**Student Marks Filter and Sort**

**Problem Statement**

You are given a list of student marks in an ArrayList<Integer>.

Your task is to:

Read in student marks from the user.

Remove all marks less than 40 (fail marks).

Sort the remaining marks in ascending order.

Display:

The total number of passing students

The highest and lowest mark

The full sorted list

Input Format

First line:

Integer n (number of marks)

Next n lines:

One integer each (student mark)

Sample Input:

 6

35

85

37

42

19

90

Expected Output

Filtered and Sorted Marks:

42

55

85

90

Total Passing Students: 4

Highest Mark: 90

Lowest Mark: 42

Constraints Class name: Main

Method name: processMarks (ArrayList<Integer> marks)

Filter Fail Marks, Sort the Remaining Marks, Display the Cleaned List, determine the highest and lowest marks among passers.

Use these inbuilt methods:

removeIf (Predicate)

Collections.sort()

Collections.max() and Collections.min()

Input variable: marks (ArrayList of Integer)

 Inputs are always valid integers ≥ 0.

**Participant Name Manager**

**Problem Statement**

You are managing the registration of participants for an event. To do this efficiently, you need to

Store all participant names in an ArrayList<String>

Remove duplicates (case-Insensitive)

**Display:**

Total number of unique participants

Names that start with a particular character

Final list of names sorted alphabetically

Update a participant's name based on the user's input (old name. new name)

static void processNames(ArrayList<String> names, char filterChar)

This method performs the following operations:

Removes duplicates from the given list of participant names in a case-insensitive manner.

Counts and displays unique participant names.

Filters and displays names that begin with a given character (case-Insensitive).

Sorts and displays the final list of names alphabetically.

static void updateName(ArrayList<String> names, String oldName, String newName)

This method allows the user to replace an existing participant's name with a new one in the list. The match is case-insensitive, if the old name is found, it is replaced

otherwise, an appropriate message is displayed.

**Sample Input**

6

Alice

Bob

alice

Diana

bob

Charlie

A

Bab

**Expected Output**

Unique Participant Count: 4

Names starting with 'A:

Alice

Sorted Participant List:

Alice

Bob

Charlie

Diana

Updated name: Bob -> Bobby

Final Participant List:

Alice

Bobby

Charlie

Diana

Constraints:

Class Name: Main

Method 1: public static void processNames(ArrayList<String> names, char filterChar)

Method 2: public static void updateName(ArrayList<String> names, String oldName, String newName)

Use equalsignoreCase(), startsWith(), and Collections.sort()

Read inputs via Scanner and display via System.out

Case-insensitive comparison must be maintained

Student Marks Manager with Add. Update, Remove

Problem Statement:

You are building a modular Student Marks Manager to dynamically manage student records using proper class separation and method structure

Requirements:

1. Student class (Model)

Define a class named Student with:

Fields

int rollNumber

String name

double marks

A parameterized constructor, getters setter for all the fields.

A toString() method that returns:

Roll: 101, Name: Arjun, Marks: 87.5

2. StudentManager class

Define a class StudentManager that uses ArrayList<Student> to: Store student objects in the variable named students.

Provide the following modular methods:

public void addStudents (Scanner sc)

public void update StudentMarks (Scanner sc)

public void removeStudent (Scanner sc)

public void displayStudents()

Each method should handle its own logic and work with the student list named students.

3. Main class:

Create a class Main that contains only the main() method and acts as the application driver:

public static void main(String[] args)

In the main method:

103 Ravi 75.0

Enter roll number to update marks: 102

Enter new marks: 95.0

Enter roll number to remove: 101

Expected Output:

Student List:

Roll: 102, Name: Meera, Marks: 95.0

Roll: 103, Name: Ravi, Marks: 75.0

Constraints:

Use ArrayList<Student> to store and manage students.

Use the exact class names:

Student

StudentManager

Main

If an invalid roll number is entered for update or removal, print:

Student not found for update

Student not found for removal

Use Scanner for all input operations.

Enhanced Task Manager Using LinkedList

Problem Statement:

Build a simple task manager application that allows users to:

Add tasks.

Mark a task as completed by name.

Remove completed tasks.

Display all pending and completed tasks separately.

Use two LinkedList<String>:

One for pending tasks.

One for completed tasks.

Introduce a method mark TaskAsCompleted() that:

Checks if the task exists in the pending list.

If found, moves it to the completed list.

If not found, display: "Task not found in pending list.

Sample Input:

Enter number of tasks to add: 2

Enter task: Read book

Enter task: Attend meeting

Enter task to mark as completed: Attend meeting

Enter completed task to remove: Attend meeting

I

Expected Output:

Pending Tasks

1. Read book

Completed Tasks:

None

Method Signatures:

public static void addTasks (LinkedList<String> pendinglist, Scanner sc, int n)

public static void mark TaskAsCompleted(LinkedList<String> pendingList, LinkedList<String> completedlist, String taskName)

public static void removeCompleted Task(LinkedList<String> completedList, String taskName)

public static void display Tasks (LinkedList<String> pendingList, LinkedList<String> completedList)

Constraints:

Use two LinkedList<String>:

pendingList for active tasks

completedList for finished tasks

Class name must be TaskManagerApp

Use Scanner for all inputs.

Add proper checks before marking/removing tasks.

Display "No tasks available if both lists are empty.

Display "Task not found in pending list if task not found when marking as completed.

Display "Task not found in completed list if task not found when removing from completed list.

Unique PIN Collector

Problem Statement

A company is collecting unique 4-digit PINs from users for access authentication.

You are to write a program that

Accepts n 4-digit PINs from the user.

Stores only unique PINs using HashSet<Integer>.

Skips duplicate entries and prints:

"Duplicate PIN ignored: 1234"

If the PIN is not a 4-digit number, throw a custom exception Invalid PinException, and print

"Invalid PIN entered: 99 (PIN must be 4 digits)"

At the end, display all the unique PINs collected.

Sample Input:

6

1234

4321

1234

99

6789

9876

Expected Output:

plicate PIN ignored: 1234

9876

Expected Output:

Duplicate PIN ignored: 1234

Invalid PIN entered: 99 (PIN must be 4 digits)

Unique PINs Collected:

1234

4321

6789

9876

Constraints

Class Name: Main

Method Name: collectPins

Method Signature: public static void collectPins(int n, Scanner sc)

Use HashSet<Integer> for storing unique PINs.

PIN must be between 1000 and 9999 (inclusive).

If a PIN is invalid, throw a custom exception:

Invalid PinException extends Exception

Catch the custom exception in the main logic and display the required message.

Handle duplicates using HashSet.add(), and print a custom message if duplicate is detected.

Use enhanced for loop to print all stored PINs in the end.

Unique Book Catalog

Problem Statement:

Create a program that maintains a catalog of unique books using a HashSet. Each Book has the following attributes

int bookid

String title

String author

You must:

Allow the user to add a list of books.

Prevent duplicate books from being added. Two books are considered duplicates if their bookid is the same

Display all the unique books at the end.

Ensure equals() and hashCode() methods are overridden in the Book class so that duplicates are properly handled by the HashSet

Sample Input

Enter number of books: 4

Enter Book ID: 101

Enter Title: Java Basics

Enter Author: Arun

Enter Book ID: 102

Enter Title: Spring Boot

Enter Author: Priya

Expected Output:

Book added successfully.

Book added successfully.

Duplicate book found with ID: 101

Book added successfully.

Unique Books in Catalog:

ID: 101 | Title: Java Basics | Author: Arun

ID: 102 | Title: Spring Boot | Author: Priya

ID: 103 | Title: Angular | Author: Ravi

Constraints & Signatures:

Class: Book

Attributes: int bookid, String title, String author

Override equals() and hashCode() based on bookid

Method:

public void display()

Class: BookCatalog

Method:

public static void addBook(HashSet<Book> catalog, Book book)

Use HashSet<Book> to store books.

No two books should have the same bookid.

Validate bookid must be positive (>0), else print: "Invalid Book ID' and skip.