**Assignment 6**

**Set A**

1. Write a Java program to join two array lists. Construct a linked List containing names of months: January, February, March and April. Then extend your program to do the following: Display the contents of the List using an Iterator Display the contents of the List in reverse order using a ListIterator Create another list containing May and June. Insert the elements of this list between February and March.

import java.util.\*;

public class MonthListExample

{

public static void main(String[] args)

{

ArrayList<String> list1 = new ArrayList<>(Arrays.asList("January", "February"));

ArrayList<String> list2 = new ArrayList<>(Arrays.asList("March", "April"));

list1.addAll(list2);

System.out.println("Combined ArrayList: " + list1);

LinkedList<String> months = new LinkedList<>(Arrays.asList("January", "February", "March", "April"));

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

System.out.print("Months using Iterator: ");

while (iterator.hasNext())

{

System.out.print(iterator.next() + " ");

}

System.out.println();

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

System.out.print("Months in reverse order using ListIterator: ");

while (listIterator.hasPrevious())

{

System.out.print(listIterator.previous() + " ");

}

System.out.println();

List<String> newMonths = new ArrayList<>(Arrays.asList("May", "June"));

months.addAll(2, newMonths);

System.out.println("Updated LinkedList after insertion: " + months);

}

}

output:

Combined ArrayList: [January, February, March, April]

Months using Iterator: January February March April

Months in reverse order using ListIterator: April March February January

Updated LinkedList after insertion: [January, February, May, June, March, April]

Q.2 Accept n integers from the user and store them in a collection. Display them in the sorted order. The collection should not accept duplicate elements. (Use a suitable collection). Search for a particular element using a predefined search method in the Collection framework.

import java.util.\*;

public class UniqueSortedIntegers

{

public static void main(String[] args)

{

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of integers you want to input: ");

int n = scanner.nextInt();

Set<Integer> numbers = new TreeSet<>();

System.out.println("Enter the integers:");

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

int num = scanner.nextInt();

numbers.add(num);

}

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

System.out.print("Enter the number you want to search for: ");

int searchElement = scanner.nextInt();

if (numbers.contains(searchElement))

{

System.out.println("Element " + searchElement + " is present in the collection.");

}

else

{

System.out.println("Element " + searchElement + " is not present in the collection.");

}

scanner.close();

}

}

output:

Enter the number of integers you want to input: 6

Enter the integers:

23

12

10

29

7

1

Numbers in sorted order: [1, 7, 10, 12, 23, 29]

Enter the number you want to search for: 23

Element 23 is present in the collection.

Q.3 Create a Hash table containing Customer name and Salary. Display the details of the hash table. Also search for a specific Customer and display Salary of that Customer.

import java.util.\*;

public class CustomerSalaryHashtable

{

public static void main(String[] args)

{

Hashtable<String, Double> customerSalaryTable = new Hashtable<>();

customerSalaryTable.put("Anurag", 50000.0);

customerSalaryTable.put("Ram", 55000.0);

customerSalaryTable.put("Radhika", 60000.0);

customerSalaryTable.put("Priya", 45000.0);

customerSalaryTable.put("Ashish", 65000.0);

System.out.println("Customer Details (Name and Salary):");

for (Map.Entry<String, Double> entry : customerSalaryTable.entrySet())

{

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

}

Scanner scanner = new Scanner(System.in);

System.out.print("\nEnter the customer name to search for: ");

String searchName = scanner.nextLine();

if (customerSalaryTable.containsKey(searchName))

{

double salary = customerSalaryTable.get(searchName);

System.out.println("Salary of " + searchName + ": " + salary);

}

else

{

System.out.println("Customer " + searchName + " not found in the database.");

}

scanner.close();

}

}

output:

Customer Details (Name and Salary):

Customer: Radhika, Salary: 60000.0

Customer: Priya, Salary: 45000.0

Customer: Ashish, Salary: 65000.0

Customer: Ram, Salary: 55000.0

Customer: Anurag, Salary: 50000.0

Enter the customer name to search for: Ashish

Salary of Ashish: 65000.0

Q.4 Write a Java program to convert a hash set to a tree set.

import java.util.\*;

public class HashSetToTreeSet

{

public static void main(String[] args)

{

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

hashSet.add(10);

hashSet.add(5);

hashSet.add(20);

hashSet.add(15);

hashSet.add(30);

System.out.println("Original HashSet: " + hashSet);

TreeSet<Integer> treeSet = new TreeSet<>(hashSet);

System.out.println("Converted TreeSet (sorted order): " + treeSet);

}

}

output:

Original HashSet: [20, 5, 10, 30, 15]

Converted TreeSet (sorted order): [5, 10, 15, 20, 30]

SET C

Q.1. Write a Java program to get the element in a tree set strictly greater than or equal to the given element.

import java.util.\*;

public class TreeSetCeilingExample

{

public static void main(String[] args)

{

TreeSet<Integer> treeSet = new TreeSet<>();

treeSet.add(10);

treeSet.add(20);

treeSet.add(30);

treeSet.add(40);

treeSet.add(50);

System.out.println("Original TreeSet: " + treeSet);

Scanner scanner = new Scanner(System.in);

System.out.print("Enter an element to find the element greater than or equal to it: ");

int element = scanner.nextInt();

Integer result = treeSet.ceiling(element);

if (result != null) {

System.out.println("The least element greater than or equal to " + element + " is: " + result);

} else {

System.out.println("No element found greater than or equal to " + element);

}

scanner.close();

}

}

output:

Enter an element to find the element greater than or equal to it: 25

The least element greater than or equal to 25 is: 30

Q.2. Create a hashtable containing Employee name & Employee ID. Display the details of the hashtable. Also search for a specific Employee name and display the Employee ID of that Employee.

import java.util.\*;

public class EmployeeHashTable

{

public static void main(String[] args)

{

Hashtable<String, Integer> employeeTable = new Hashtable<>();

employeeTable.put("Anurag", 101);

employeeTable.put("Ram", 102);

employeeTable.put("Charlie", 103);

employeeTable.put("Priya", 104);

employeeTable.put("Ashish", 105);

System.out.println("Employee Details (Name and Employee ID):");

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

{

System.out.println("Employee Name: " + entry.getKey() + ", Employee ID: " + entry.getValue());

}

Scanner scanner = new Scanner(System.in);

System.out.print("\nEnter the employee name to search for: ");

String searchName = scanner.nextLine();

if (employeeTable.containsKey(searchName))

{

int employeeId = employeeTable.get(searchName);

System.out.println("Employee ID of " + searchName + ": " + employeeId);

}

else

{

System.out.println("Employee " + searchName + " not found in the database.");

}

scanner.close();

}

}

output:

Enter the employee name to search for: Anurag

Employee ID of Anurag: 101

Q.3 Write a Java program to retrieve, but not remove, the first element of a linked list.

import java.util.\*;

public class RetrieveFirstElement

{

public static void main(String[] args)

{

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

linkedList.add("PineApple");

linkedList.add("Banana");

linkedList.add("Cherry");

linkedList.add("Grapes");

System.out.println("Original LinkedList: " + linkedList);

String firstElement = linkedList.getFirst();

System.out.println("The first element in the LinkedList is: " + firstElement);

System.out.println("LinkedList after retrieving the first element: " + linkedList);

}

}

output:

The first element in the LinkedList is: PineApple

LinkedList after retrieving the first element: [PineApple, Banana, Cherry, Grapes]

Q.4.Write a Java program to group students based on their departments and further sorts them by their scores.

import java.util.\*;

class Student {

private String name;

private int score;

public Student(String name, int score) {

this.name = name;

this.score = score;

}

public String getName() {

return name;

}

public int getScore() {

return score;

}

public String toString() {

return name + " (" + score + ")";

}

}

public class GroupStudentsByDepartment

{

public static void main(String[] args)

{

Map<String, List<Student>> departmentMap = new HashMap<>();

addStudentToDepartment(departmentMap, "Computer Science", new Student("Rutuja", 85));

addStudentToDepartment(departmentMap, "Computer Science", new Student("Mohan", 92));

addStudentToDepartment(departmentMap, "Mathematics", new Student("Charlie", 78));

addStudentToDepartment(departmentMap, "Mathematics", new Student("Sharad", 88));

addStudentToDepartment(departmentMap, "Physics", new Student("Kritika", 90));

for (Map.Entry<String, List<Student>> entry : departmentMap.entrySet())

{

String department = entry.getKey();

List<Student> students = entry.getValue();

students.sort((s1, s2) -> Integer.compare(s2.getScore(), s1.getScore()));

System.out.println("Department: " + department);

for (Student student : students) {

System.out.println(" " + student);

}

System.out.println();

}

}

private static void addStudentToDepartment(Map<String, List<Student>> departmentMap, String department, Student student) {

departmentMap.computeIfAbsent(department, k -> new ArrayList<>()).add(student);

}

}

output:

Department: Computer Science

Mohan (92)

Rutuja (85)

Department: Mathematics

Sharad (88)

Charlie (78)

Department: Physics

Kritika (90)