**Report By:Manahil Gul-QA Intern**

**Assignment 1-Java Learning**

Book Class:

package com.example.bookproject;

import java.util.List;

public class Book {

private String title;

private String author;

private Metadata metadata;

// Getters and Setters

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public Metadata getMetadata() {

return metadata;

}

public void setMetadata(Metadata metadata) {

this.metadata = metadata;

}

public void displayBookInfo() {

System.***out***.println("Title of the Book: " + title);

System.***out***.println("Author of the Book: " + author);

System.***out***.println("Price of "+title+" :" + metadata.getPrice());

System.***out***.println("Categories: " + metadata.getCategories());

System.***out***.println("International Standard Book Number: " + metadata.getIsbn());

System.***out***.println("No.of Pages: " + metadata.getPages());

System.***out***.println();

}

}

BookWarehouse Class:

package com.example.bookproject;

import com.google.gson.Gson;

import com.google.gson.reflect.TypeToken;

import java.io.FileReader;

import java.io.IOException;

import java.lang.reflect.Type;

import java.util.List;

import java.util.OptionalDouble;

import java.util.stream.Collectors;

public class BookWarehouse {

private List<Book> books;

public BookWarehouse(String jsonFilePath) {

try (FileReader reader = new FileReader(jsonFilePath)) {

Type bookListType = new TypeToken<List<Book>>() {}.getType();

books = new Gson().fromJson(reader, bookListType);

} catch (IOException e) {

e.printStackTrace();

}

}

// Display details of all books

public void display\_All() {

if (books.isEmpty()) {

System.***out***.println("No books available.");

} else {

books.forEach(Book::displayBookInfo);

}

}

// Find and display books by a specific author

// Returns true if books are found, false otherwise

public boolean search\_Book\_by\_Author(String authorName) {

List<Book> booksByAuthor = books.stream()

.filter(book -> book.getAuthor().equalsIgnoreCase(authorName))

.collect(Collectors.*toList*());

if (booksByAuthor.isEmpty()) {

return false;

} else {

booksByAuthor.forEach(Book::displayBookInfo);

return true;

}

}

// Find and display books by a specific category

// Returns true if books are found, false otherwise

public boolean search\_book\_by\_category(String category) {

List<Book> booksByCategory = books.stream()

.filter(book -> book.getMetadata().getCategories().contains(category))

.collect(Collectors.*toList*());

if (booksByCategory.isEmpty()) {

return false;

} else {

booksByCategory.forEach(Book::displayBookInfo);

return true;

}

}

// Calculate and display the average price of all books

public void show\_average\_price() {

OptionalDouble average = books.stream()

.mapToDouble(book -> book.getMetadata().getPrice())

.average();

System.***out***.println("Average Price: " + (average.isPresent() ? average.getAsDouble() : "No books available"));

}

// Display the title of the book with the highest price

public void show\_expensive\_book() {

books.stream()

.max((book1, book2) -> Double.*compare*(book1.getMetadata().getPrice(), book2.getMetadata().getPrice()))

.ifPresent(book -> System.***out***.println("Highest Priced Book Title: " + book.getTitle()));

}

}

Metadata Class:

package com.example.bookproject;

import java.util.List;

public class Metadata {

private double price;

private List<String> categories;

private String isbn;

private int pages;

// Getters and Setters

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public List<String> getCategories() {

return categories;

}

public void setCategories(List<String> categories) {

this.categories = categories;

}

public String getIsbn() {

return isbn;

}

public void setIsbn(String isbn) {

this.isbn = isbn;

}

public int getPages() {

return pages;

}

public void setPages(int pages) {

this.pages = pages;

}

}

books.Json File:

[

{

"title": "Hamlet",

"author": "William Shakespeare",

"metadata": {

"price": 2500,

"categories": ["drama", "tragedy"],

"isbn": "9781234567890",

"pages": 320

}

},

{

"title": "1984",

"author": "George Orwell",

"metadata": {

"price": 1800,

"categories": ["dystopian", "sci-fi"],

"isbn": "9781234567891",

"pages": 268

}

},

{

"title": "The Hobbit",

"author": "J.R.R. Tolkien",

"metadata": {

"price": 2200,

"categories": ["fantasy", "adventure"],

"isbn": "9781234567892",

"pages": 310

}

},

{

"title": "Macbeth",

"author": "William Shakespeare",

"metadata": {

"price": 2000,

"categories": ["drama", "tragedy"],

"isbn": "9781234567893",

"pages": 250

}

},

{

"title": "To Kill a Mockingbird",

"author": "Harper Lee",

"metadata": {

"price": 1700,

"categories": ["drama"],

"isbn": "9781234567894",

"pages": 281

}

},

{

"title": "Brave New World",

"author": "Aldous Huxley",

"metadata": {

"price": 1900,

"categories": ["dystopian", "sci-fi"],

"isbn": "9781234567895",

"pages": 311

}

},

{

"title": "The Catcher in the Rye",

"author": "J.D. Salinger",

"metadata": {

"price": 1600,

"categories": ["drama", "coming-of-age"],

"isbn": "9781234567896",

"pages": 214

}

}

]

Pom.xml File:

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>BookProject</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<!-- https://mvnrepository.com/artifact/com.google.code.gson/gson -->

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.11.0</version>

</dependency>

</dependencies>

</project>

Output:









