear,
        @JsonProperty("pageCount") int pageCount) {

        this.author = author;
        this.title = title;
        this.publishYear = publishYear;
        this.pageCount = pageCount;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) {
            return true;
        }

        if (!(o instanceof Book book)) {
            return false;
        }

        return pageCount == book.pageCount
            && author.equals(book.author)
            && title.equals(book.title)
            && publishYear.equals(book.publishYear);
    }
}

```

```

        && publishYear.equals(book.publishYear);
    }

    @Override
    public int hashCode() {
        return Objects.hash(author, title, publishYear, pageCount);
    }

    @Override
    public String toString() {
        return "Book{"
            + "author='"
            + author
            + '\''
            + ", title='"
            + title
            + '\''
            + ", publishYear="
            + publishYear
            + ", pageCount="
            + pageCount
            + '}';
    }

    public int getPageCount() {
        return pageCount;
    }

    public Year getPublishYear() {
        return publishYear;
    }

    private final String author;
    private final String title;
    private final Year publishYear;
    private final int pageCount;
}

```

```

1 package com.example;
2
3 import com.fasterxml.jackson.annotation.JsonCreator;
4 import com.fasterxml.jackson.annotation.JsonProperty;
5 import java.util.List;
6
7 public class Student {
8     @JsonCreator
9     public Student(
10         @JsonProperty("name") String name,
11         @JsonProperty("surname") String surname,
12         @JsonProperty("books") List<Book> books) {
13
14         this.name = name;
15         this.surname = surname;
16         this.books = List.copyOf(books);
17     }
18
19     @Override
20     public String toString() {
21         StringBuilder booksString = new StringBuilder();
22
23         for (Book book : books) {
24             booksString.append("\n").append(book);
25         }
26
27         return "Student{\n"
28             + "    name=' "
29             + name
30             + "'\n"
31             + "    surname=' "
32             + surname
33             + "'\n"
34             + "    books="
35             + booksString
36             + "\n}";
37     }

```

```

37 public Student(
36     @JsonProperty("name") String name,
35     @JsonProperty("surname") String surname,
34     @JsonProperty("books") List<Book> books) {
33
32     this.name = name;
31     this.surname = surname;
30     this.books = List.copyOf(books);
29 }
28
27 @Override
26 public String toString() {
25     StringBuilder booksString = new StringBuilder();
24
23     for (Book book : books) {
22         booksString.append("\n").append(book);
21     }
20
19     return "Student{\n"
18         + "    name=' "
17         + name
16         + "'\n"
15         + "    surname=' "
14         + surname
13         + "'\n"
12         + "    books="
11         + booksString
10         + "\n}";
9 }
8
7 public List<Book> getBooks() {
6     return books;
5 }
4
3 private final String name;
2 private final String surname;
1 private final List<Book> books;
47 }

```

```
1 package com.example;
2
3 import com.fasterxml.jackson.core.type.TypeReference;
4 import com.fasterxml.jackson.databind.ObjectMapper;
5 import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;
6 import java.io.IOException;
7 import java.nio.file.Path;
8 import java.util.List;
9
10 public final class StudentReader {
11     public static List<Student> read(Path path) throws IOException {
12         return MAPPER.readValue(path.toFile(), new TypeReference<List<Student>>() {});
13     }
14
15     private StudentReader() {}
16
17     private static final ObjectMapper MAPPER =
18         new ObjectMapper().registerModule(new JavaTimeModule());
19 }
```

```

package com.example;

import java.io.IOException;
import java.nio.file.Path;
import java.time.Year;
import java.util.Comparator;
import java.util.List;

public class App {

    public static void main(String[] args) {

        try {

            Path studentsJsonFile = Path.of("students.json");
            List<Student> students = StudentReader.read(path: studentsJsonFile);

            // Условие использования одного stream сохраняется
            students.stream()

                // Вывести в консоль каждого студента (переопределите toString)
                .peek(System.out::println)

                // Получить для каждого студента список книг
                .map(s -> s.getBooks()) // Stream<List<Book>>

                // Получить книги
                .flatMap(List::stream) // Stream<Book>;

                // Отсортировать книги по количеству страниц (Не забывайте про
условия для сравнения
                // объектов)
                .sorted(Comparator.comparingInt(Book::getPageCount))

                // Оставить только уникальные книги
                .distinct()

                // Отфильтровать книги, оставив только те, которые были

```

```

                // Отфильтровать книги, оставив только те, которые были выпущены после 2000 года
                .filter(b -> b.getPublishYear().isAfter(Year.of(2000)))

                // Ограничить стрим на 3 элементах
                .limit(3)

                // Получить из книг годы выпуска
                .map(Book::getPublishYear)

                // При помощи методов короткого замыкания (почитайте самостоятельно что это
такое) вернуть
                // Optional от года
                .findFirst()

                // При помощи методов получения значения из Optional вывести в консоль год
выпуска
                // найденной книги, либо запись о том, что такая книга отсутствует
                .ifPresentOrElse(
                    year -> System.out.println("Год выпуска: " + year),
                    () -> System.out.println("Подходящая книга отсутствует"));

        } catch (IOException e) {

            System.err.println("Сударь, вы передали неправильный файл!");
            e.printStackTrace();
            System.exit(1);

        } catch (Exception e) {

            System.err.println("Ну тут мои полномочия всё!");
            e.printStackTrace();
            System.exit(1);

        }

        System.out.println("Выполнение программы завершено");
    }
}

```

```
1 [{
2   {
3     "name": "Aaa",
4     "surname": "Bbb",
5     "books": [
6       {
7         "author": "Aaa Aaa",
8         "title": "Book Aaa",
9         "publishYear": "2027",
10        "pageCount": 500
11      },
12      {
13        "author": "Bbb Bbb",
14        "title": "Book Bbb",
15        "publishYear": "2018",
16        "pageCount": 200
17      },
18      {
19        "author": "Ccc Ccc",
20        "title": "Book Ccc",
21        "publishYear": "2024",
22        "pageCount": 900
23      },
24      {
25        "author": "Ddd Ddd",
26        "title": "Book Ddd",
27        "publishYear": "2015",
28        "pageCount": 400
29      },
30      {
31        "author": "Ddd Ddd",
32        "title": "Book Ddd",
33        "publishYear": "2015",
34        "pageCount": 400
35      }
36    ]
37  },
```

```
38  {
39    "name": "Ccc",
40    "surname": "Ddd",
41    "books": [
42      {
43        "author": "Eee Eee",
44        "title": "Book Eee",
45        "publishYear": "1999",
46        "pageCount": 300
47      },
48      {
49        "author": "Fff Fff",
50        "title": "Book Fff",
51        "publishYear": "2030",
52        "pageCount": 1200
53      },
54      {
55        "author": "Ggg Ggg",
56        "title": "Book Ggg",
57        "publishYear": "2005",
58        "pageCount": 100
59      },
60      {
61        "author": "Hhh Hhh",
62        "title": "Book Hhh",
63        "publishYear": "2021",
64        "pageCount": 800
65      },
66      {
67        "author": "Iii Iii",
68        "title": "Book Iii",
69        "publishYear": "2012",
70        "pageCount": 600
71      }
72    ]
73  },
74  {
75    "name": "Eee",
```

```
76  {
77    "name": "Eee",
78    "surname": "Fff",
79    "books": [
80      {
81        "author": "Aaa Aaa",
82        "title": "Book Aaa",
83        "publishYear": "2027",
84        "pageCount": 500
85      },
86      {
87        "author": "Bbb Bbb",
88        "title": "Book Bbb",
89        "publishYear": "2018",
90        "pageCount": 200
91      },
92      {
93        "author": "Jjj Jjj",
94        "title": "Book Jjj",
95        "publishYear": "2001",
96        "pageCount": 1000
97      },
98      {
99        "author": "Kkk Kkk",
100       "title": "Book Kkk",
101       "publishYear": "1995",
102       "pageCount": 700
103     },
104     {
105       "author": "Lll Lll",
106       "title": "Book Lll",
107       "publishYear": "2025",
108       "pageCount": 1100
109     }
110   ]
111 }
```

```
1 Book{author='Ggg Ggg', title='Book Ggg', publishYear=2005, pageCount=100}
1 Book{author='Bbb Bbb', title='Book Bbb', publishYear=2018, pageCount=200}
2 Book{author='Ddd Ddd', title='Book Ddd', publishYear=2015, pageCount=400}
3 Book{author='Aaa Aaa', title='Book Aaa', publishYear=2027, pageCount=500}
4 Book{author='Iii Iii', title='Book Iii', publishYear=2012, pageCount=600}
5 Book{author='Hhh Hhh', title='Book Hhh', publishYear=2021, pageCount=800}
6 Book{author='Ccc Ccc', title='Book Ccc', publishYear=2024, pageCount=900}
7 Book{author='Jjj Jjj', title='Book Jjj', publishYear=2001, pageCount=1000}
8 Book{author='Lll Lll', title='Book Lll', publishYear=2025, pageCount=1100}
9 Book{author='Fff Fff', title='Book Fff', publishYear=2030, pageCount=1200}
```

```
1 < mvn exec:java -Dexec.mainClass="com.example.App" > output.txt
2
3 [INFO] Scanning for projects...
4 [INFO] -----< com.example:student-reader >-----
5 [INFO] Building student-reader 1.0-SNAPSHOT
6 [INFO]    from pom.xml
7 [INFO] -----[ jar ]-----
8 [INFO]
9 [INFO] --- exec:3.6.3:java (default-cli) @ student-reader ---
10 Student{
11   name='Aaa'
12   surname='Bbb'
13   books=
14     Book{author='Aaa Aaa', title='Book Aaa', publishYear=2027, pageCount=500}
15     Book{author='Bbb Bbb', title='Book Bbb', publishYear=2018, pageCount=200}
16     Book{author='Ccc Ccc', title='Book Ccc', publishYear=2024, pageCount=900}
17     Book{author='Ddd Ddd', title='Book Ddd', publishYear=2015, pageCount=400}
18     Book{author='Ddd Ddd', title='Book Ddd', publishYear=2015, pageCount=400}
19 }
20 Student{
21   name='Ccc'
22   surname='Ddd'
23   books=
24     Book{author='Eee Eee', title='Book Eee', publishYear=1999, pageCount=300}
25     Book{author='Fff Fff', title='Book Fff', publishYear=2030, pageCount=1200}
26     Book{author='Ggg Ggg', title='Book Ggg', publishYear=2005, pageCount=100}
27     Book{author='Hhh Hhh', title='Book Hhh', publishYear=2021, pageCount=800}
28     Book{author='Iii Iii', title='Book Iii', publishYear=2012, pageCount=600}
29 }
30 Student{
31   name='Eee'
32   surname='Fff'
33   books=
34     Book{author='Aaa Aaa', title='Book Aaa', publishYear=2027, pageCount=500}
35     Book{author='Bbb Bbb', title='Book Bbb', publishYear=2018, pageCount=200}
36     Book{author='Jjj Jjj', title='Book Jjj', publishYear=2001, pageCount=1000}
37     Book{author='Kkk Kkk', title='Book Kkk', publishYear=1995, pageCount=700}
```

```
37 Student{
36   name='Aaa'
35   surname='Bbb'
34   books=
33     Book{author='Aaa Aaa', title='Book Aaa', publishYear=2027, pageCount=500}
32     Book{author='Bbb Bbb', title='Book Bbb', publishYear=2018, pageCount=200}
31     Book{author='Ccc Ccc', title='Book Ccc', publishYear=2024, pageCount=900}
30     Book{author='Ddd Ddd', title='Book Ddd', publishYear=2015, pageCount=400}
29     Book{author='Ddd Ddd', title='Book Ddd', publishYear=2015, pageCount=400}
28 }
27 Student{
26   name='Ccc'
25   surname='Ddd'
24   books=
23     Book{author='Eee Eee', title='Book Eee', publishYear=1999, pageCount=300}
22     Book{author='Fff Fff', title='Book Fff', publishYear=2030, pageCount=1200}
21     Book{author='Ggg Ggg', title='Book Ggg', publishYear=2005, pageCount=100}
20     Book{author='Hhh Hhh', title='Book Hhh', publishYear=2021, pageCount=800}
19     Book{author='Iii Iii', title='Book Iii', publishYear=2012, pageCount=600}
18 }
17 Student{
16   name='Eee'
15   surname='Fff'
14   books=
13     Book{author='Aaa Aaa', title='Book Aaa', publishYear=2027, pageCount=500}
12     Book{author='Bbb Bbb', title='Book Bbb', publishYear=2018, pageCount=200}
11     Book{author='Jjj Jjj', title='Book Jjj', publishYear=2001, pageCount=1000}
10     Book{author='Kkk Kkk', title='Book Kkk', publishYear=1995, pageCount=700}
9     Book{author='Lll Lll', title='Book Lll', publishYear=2025, pageCount=1100}
8 }
7 Год выпуска: 2005
6 Выполнение программы завершено
5 [INFO] -----
4 [INFO] BUILD SUCCESS
3 [INFO] -----
2 [INFO] Total time: 1.025 s
1 [INFO] Finished at: 2026-08-06T23:30:03+04:00
48 [INFO] -----
```