

## **Task 1**

**Write a Java program that performs the following operations on a given string: find its length, convert it to uppercase, extract a substring, and replace a character.**

```
public class StringOperations {  
  
    {  
  
        public static void main(String[] args) {  
  
            String input = "Hello, World!";  
  
  
            // Find the length of the string  
            int length = input.length();  
  
            System.out.println("Length of the string: " + length);  
  
  
            // Convert the string to uppercase  
            String upperCaseString = input.toUpperCase();  
  
            System.out.println("Uppercase string: " + upperCaseString);  
  
  
            // Extract a substring  
            //extracting "World" from "Hello, World!"  
            String substring = input.substring(7, 12);  
  
            System.out.println("Extracted substring: " + substring);  
  
  
            // Replace a character  
            //replacing 'o' with 'a'  
            String replacedString = input.replace('o', 'a');  
  
            System.out.println("String after replacement: " + replacedString);  
        }  
    }  
}
```

## **Output:**

```
Length of the string: 13  
Uppercase string: HELLO, WORLD!  
Extracted substring: World  
String after replacement: Hella, World!  
PS C:\Users\ASUS\OneDrive\Desktop\Tarento>
```

## **Task 2**

**Write a Java program to parse a string into different primitive data types using wrapper class methods like `parseInt`, `parseDouble`, `parseBoolean`, etc., and convert primitive types to strings using `valueOf`**

```
public class ParseStringToPrimitives {  
    public static void main(String[] args) {  
        // Sample input Strings  
        String intString = "123";  
        String doubleString = "45.67";  
        String booleanString = "true";  
        String charString = "A";  
  
        // Parsing strings to primitive data types  
        int parsedInt = Integer.parseInt(intString);  
        double parsedDouble = Double.parseDouble(doubleString);  
        boolean parsedBoolean = Boolean.parseBoolean(booleanString);  
        char parsedChar = charString.charAt(0); // Since char does not have a parse method  
  
        // Printing parsed values  
        System.out.println("Parsed int: " + parsedInt);  
        System.out.println("Parsed double: " + parsedDouble);  
        System.out.println("Parsed boolean: " + parsedBoolean);  
        System.out.println("Parsed char: " + parsedChar);  
  
        // Converting primitive types to strings using valueOf  
        String intToString = String.valueOf(parsedInt);  
        String doubleToString = String.valueOf(parsedDouble);  
        String booleanToString = String.valueOf(parsedBoolean);  
        String charToString = String.valueOf(parsedChar);  
  
        // Printing converted string values
```

```
        System.out.println("String from int: " + intToString);

        System.out.println("String from double: " + doubleToString);

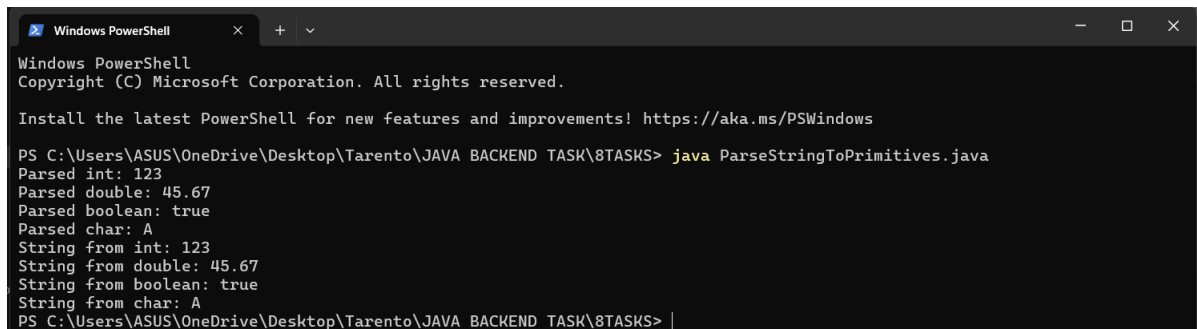
        System.out.println("String from boolean: " + booleanToString);

        System.out.println("String from char: " + charToString);

    }

}
```

### OUTPUT:

A screenshot of a Windows PowerShell terminal window. The window title is "Windows PowerShell". The text inside shows the standard PowerShell startup messages, followed by the command to run a Java program. The output of the program is displayed line by line.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ASUS\OneDrive\Desktop\Tarento\JAVA BACKEND TASK\8TASKS> java ParseStringToPrimitives.java
Parsed int: 123
Parsed double: 45.67
Parsed boolean: true
Parsed char: A
String from int: 123
String from double: 45.67
String from boolean: true
String from char: A
PS C:\Users\ASUS\OneDrive\Desktop\Tarento\JAVA BACKEND TASK\8TASKS> |
```

### **Task 3**

**Write a Java program to sort an array of integers in ascending order using a sorting algorithm of your choice.**

```
public class BubbleSort {  
    public static void main(String[] args) {  
        int[] array = {64, 34, 25, 12, 22, 11, 90};  
  
        System.out.println("Original array:");  
        printArray(array);  
  
        bubbleSort(array);  
  
        System.out.println("Sorted array in ascending order:");  
        printArray(array);  
    }  
  
    // Bubble Sort algorithm  
    public static void bubbleSort(int[] arr) {  
        int n = arr.length;  
        for (int i = 0; i < n - 1; i++) {  
            for (int j = 0; j < n - i - 1; j++) {  
                if (arr[j] > arr[j + 1]) {  
                    // Swap arr[j] and arr[j + 1]  
                    int temp = arr[j];  
                    arr[j] = arr[j + 1];  
                    arr[j + 1] = temp;  
                }  
            }  
        }  
    }  
}
```

```
// method to print an array

public static void printArray(int[] arr) {

    for (int num : arr) {

        System.out.print(num + " ");

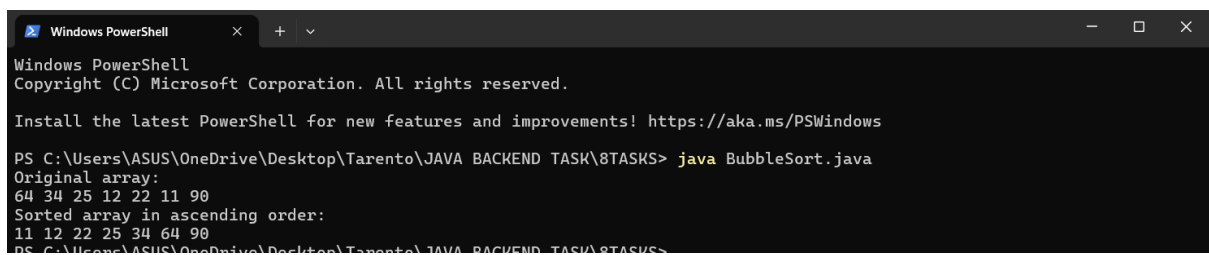
    }

    System.out.println();

}

}
```

### OUTPUT:

A screenshot of a Windows PowerShell terminal window. The window title is "Windows PowerShell". The text inside shows the copyright notice for Microsoft Corporation, a link to update PowerShell, and the execution of a Java program. The command entered is "java BubbleSort.java". The output shows the original array "64 34 25 12 22 11 90" and the sorted array in ascending order "11 12 22 25 34 64 90".

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ASUS\OneDrive\Desktop\Tarento\JAVA BACKEND TASK\8TASKS> java BubbleSort.java
Original array:
64 34 25 12 22 11 90
Sorted array in ascending order:
11 12 22 25 34 64 90
PS C:\Users\ASUS\OneDrive\Desktop\Tarento\JAVA BACKEND TASK\8TASKS>
```

## **Task 4**

**Use an ArrayList to store the list of books. Each book should have attributes such as title, author, ISBN, and price Implement functionalities to add new books, remove existing books, and display all books in the library.**

```
import java.util.ArrayList;
```

```
import java.util.Scanner;
```

```
class Book {
```

```
    private String title;
```

```
    private String author;
```

```
    private String ISBN;
```

```
    private double price;
```

```
    public Book(String title, String author, String ISBN, double price) {
```

```
        this.title = title;
```

```
        this.author = author;
```

```
        this.ISBN = ISBN;
```

```
        this.price = price;
```

```
    }
```

```
    public String getTitle() {
```

```
        return title;
```

```
    }
```

```
    public String getAuthor() {
```

```
        return author;
```

```
    }
```

```
    public String getISBN() {
```

```
        return ISBN;
```

```
    }
```

```
public double getPrice() {  
    return price;  
}
```

```
@Override
```

```
public String toString() {  
    return "Book [Title=" + title + ", Author=" + author + ", ISBN=" + ISBN + ", Price=" + price + "];"  
}  
}
```

```
public class Library {  
    private ArrayList<Book> books;
```

```
public Library() {  
    books = new ArrayList<>();  
}
```

```
// Add a new book to the library
```

```
public void addBook(Book book) {  
    books.add(book);  
    System.out.println("Book added: " + book);  
}
```

```
// Remove an existing book from the library by ISBN
```

```
public void removeBook(String ISBN) {  
    Book bookToRemove = null;  
    for (Book book : books) {  
        if (book.getISBN().equals(ISBN)) {  
            bookToRemove = book;  
            break;
```

```

    }
}
if (bookToRemove != null) {
    books.remove(bookToRemove);
    System.out.println("Book removed: " + bookToRemove);
} else {
    System.out.println("Book with ISBN " + ISBN + " not found.");
}
}

```

*// Display all books in the library*

```

public void displayBooks() {
    if (books.isEmpty()) {
        System.out.println("No books in the library.");
    } else {
        System.out.println("Books in the library:");
        for (Book book : books) {
            System.out.println(book);
        }
    }
}
}

```

```

public static void main(String[] args) {
    Library library = new Library();
    Scanner scanner = new Scanner(System.in);

    while (true) {
        System.out.println("\nLibrary Menu:");
        System.out.println("1. Add a book");
        System.out.println("2. Remove a book");
        System.out.println("3. Display all books");
    }
}

```



```

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline


switch (choice) {

    case 1:

        System.out.print("Enter title: ");

        String title = scanner.nextLine();

        System.out.print("Enter author: ");

        String author = scanner.nextLine();

        System.out.print("Enter ISBN: ");

        String ISBN = scanner.nextLine();

        System.out.print("Enter price: ");

        double price = scanner.nextDouble();

        scanner.nextLine(); // Consume newline

        library.addBook(new Book(title, author, ISBN, price));

        break;

    case 2:

        System.out.print("Enter ISBN of the book to remove: ");

        String isbnToRemove = scanner.nextLine();

        library.removeBook(isbnToRemove);

        break;

    case 3:

        library.displayBooks();

        break;

    case 4:

        System.out.println("Exiting the library system. Goodbye!");

        scanner.close();

        System.exit(0);

    default:

```

```

        System.out.println("Invalid choice. Please try again.");
    }

}

}

}

```

### OUTPUT:

```

Library Menu:
1. Add a book
2. Remove a book
3. Display all books
4. Exit
Enter your choice: 1
Enter title: NEWBOOK
Enter author: NEWAUTHOR
Enter ISBN: 12AB12CD
Enter price: 500
Book added: Book [Title=NEWBOOK, Author=NEWAUTHOR, ISBN=12AB12CD, Price=500.0]

Library Menu:
1. Add a book
2. Remove a book
3. Display all books
4. Exit
Enter your choice: 3
Books in the library:
Book [Title=NEWBOOK, Author=NEWAUTHOR, ISBN=12AB12CD, Price=500.0]

Library Menu:
1. Add a book
2. Remove a book
3. Display all books
4. Exit
Enter your choice: 2
Enter ISBN of the book to remove: 12AB12CD
Book removed: Book [Title=NEWBOOK, Author=NEWAUTHOR, ISBN=12AB12CD, Price=500.0]

Library Menu:
1. Add a book
2. Remove a book
3. Display all books
4. Exit
Enter your choice: 3
No books in the library.

```

## **Task 5**

**Use a HashSet to manage the unique genres available in the library. Ensure that new genres can be added without duplicating existing genres**

```
import java.util.ArrayList;
```

```
import java.util.HashSet;
```

```
import java.util.Scanner;
```

```
class Book {
```

```
    private String title;
```

```
    private String author;
```

```
    private String ISBN;
```

```
    private double price;
```

```
    private String genre;
```

```
    public Book(String title, String author, String ISBN, double price, String genre) {
```

```
        this.title = title;
```

```
        this.author = author;
```

```
        this.ISBN = ISBN;
```

```
        this.price = price;
```

```
        this.genre = genre;
```

```
    }
```

```
    public String getTitle() {
```

```
        return title;
```

```
    }
```

```
    public String getAuthor() {
```

```
        return author;
```

```
    }
```

```
    public String getISBN() {
```

```

        return ISBN;
    }

    public double getPrice() {
        return price;
    }

    public String getGenre() {
        return genre;
    }

    @Override
    public String toString() {
        return "Book [Title=" + title + ", Author=" + author + ", ISBN=" + ISBN + ", Price=" + price + ", Genre=" + genre + "]";
    }
}

public class Library {
    private ArrayList<Book> books;
    private HashSet<String> genres;

    public Library() {
        books = new ArrayList<>();
        genres = new HashSet<>();
    }

    // Add a new book to the library
    public void addBook(Book book) {
        books.add(book);
        genres.add(book.getGenre()); // Add genre to the set
    }
}

```

```

        System.out.println("Book added: " + book);
    }

    // Remove an existing book from the library by ISBN
    public void removeBook(String ISBN) {
        Book bookToRemove = null;
        for (Book book : books) {
            if (book.getISBN().equals(ISBN)) {
                bookToRemove = book;
                break;
            }
        }
        if (bookToRemove != null) {
            books.remove(bookToRemove);
            System.out.println("Book removed: " + bookToRemove);
        } else {
            System.out.println("Book with ISBN " + ISBN + " not found.");
        }
    }

    // Display all books in the library
    public void displayBooks() {
        if (books.isEmpty()) {
            System.out.println("No books in the library.");
        } else {
            System.out.println("Books in the library:");
            for (Book book : books) {
                System.out.println(book);
            }
        }
    }
}

```

```
// Display all unique genres in the library
public void displayGenres() {
    if (genres.isEmpty()) {
        System.out.println("No genres available.");
    } else {
        System.out.println("Genres available in the library:");
        for (String genre : genres) {
            System.out.println(genre);
        }
    }
}
```

```
public static void main(String[] args) {
    Library library = new Library();
    Scanner scanner = new Scanner(System.in);

    while (true) {
        System.out.println("\nLibrary Menu:");
        System.out.println("1. Add a book");
        System.out.println("2. Remove a book");
        System.out.println("3. Display all books");
        System.out.println("4. Display all genres");
        System.out.println("5. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        switch (choice) {
            case 1:
                System.out.print("Enter title: ");
```

```

        String title = scanner.nextLine();

        System.out.print("Enter author: ");

        String author = scanner.nextLine();

        System.out.print("Enter ISBN: ");

        String ISBN = scanner.nextLine();

        System.out.print("Enter price: ");

        double price = scanner.nextDouble();

        scanner.nextLine(); // Consume newline

        System.out.print("Enter genre: ");

        String genre = scanner.nextLine();

        library.addBook(new Book(title, author, ISBN, price, genre));

        break;
    case 2:

        System.out.print("Enter ISBN of the book to remove: ");

        String isbnToRemove = scanner.nextLine();

        library.removeBook(isbnToRemove);

        break;
    case 3:

        library.displayBooks();

        break;
    case 4:

        library.displayGenres();

        break;
    case 5:

        System.out.println("Exiting the library system. Goodbye!");

        scanner.close();

        System.exit(0);
    default:

        System.out.println("Invalid choice. Please try again.");

    }
}

```

```
}  
  
}
```

### OUTPUT:

```
Ctrl+Shift+M) - Total 19 Problems  
1. Add a book  
2. Remove a book  
3. Display all books  
4. Display all genres  
5. Exit  
Enter your choice: 4  
No genres available.  
  
Library Menu:  
1. Add a book  
2. Remove a book  
3. Display all books  
4. Display all genres  
5. Exit  
Enter your choice: 1  
Enter title: sag  
Enter author: sag  
Enter ISBN: sa123  
Enter price: 500  
Enter genre: horror  
Book added: Book [Title=sag, Author=sag, ISBN=sa123, Price=500.0, Genre=horror]  
  
Library Menu:  
1. Add a book  
2. Remove a book  
3. Display all books  
4. Display all genres  
5. Exit  
Enter your choice: 4  
Genres available in the library:  
horror
```



## **Task 6**

**Use a HashMap to map ISBN numbers to books for quick lookup. Implement functionalities to add, update, and retrieve book details using ISBN.**

```
import java.util.HashMap;
```

```
import java.util.HashSet;
```

```
import java.util.Scanner;
```

```
class Book {
```

```
    private String title;
```

```
    private String author;
```

```
    private String ISBN;
```

```
    private double price;
```

```
    private String genre;
```

```
    public Book(String title, String author, String ISBN, double price, String genre) {
```

```
        this.title = title;
```

```
        this.author = author;
```

```
        this.ISBN = ISBN;
```

```
        this.price = price;
```

```
        this.genre = genre;
```

```
    }
```

```
    public String getTitle() {
```

```
        return title;
```

```
    }
```

```
    public String getAuthor() {
```

```
        return author;
```

```
    }
```

```
    public String getISBN() {
```

```

        return ISBN;
    }

    public double getPrice() {
        return price;
    }

    public String getGenre() {
        return genre;
    }

    @Override
    public String toString() {
        return "Book [Title=" + title + ", Author=" + author + ", ISBN=" + ISBN + ", Price=" + price + ", Genre=" + genre + "]";
    }
}

public class Library {
    private HashMap<String, Book> books;
    private HashSet<String> genres;

    public Library() {
        books = new HashMap<>();
        genres = new HashSet<>();
    }

    // Add a new book to the library
    public void addBook(Book book) {
        books.put(book.getISBN(), book);
        genres.add(book.getGenre()); // Add genre to the set
    }
}

```

```
        System.out.println("Book added: " + book);
    }

    // Update an existing book in the library by ISBN
    public void updateBook(String ISBN, Book updatedBook) {
        if (books.containsKey(ISBN)) {
            books.put(ISBN, updatedBook);
            genres.add(updatedBook.getGenre()); // Add or update genre
            System.out.println("Book updated: " + updatedBook);
        } else {
            System.out.println("Book with ISBN " + ISBN + " not found.");
        }
    }
}
```

```
    // Remove an existing book from the library by ISBN
    public void removeBook(String ISBN) {
        Book bookToRemove = books.remove(ISBN);
        if (bookToRemove != null) {
            System.out.println("Book removed: " + bookToRemove);
        } else {
            System.out.println("Book with ISBN " + ISBN + " not found.");
        }
    }
}
```

```
    // Retrieve and display a book by ISBN
    public void getBook(String ISBN) {
        Book book = books.get(ISBN);
        if (book != null) {
            System.out.println("Book found: " + book);
        } else {
            System.out.println("Book with ISBN " + ISBN + " not found.");
        }
    }
}
```

```
}  
}
```

```
// Display all books in the library
```

```
public void displayBooks() {  
    if (books.isEmpty()) {  
        System.out.println("No books in the library.");  
    } else {  
        System.out.println("Books in the library:");  
        for (Book book : books.values()) {  
            System.out.println(book);  
        }  
    }  
}
```

```
// Display all unique genres in the library
```

```
public void displayGenres() {  
    if (genres.isEmpty()) {  
        System.out.println("No genres available.");  
    } else {  
        System.out.println("Genres available in the library:");  
        for (String genre : genres) {  
            System.out.println(genre);  
        }  
    }  
}
```

```
public static void main(String[] args) {  
    Library library = new Library();  
    Scanner scanner = new Scanner(System.in);
```

```

while (true) {

    System.out.println("\nLibrary Menu:");

    System.out.println("1. Add a book");
    System.out.println("2. Update a book");
    System.out.println("3. Remove a book");
    System.out.println("4. Retrieve a book by ISBN");
    System.out.println("5. Display all books");
    System.out.println("6. Display all genres");
    System.out.println("7. Exit");

    System.out.print("Enter your choice: ");

    int choice = scanner.nextInt();

    scanner.nextLine(); // Consume newline

    switch (choice) {

        case 1:

            System.out.print("Enter title: ");

            String title = scanner.nextLine();

            System.out.print("Enter author: ");

            String author = scanner.nextLine();

            System.out.print("Enter ISBN: ");

            String ISBN = scanner.nextLine();

            System.out.print("Enter price: ");

            double price = scanner.nextDouble();

            scanner.nextLine(); // Consume newline

            System.out.print("Enter genre: ");

            String genre = scanner.nextLine();

            library.addBook(new Book(title, author, ISBN, price, genre));

            break;

        case 2:

            System.out.print("Enter ISBN of the book to update: ");

            String isbnToUpdate = scanner.nextLine();

```

```
System.out.print("Enter new title: ");
String newTitle = scanner.nextLine();
System.out.print("Enter new author: ");
String newAuthor = scanner.nextLine();
System.out.print("Enter new price: ");
double newPrice = scanner.nextDouble();
scanner.nextLine(); // Consume newline
System.out.print("Enter new genre: ");
String newGenre = scanner.nextLine();

library.updateBook(isbnToUpdate, new Book(newTitle, newAuthor, isbnToUpdate,
newPrice, newGenre));

break;
case 3:
    System.out.print("Enter ISBN of the book to remove: ");
    String isbnToRemove = scanner.nextLine();
    library.removeBook(isbnToRemove);
    break;
case 4:
    System.out.print("Enter ISBN of the book to retrieve: ");
    String isbnToRetrieve = scanner.nextLine();
    library.getBook(isbnToRetrieve);
    break;
case 5:
    library.displayBooks();
    break;
case 6:
    library.displayGenres();
    break;
case 7:
    System.out.println("Exiting the library system. Goodbye!");
    scanner.close();
```

```

        System.exit(0);

    default:

        System.out.println("Invalid choice. Please try again.");

    }

}

}

}

```

### OUTPUT:

```

Library Menu:
1. Add a book
2. Update a book
3. Remove a book
4. Retrieve a book by ISBN
5. Display all books
6. Display all genres
7. Exit
Enter your choice: 1
Enter title: new
Enter author: new
Enter ISBN: new123
Enter price: 500
Enter genre: horror
Book added: Book [Title=new, Author=new, ISBN=new123, Price=500.0, Genre=horror]

Library Menu:
1. Add a book
2. Update a book
3. Remove a book
4. Retrieve a book by ISBN
5. Display all books
6. Display all genres
7. Exit
Enter your choice: 4
Enter ISBN of the book to retrieve: new123
Book found: Book [Title=new, Author=new, ISBN=new123, Price=500.0, Genre=horror]

Library Menu:
1. Add a book
2. Update a book
3. Remove a book
4. Retrieve a book by ISBN
5. Display all books
6. Display all genres
7. Exit
Enter your choice: █

```

## **Task 7**

**Implement a custom exception called `ProductNotFoundException` that is thrown when a product is not found in the inventory. Use `try`, `catch`, `finally`, `throw`, and `throws` to handle exceptions appropriately.**

```
public class ProductNotFoundException extends Exception {  
    public ProductNotFoundException(String message) {  
        super(message);  
    }  
}
```

```
import java.util.HashMap;  
import java.util.HashSet;  
import java.util.Scanner;
```

```
class Book {  
    private String title;  
    private String author;  
    private String ISBN;  
    private double price;  
    private String genre;  
  
    public Book(String title, String author, String ISBN, double price, String genre) {  
        this.title = title;  
        this.author = author;  
        this.ISBN = ISBN;  
        this.price = price;  
        this.genre = genre;  
    }  
  
    public String getTitle() {  
        return title;  
    }  
}
```



```
}
```

```
public String getAuthor() {  
    return author;  
}
```

```
public String getISBN() {  
    return ISBN;  
}
```

```
public double getPrice() {  
    return price;  
}
```

```
public String getGenre() {  
    return genre;  
}
```

```
@Override
```

```
public String toString() {  
    return "Book [Title=" + title + ", Author=" + author + ", ISBN=" + ISBN + ", Price=" + price + ",  
Genre=" + genre + "];"  
}  
}
```

```
public class Library {  
    private HashMap<String, Book> books;  
    private HashSet<String> genres;  
  
    public Library() {  
        books = new HashMap<>();  
    }  
}
```

```

    genres = new HashSet<>();
}

// Add a new book to the library
public void addBook(Book book) {
    books.put(book.getISBN(), book);
    genres.add(book.getGenre()); // Add genre to the set
    System.out.println("Book added: " + book);
}

// Update an existing book in the library by ISBN
public void updateBook(String ISBN, Book updatedBook) throws ProductNotFoundException {
    if (books.containsKey(ISBN)) {
        books.put(ISBN, updatedBook);
        genres.add(updatedBook.getGenre()); // Add or update genre
        System.out.println("Book updated: " + updatedBook);
    } else {
        throw new ProductNotFoundException("Book with ISBN " + ISBN + " not found.");
    }
}

// Remove an existing book from the library by ISBN
public void removeBook(String ISBN) throws ProductNotFoundException {
    Book bookToRemove = books.remove(ISBN);
    if (bookToRemove != null) {
        System.out.println("Book removed: " + bookToRemove);
    } else {
        throw new ProductNotFoundException("Book with ISBN " + ISBN + " not found.");
    }
}

```

*// Retrieve and display a book by ISBN*

```
public void getBook(String ISBN) throws ProductNotFoundException {  
    Book book = books.get(ISBN);  
    if (book != null) {  
        System.out.println("Book found: " + book);  
    } else {  
        throw new ProductNotFoundException("Book with ISBN " + ISBN + " not found.");  
    }  
}
```

*// Display all books in the library*

```
public void displayBooks() {  
    if (books.isEmpty()) {  
        System.out.println("No books in the library.");  
    } else {  
        System.out.println("Books in the library:");  
        for (Book book : books.values()) {  
            System.out.println(book);  
        }  
    }  
}
```

*// Display all unique genres in the library*

```
public void displayGenres() {  
    if (genres.isEmpty()) {  
        System.out.println("No genres available.");  
    } else {  
        System.out.println("Genres available in the library:");  
        for (String genre : genres) {  
            System.out.println(genre);  
        }  
    }  
}
```



```
scanner.nextLine(); // Consume newline
System.out.print("Enter genre: ");
String genre = scanner.nextLine();
library.addBook(new Book(title, author, ISBN, price, genre));
break;
```

case 2:

```
System.out.print("Enter ISBN of the book to update: ");
String isbnToUpdate = scanner.nextLine();
System.out.print("Enter new title: ");
String newTitle = scanner.nextLine();
System.out.print("Enter new author: ");
String newAuthor = scanner.nextLine();
System.out.print("Enter new price: ");
double newPrice = scanner.nextDouble();
scanner.nextLine(); // Consume newline
System.out.print("Enter new genre: ");
String newGenre = scanner.nextLine();

library.updateBook(isbnToUpdate, new Book(newTitle, newAuthor, isbnToUpdate,
newPrice, newGenre));

break;
```

case 3:

```
System.out.print("Enter ISBN of the book to remove: ");
String isbnToRemove = scanner.nextLine();
library.removeBook(isbnToRemove);
break;
```

case 4:

```
System.out.print("Enter ISBN of the book to retrieve: ");
String isbnToRetrieve = scanner.nextLine();
library.getBook(isbnToRetrieve);
break;
```

case 5:

```

        library.displayBooks();

        break;

    case 6:

        library.displayGenres();

        break;

    case 7:

        System.out.println("Exiting the library system. Goodbye!");

        scanner.close();

        System.exit(0);

    default:

        System.out.println("Invalid choice. Please try again.");

    }

} catch (ProductNotFoundException e) {

    System.out.println(e.getMessage());

} finally {

    // Code that should always run, e.g., closing resources, can go here

}

}

}

}

```

## OUTPUT:

```

Library Menu:
1. Add a book
2. Update a book
3. Remove a book
4. Retrieve a book by ISBN
5. Display all books
6. Display all genres
7. Exit
Enter your choice: 1
Enter title: new
Enter author: new
Enter ISBN: new123
Enter price: 500
Enter genre: horror
Book added: Book [Title=new, Author=new, ISBN=new123, Price=500.0, Genre=horror]

Library Menu:
1. Add a book
2. Update a book
3. Remove a book
4. Retrieve a book by ISBN
5. Display all books
6. Display all genres
7. Exit
Enter your choice: 2
Enter ISBN of the book to update: new12
Enter new title: new
Enter new author: new
Enter new price: 500
Enter new genre: horror
Book with ISBN new12 not found.

Library Menu:
1. Add a book

```

## **Task 8**

**Use any one of the file handling to read employee records from a text file and write employee records to a text file**

```
import java.io.*;

import java.util.ArrayList;
import java.util.List;

// Employee class
class Employee {
    private String id;
    private String name;
    private String position;

    public Employee(String id, String name, String position) {
        this.id = id;
        this.name = name;
        this.position = position;
    }

    public String getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getPosition() {
        return position;
    }
}
```

```

@Override

public String toString() {

    return id + "," + name + "," + position;

}

}

// FileHandler class

class FileHandler {

    // Read employee records from a file

    public static List<Employee> readEmployeesFromFile(String filePath) {

        List<Employee> employees = new ArrayList<>();

        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {

            String line;

            while ((line = reader.readLine()) != null) {

                String[] parts = line.split(",");

                if (parts.length == 3) {

                    Employee employee = new Employee(parts[0], parts[1], parts[2]);

                    employees.add(employee);

                }

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

        return employees;

    }

    // Write employee records to a file

    public static void writeEmployeesToFile(String filePath, List<Employee> employees) {

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {

            for (Employee employee : employees) {

```



```

        writer.write(employee.toString());

        writer.newLine();
    }
} catch (IOException e) {
    e.printStackTrace();
}
}
}

// Main class

public class Main {

    public static void main(String[] args) {

        String inputFilePath = "C:\\Users\\ASUS\\OneDrive\\Desktop\\Tarento\\JAVA BACKEND TASK\\8TASKS\\employees.txt";

        String outputFilePath = "C:\\Users\\ASUS\\OneDrive\\Desktop\\Tarento\\JAVA BACKEND TASK\\8TASKS\\output_employees.txt";

        // Read employee records from file

        List<Employee> employees = FileHandler.readEmployeesFromFile(inputFilePath);

        // Optionally, print out the employees to verify

        System.out.println("Employees read from file:");

        for (Employee employee : employees) {

            System.out.println(employee);

        }

        // Write employee records to a new file

        FileHandler.writeEmployeesToFile(outputFilePath, employees);

        System.out.println("\nEmployee records have been written to " + outputFilePath);

    }

}

```

## OUTPUT:

```
Employees read from file:
E001,John Doe,Software Engineer
E002,Jane Smith,Project Manager
E003,Bob Johnson,Quality Assurance
E004,Alice Williams,Data Scientist
E005,Charlie Brown,UI/UX Designer

Employee records have been written to C:\Users\ASUS\OneDrive\Desktop\Tarento\JAVA BACKEND TASK\8TASKS\output_employees.txt
PS C:\Users\ASUS\OneDrive\Desktop\Tarento>
```

## employees.txt

```
File Edit View

E001,John Doe,Software Engineer
E002,Jane Smith,Project Manager
E003,Bob Johnson,Quality Assurance
E004,Alice Williams,Data Scientist
E005,Charlie Brown,UI/UX Designer
```

## Output\_employees.txt

```
File Edit View

E001,John Doe,Software Engineer
E002,Jane Smith,Project Manager
E003,Bob Johnson,Quality Assurance
E004,Alice Williams,Data Scientist
E005,Charlie Brown,UI/UX Designer
```