1. **Introduction to Collections Framework**

**Write a program to demonstrate adding and printing elements from an ArrayList.**

package CollectionCoding;

import java.util.ArrayList;

public class ArrayListDemo {

public static void main(String[] args) {

ArrayList<String> fruits = new ArrayList<>();

fruits.add("Apple");

fruits.add("Banana");

fruits.add("Mango");

System.out.println("Fruits in the list:");

for (String fruit : fruits) {

System.out.println(fruit);

}

}

}

**Show how to use Collections.max() and Collections.min() on a list of integers.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.Collections;

public class MaxMinDemo {

public static void main(String[] args) {

ArrayList<Integer> numbers = new ArrayList<>();

numbers.add(12);

numbers.add(89);

numbers.add(3);

numbers.add(45);

int max = Collections.max(numbers);

int min = Collections.min(numbers);

System.out.println("Numbers: " + numbers);

System.out.println("Maximum: " + max);

System.out.println("Minimum: " + min);

}

}

**Demonstrate the use of Collections.sort() on a list of strings.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.Collections;

public class SortStringsDemo {

public static void main(String[] args) {

ArrayList<String> names = new ArrayList<>();

names.add("Zara");

names.add("Alice");

names.add("Bob");

System.out.println("Before Sorting: " + names);

Collections.sort(names);

System.out.println("After Sorting: " + names);

}

}

**You need to store a dynamic list of student names and display them in alphabetical order. Implement this using a suitable collection.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Scanner;

public class StudentNameSorter {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

ArrayList<String> studentNames = new ArrayList<>();

System.out.print("Enter number of students: ");

int count = sc.nextInt();

sc.nextLine();

for (int i = 0; i < count; i++) {

System.out.print("Enter student name: ");

String name = sc.nextLine();

studentNames.add(name);

}

Collections.sort(studentNames);

System.out.println("Student names in alphabetical order:");

for (String name : studentNames) {

System.out.println(name);

}

sc.close();

}

}

**A user can input any number of integers. Your program should store them and display the sum of all elements using the Collection Framework.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.Scanner;

public class IntegerSumCalculator {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

ArrayList<Integer> numbers = new ArrayList<>();

System.out.println("Enter integers (type 'q' to quit):");

while (true) {

String input = sc.nextLine();

if (input.equalsIgnoreCase("q")) {

break;

}

try {

int number = Integer.parseInt(input);

numbers.add(number);

} catch (NumberFormatException e) {

System.out.println("Invalid input. Please enter an integer or 'q' to quit.");

}

}

int sum = 0;

for (int num : numbers) {

sum += num;

}

System.out.println("Entered numbers: " + numbers);

System.out.println("Sum of all numbers: " + sum);

sc.close();

}

}

1. **List Interface**

**Write a Java program to add, remove, and access elements in an ArrayList.**

package CollectionCoding;

import java.util.ArrayList;

public class ArrayListOperations {

public static void main(String[] args) {

ArrayList<String> items = new ArrayList<>();

items.add("Pen");

items.add("Notebook");

items.add("Eraser");

System.out.println("First item: " + items.get(0));

items.remove("Notebook");

System.out.println("Updated list:");

for (String item : items) {

System.out.println(item);

}

}

}

**Implement a LinkedList that stores and prints employee names.**

package CollectionCoding;

import java.util.LinkedList;

public class LinkedListDemo {

public static void main(String[] args) {

LinkedList<String> employees = new LinkedList<>();

employees.add("Alice");

employees.add("Bob");

employees.add("Charlie");

System.out.println("Employee List:");

for (String name : employees) {

System.out.println(name);

}

}

}

**Demonstrate inserting an element at a specific position in a List.**

package CollectionCoding;

import java.util.ArrayList;

public class InsertAtPosition {

public static void main(String[] args) {

ArrayList<String> colors = new ArrayList<>();

colors.add("Red");

colors.add("Green");

colors.add("Blue");

colors.add(1, "Yellow");

System.out.println("Colors list after insertion:");

for (String color : colors) {

System.out.println(color);

}

}

}

**You're building a to-do list manager. Use ArrayList to add tasks, remove completed ones, and display pending tasks.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.Scanner;

public class ToDoListManager {

public static void main(String[] args) {

ArrayList<String> tasks = new ArrayList<>();

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\nTo-Do List Menu:");

System.out.println("1. Add Task");

System.out.println("2. Remove Completed Task");

System.out.println("3. Show Pending Tasks");

System.out.println("0. Exit");

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

choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

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

String task = sc.nextLine();

tasks.add(task);

break;

case 2:

System.out.print("Enter task to remove: ");

String removeTask = sc.nextLine();

tasks.remove(removeTask);

break;

case 3:

System.out.println("Pending Tasks:");

for (String t : tasks) {

System.out.println("- " + t);

}

break;

case 0:

System.out.println("Exiting To-Do List Manager.");

break;

default:

System.out.println("Invalid choice.");

}

} while (choice != 0);

sc.close();

}

}

**Create a simple shopping cart system where users can add/remove products using a List.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.Scanner;

public class ShoppingCart {

public static void main(String[] args) {

ArrayList<String> cart = new ArrayList<>();

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\nShopping Cart Menu:");

System.out.println("1. Add Product");

System.out.println("2. Remove Product");

System.out.println("3. View Cart");

System.out.println("0. Checkout");

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

choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

System.out.print("Enter product name to add: ");

String product = sc.nextLine();

cart.add(product);

break;

case 2:

System.out.print("Enter product name to remove: ");

String removeProduct = sc.nextLine();

cart.remove(removeProduct);

break;

case 3:

System.out.println("Your Cart:");

for (String item : cart) {

System.out.println("- " + item);

}

break;

case 0:

System.out.println("Checkout complete. Final Cart:");

for (String item : cart) {

System.out.println("- " + item);

}

break;

default:

System.out.println("Invalid choice.");

}

} while (choice != 0);

sc.close();

}

}

1. **Set Interface**

**Write a program using HashSet to store unique student roll numbers.**

package CollectionCoding;

import java.util.HashSet;

public class UniqueRollNumbers {

public static void main(String[] args) {

HashSet<Integer> rollNumbers = new HashSet<>();

rollNumbers.add(101);

rollNumbers.add(102);

rollNumbers.add(103);

rollNumbers.add(101);

System.out.println("Unique Roll Numbers:");

for (int roll : rollNumbers) {

System.out.println(roll);

}

}

}

**Demonstrate how to use TreeSet to automatically sort elements.**

package CollectionCoding;

import java.util.TreeSet;

public class SortedNamesWithTreeSet {

public static void main(String[] args) {

TreeSet<String> names = new TreeSet<>();

names.add("Zara");

names.add("Alice");

names.add("John");

names.add("Bob");

System.out.println("Sorted Names:");

for (String name : names) {

System.out.println(name);

}

}

}

**Use LinkedHashSet to maintain insertion order and prevent duplicates.**

package CollectionCoding;

import java.util.LinkedHashSet;

public class LinkedHashSetExample {

public static void main(String[] args) {

LinkedHashSet<String> animals = new LinkedHashSet<>();

animals.add("Dog");

animals.add("Cat");

animals.add("Elephant");

animals.add("Dog");

System.out.println("Animals (Insertion Order):");

for (String animal : animals) {

System.out.println(animal);

}

}

}

**Design a program to store registered email IDs of users such that no duplicates are allowed.**

package CollectionCoding;

import java.util.HashSet;

import java.util.Scanner;

public class UniqueEmailRegistry {

public static void main(String[] args) {

HashSet<String> emailSet = new HashSet<>();

Scanner sc = new Scanner(System.in);

String input;

System.out.println("Enter email IDs (type 'exit' to stop):");

while (true) {

input = sc.nextLine();

if (input.equalsIgnoreCase("exit")) break;

if (emailSet.add(input)) {

System.out.println("Email registered.");

} else {

System.out.println("Duplicate email! Already registered.");

}

}

System.out.println("\nRegistered Emails:");

for (String email : emailSet) {

System.out.println(email);

}

sc.close();

}

}

**Create a program where a Set is used to eliminate duplicate entries from a list of city names entered by users.**

package CollectionCoding;

import java.util.HashSet;

import java.util.Scanner;

public class UniqueCityNames {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

HashSet<String> cities = new HashSet<>();

String city;

System.out.println("Enter city names (type 'done' to finish):");

while (true) {

city = sc.nextLine();

if (city.equalsIgnoreCase("done")) break;

cities.add(city);

}

System.out.println("\nUnique Cities Entered:");

for (String name : cities) {

System.out.println(name);

}

sc.close();

}

}

1. **Map Interface**

**Write a program using HashMap to store student names and their marks.**

package CollectionCoding;

import java.util.HashMap;

public class StudentMarks {

public static void main(String[] args) {

HashMap<String, Integer> studentMarks = new HashMap<>();

studentMarks.put("Alice", 85);

studentMarks.put("Bob", 90);

studentMarks.put("Charlie", 78);

System.out.println("Student Marks:");

for (String name : studentMarks.keySet()) {

System.out.println(name + ": " + studentMarks.get(name));

}

}

}

**Demonstrate how to iterate over a Map using entrySet().**

package CollectionCoding;

import java.util.HashMap;

import java.util.Map;

public class MapIterationDemo {

public static void main(String[] args) {

HashMap<String, String> countries = new HashMap<>();

countries.put("IN", "India");

countries.put("US", "United States");

countries.put("FR", "France");

System.out.println("Country Codes:");

for (Map.Entry<String, String> entry : countries.entrySet()) {

System.out.println(entry.getKey() + " => " + entry.getValue());

}

}

}

**Show how to update the value associated with a key in a Map.**

package CollectionCoding;

import java.util.HashMap;

public class MapUpdateValue {

public static void main(String[] args) {

HashMap<String, Integer> scores = new HashMap<>();

scores.put("Alice", 70);

scores.put("Bob", 80);

System.out.println("Original Score of Alice: " + scores.get("Alice"));

scores.put("Alice", 95);

System.out.println("Updated Score of Alice: " + scores.get("Alice"));

}

}

**Build a phone directory where names are keys and phone numbers are values.**

package CollectionCoding;

import java.util.HashMap;

import java.util.Scanner;

public class PhoneDirectory {

public static void main(String[] args) {

HashMap<String, String> phoneDirectory = new HashMap<>();

Scanner sc = new Scanner(System.in);

String name, number;

System.out.println("Enter phone directory entries (type 'exit' to stop):");

while (true) {

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

name = sc.nextLine();

if (name.equalsIgnoreCase("exit")) break;

System.out.print("Enter phone number: ");

number = sc.nextLine();

phoneDirectory.put(name, number);

}

System.out.println("\nPhone Directory:");

for (Map.Entry<String, String> entry : phoneDirectory.entrySet()) {

System.out.println(entry.getKey() + ": " + entry.getValue());

}

sc.close();

}

}

**Create a frequency counter for words in a sentence using a Map.**

package CollectionCoding;

import java.util.HashMap;

import java.util.Scanner;

public class WordFrequencyCounter {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a sentence:");

String sentence = sc.nextLine();

String[] words = sentence.toLowerCase().split("\\s+");

HashMap<String, Integer> frequencyMap = new HashMap<>();

for (String word : words) {

frequencyMap.put(word, frequencyMap.getOrDefault(word, 0) + 1);

}

System.out.println("Word Frequencies:");

for (Map.Entry<String, Integer> entry : frequencyMap.entrySet()) {

System.out.println(entry.getKey() + ": " + entry.getValue());

}

sc.close();

}

}

1. **Queue Interface**

**Implement a simple task queue using LinkedList as a Queue.**

package CollectionCoding;

import java.util.LinkedList;

import java.util.Queue;

public class TaskQueue {

public static void main(String[] args) {

Queue<String> tasks = new LinkedList<>();

tasks.add("Task 1 - Email");

tasks.add("Task 2 - Report");

tasks.add("Task 3 - Meeting");

System.out.println("Task Queue:");

for (String task : tasks) {

System.out.println(task);

}

}

}

**Demonstrate how to add and remove elements using offer() and poll().**

package CollectionCoding;

import java.util.LinkedList;

import java.util.Queue;

public class OfferPollExample {

public static void main(String[] args) {

Queue<String> queue = new LinkedList<>();

queue.offer("Alice");

queue.offer("Bob");

queue.offer("Charlie");

System.out.println("Polling from queue:");

while (!queue.isEmpty()) {

System.out.println("Serving: " + queue.poll());

}

}

}

**Use a PriorityQueue to order tasks by priority (integers).**

package CollectionCoding;

import java.util.PriorityQueue;

public class PriorityTaskQueue {

public static void main(String[] args) {

PriorityQueue<Integer> taskQueue = new PriorityQueue<>();

taskQueue.add(5);

taskQueue.add(2);

taskQueue.add(9);

taskQueue.add(1)

System.out.println("Processing tasks by priority:");

while (!taskQueue.isEmpty()) {

System.out.println("Processing task with priority: " + taskQueue.poll());

}

}

}

**Simulate a print queue system where print jobs are processed in order.**

package CollectionCoding;

import java.util.LinkedList;

import java.util.Queue;

import java.util.Scanner;

public class PrintQueueSimulator {

public static void main(String[] args) {

Queue<String> printQueue = new LinkedList<>();

Scanner sc = new Scanner(System.in);

String input;

System.out.println("Enter print jobs (type 'print' to process, 'exit' to stop):");

while (true) {

input = sc.nextLine();

if (input.equalsIgnoreCase("exit")) {

break;

} else if (input.equalsIgnoreCase("print")) {

if (!printQueue.isEmpty()) {

System.out.println("Printing: " + printQueue.poll());

} else {

System.out.println("No print jobs in the queue.");

}

} else {

printQueue.offer(input);

System.out.println("Job added to queue.");

}

}

sc.close();

}

}

**Create a ticket booking system where customer names are added to a queue and served in order.**

package CollectionCoding;

import java.util.LinkedList;

import java.util.Queue;

import java.util.Scanner;

public class TicketBookingSystem {

public static void main(String[] args) {

Queue<String> customerQueue = new LinkedList<>();

Scanner sc = new Scanner(System.in);

String name;

System.out.println("Enter customer names for ticket booking (type 'serve' to serve next, 'exit' to quit):");

while (true) {

name = sc.nextLine();

if (name.equalsIgnoreCase("exit")) {

break;

} else if (name.equalsIgnoreCase("serve")) {

if (!customerQueue.isEmpty()) {

System.out.println("Serving: " + customerQueue.poll());

} else {

System.out.println("No customers in queue.");

}

} else {

customerQueue.offer(name);

System.out.println(name + " added to booking queue.");

}

}

sc.close();

}

}

1. **Iterator Interface**

**Write a program to iterate through a list using Iterator.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.Iterator;

public class IteratorDemo {

public static void main(String[] args) {

ArrayList<String> names = new ArrayList<>();

names.add("Alice");

names.add("Bob");

names.add("Charlie");

Iterator<String> iterator = names.iterator();

System.out.println("Iterating through list:");

while (iterator.hasNext()) {

System.out.println(iterator.next());

}

}

}

**Demonstrate removing an element from a list while iterating using Iterator.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.Iterator;

public class IteratorRemoveDemo {

public static void main(String[] args) {

ArrayList<String> items = new ArrayList<>();

items.add("Apple");

items.add("Banana");

items.add("Cherry");

Iterator<String> iterator = items.iterator();

while (iterator.hasNext()) {

String item = iterator.next();

if (item.equalsIgnoreCase("Banana")) {

iterator.remove();

}

}

System.out.println("List after removal:");

for (String item : items) {

System.out.println(item);

}

}

}

**Show how to use ListIterator to iterate in both directions.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.ListIterator;

public class ListIteratorBidirectional {

public static void main(String[] args) {

ArrayList<String> colors = new ArrayList<>();

colors.add("Red");

colors.add("Green");

colors.add("Blue");

ListIterator<String> listIterator = colors.listIterator();

System.out.println("Forward iteration:");

while (listIterator.hasNext()) {

System.out.println(listIterator.next());

}

System.out.println("Backward iteration:");

while (listIterator.hasPrevious()) {

System.out.println(listIterator.previous());

}

}

}

**Design a program that reads a list of book titles and removes those starting with a specific letter using an iterator.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.Scanner;

public class BookTitleFilter {

public static void main(String[] args) {

ArrayList<String> books = new ArrayList<>();

Scanner sc = new Scanner(System.in);

books.add("Algorithms");

books.add("Brave New World");

books.add("Clean Code");

books.add("Design Patterns");

System.out.print("Enter the starting letter to remove books: ");

String letter = sc.nextLine().toLowerCase();

Iterator<String> iterator = books.iterator();

while (iterator.hasNext()) {

String book = iterator.next();

if (book.toLowerCase().startsWith(letter)) {

iterator.remove();

}

}

System.out.println("Books after removal:");

for (String book : books) {

System.out.println(book);

}

sc.close();

}

}

**Create a program that reverses the elements in a list using ListIterator.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.ListIterator;

public class ListReverser {

public static void main(String[] args) {

ArrayList<String> cities = new ArrayList<>();

cities.add("New York");

cities.add("London");

cities.add("Paris");

cities.add("Tokyo");

System.out.println("Original list:");

for (String city : cities) {

System.out.println(city);

}

System.out.println("Reversed list:");

ListIterator<String> listIterator = cities.listIterator(cities.size());

while (listIterator.hasPrevious()) {

System.out.println(listIterator.previous());

}

}

}

1. **Sorting and Searching Collections**

**Sort an ArrayList of integers in ascending and descending order.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.Collections;

public class IntegerSortDemo {

public static void main(String[] args) {

ArrayList<Integer> numbers = new ArrayList<>();

numbers.add(30);

numbers.add(10);

numbers.add(50);

numbers.add(20);

Collections.sort(numbers);

System.out.println("Ascending: " + numbers);

Collections.sort(numbers, Collections.reverseOrder());

System.out.println("Descending: " + numbers);

}

}

**Use Collections.binarySearch() to find an element in a sorted list.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.Collections;

public class BinarySearchDemo {

public static void main(String[] args) {

ArrayList<String> fruits = new ArrayList<>();

fruits.add("Mango");

fruits.add("Apple");

fruits.add("Banana");

fruits.add("Orange");

Collections.sort(fruits);

System.out.println("Sorted list: " + fruits);

int index = Collections.binarySearch(fruits, "Banana");

if (index >= 0) {

System.out.println("Found 'Banana' at index: " + index);

} else {

System.out.println("'Banana' not found.");

}

}

}

**Sort a list of custom objects like Employees by name using Comparator.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

class Employee {

String name;

int id;

Employee(String name, int id) {

this.name = name;

this.id = id;

}

public String toString() {

return name + " (ID: " + id + ")";

}

}

public class EmployeeSortByName {

public static void main(String[] args) {

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

employees.add(new Employee("Charlie", 103));

employees.add(new Employee("Alice", 101));

employees.add(new Employee("Bob", 102));

Collections.sort(employees, Comparator.comparing(e -> e.name));

System.out.println("Employees sorted by name:");

for (Employee e : employees) {

System.out.println(e);

}

}

}

**You have a list of products with prices. Sort them by price and then search for a product within a specific price range.**

package CollectionCoding;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.Scanner;

class Product {

String name;

double price;

Product(String name, double price) {

this.name = name;

this.price = price;

}

public String toString() {

return name + " - $" + price;

}

}

public class ProductSortSearch {

public static void main(String[] args) {

ArrayList<Product> products = new ArrayList<>();

products.add(new Product("Laptop", 1200));

products.add(new Product("Phone", 800));

products.add(new Product("Mouse", 25));

products.add(new Product("Monitor", 300));

products.sort(Comparator.comparingDouble(p -> p.price));

System.out.println("Products sorted by price:");

for (Product p : products) {

System.out.println(p);

}

Scanner sc = new Scanner(System.in);

System.out.print("\nEnter min price: ");

double min = sc.nextDouble();

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

double max = sc.nextDouble();

System.out.println("\nProducts within price range:");

for (Product p : products) {

if (p.price >= min && p.price <= max) {

System.out.println(p);

}

}

sc.close();

}

}

**Build a leaderboard system that keeps players sorted by scores (highest first). Allow searching for a specific player's rank.**

package CollectionCoding;

import java.util.\*;

class Player {

String name;

int score;

Player(String name, int score) {

this.name = name;

this.score = score;

}

public String toString() {

return name + " - " + score;

}

}

public class Leaderboard {

public static void main(String[] args) {

ArrayList<Player> players = new ArrayList<>();

players.add(new Player("Alice", 200));

players.add(new Player("Bob", 150));

players.add(new Player("Charlie", 300));

players.add(new Player("Diana", 180));

players.sort((p1, p2) -> Integer.compare(p2.score, p1.score));

System.out.println("Leaderboard:");

for (int i = 0; i < players.size(); i++) {

System.out.println((i + 1) + ". " + players.get(i));

}

Scanner sc = new Scanner(System.in);

System.out.print("\nEnter player name to find rank: ");

String nameToSearch = sc.nextLine();

boolean found = false;

for (int i = 0; i < players.size(); i++) {

if (players.get(i).name.equalsIgnoreCase(nameToSearch)) {

System.out.println(nameToSearch + "'s Rank: " + (i + 1));

found = true;

break;

}

}

if (!found) {

System.out.println("Player not found.");

}

sc.close();

}

}