Collections

List(ArrayList)

**2. Search an Element**

Write a program to:

* Create an ArrayList of integers.
* Ask the user to enter a number.
* Check if the number exists in the list.

Code: **package** Collection\_framework;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** Searchforelement {

**public** **static** **void** main(String[] args) {

ArrayList<Integer> l1=**new** ArrayList<Integer>(List.*of*(4,8,2,3,9,7));

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("Enter a number:");

**int** n=sc.nextInt();

**if**(l1.contains(n)) {

System.***out***.println("The entered number is exists in the list.");

}**else** {

System.***out***.println("The entered number is not exists in the list:");

}

}

}

Output; Enter a number:

7

The entered number is exists in the list.

**3. Remove Specific Element**

Write a program to:

* Create an ArrayList of Strings.
* Add 5 fruits.
* Remove a specific fruit by name.
* Display the updated list.

Code: **package** Collection\_framework;

**import** java.util.ArrayList;

**public** **class** Removespecific\_Element {

**public** **static** **void** main(String[] args) {

ArrayList<String> l1=**new** ArrayList<String>();

l1.add("Apple");

l1.add("Mango");

l1.add("Graphes");

l1.add("Kiwi");

l1.add("orange");

System.***out***.println("Original list:"+l1);

l1.remove("Mango");

System.***out***.println("Updated list after removing spefic element by name:"+l1);

}

}

Output; Original list:[Apple, Mango, Graphes, Kiwi, orange]

Updated list after removing spefic element by name:[Apple, Graphes, Kiwi, orange]

**4. Sort Elements**

Write a program to:

* Create an ArrayList of integers.
* Add at least 7 random numbers.
* Sort the list in ascending order.
* Display the sorted list.

Code: **package** Collection\_framework;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** Sort\_Elements {

**public** **static** **void** main(String[] args) {

ArrayList<Integer> l1=**new** ArrayList<Integer>(List.*of*(9,4,7,2,6,1,0,5,8));

l1.sort(**null**);

System.***out***.println("Sorted list is:"+l1);

}

}

Output; Sorted list is:[0, 1, 2, 4, 5, 6, 7, 8, 9]

**5. Reverse the ArrayList**

Write a program to:

* Create an ArrayList of characters.
* Add 5 characters.
* Reverse the list using Collections.reverse() and display it.

Code; **package** Collection\_framework;

**import** java.util.ArrayList;

**import** java.util.Collections;

**public** **class** Reversethe\_ArrayList {

**public** **static** **void** main(String[] args) {

ArrayList<Character> c=**new** ArrayList<Character>();

c.add('R');

c.add('E');

c.add('D');

c.add('D');

c.add('Y');

Collections.*reverse*(c);

System.***out***.println("Reversed list of characters is:"+c);

}

}

Output; Reversed list of characters is:[Y, D, D, E, R]

**6. Update an Element**

Write a program to:

* Create an ArrayList of subjects.
* Replace one of the subjects (e.g., “Math” to “Statistics”).
* Print the list before and after the update.

Code: **package** Collection\_framework;

**import** java.util.ArrayList;

**public** **class** Updatean\_Element {

**public** **static** **void** main(String[] args) {

ArrayList<String> subjects=**new** ArrayList<String>();

subjects.add("science");

subjects.add("Math");

subjects.add("social");

subjects.add("telugu");

subjects.add("Hindi");

System.***out***.println("List before updating:"+subjects);

**int** index = subjects.indexOf("Math");

**if** (index != -1) {

subjects.set(index, "Statistics");

} **else** {

System.***out***.println("\"Math\" not found in the list.");

}

System.***out***.println("List after updating:"+subjects);

}

}

Output; List before updating:[science, Math, social, telugu, Hindi]

List after updating:[science, Statistics, social, telugu, Hindi]

**7. Remove All Elements**

Write a program to:

* Create an ArrayList of integers.
* Add multiple elements.
* Remove all elements using clear() method.
* Display the size of the list.

Code: **package** MAp\_collections;

**import** java.util.ArrayList;

**public** **class** Removeallelements {

**public** **static** **void** main(String[] args) {

ArrayList<Integer> a=**new** ArrayList<Integer>();

a.add(0, 7);

a.add(1,4);

a.add(2,9);

a.add(3,8);

**int** b=a.size();

System.***out***.println("Size of the arraylist is:"+b);

a.clear();

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

}

}

Output; Size of the arraylist is:4

[]

**8. Iterate using Iterator**

Write a program to:

* Create an ArrayList of cities.
* Use Iterator to display each city.

Code; **package** MAp\_collections;

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** iterato1r {

**public** **static** **void** main(String[] args) {

ArrayList<String> city=**new** ArrayList<String>();

city.add("Hyderabad");

city.add("Mumbai");

city.add("Pune");

city.add("Jaipur");

city.add("Chennai");

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

**while**(iterator.hasNext()) {

String city1=iterator.next();

System.***out***.println(city1);

}

}

}

Output; Hyderabad

Mumbai

Pune

Jaipur

Chennai

**9. Store Custom Objects**

Write a program to:

* Create a class Student with fields: id, name, and marks.
* Create an ArrayList of Student objects.
* Add at least 3 students.
* Display the details using a loop.

Code; **package** MAp\_collections;

**import** java.util.ArrayList;

**class** Student4{

**private** **int** id;

**private** String name;

**private** **double** marks;

**public** Student4(**int** id,String name,**double** marks){

**this**.id=id;

**this**.name=name;

**this**.marks=marks;

}

**public** **int** getId() {

**return** id;

}

**public** String getName() {

**return** name;

}

**public** **double** getmarks() {

**return** marks;

}

**public** String toString() {

**return** "ID:"+id +", Name:"+name+", Marks:"+marks;

}

}

**public** **class** Storecustomobject {

**public** **static** **void** main(String[] args) {

ArrayList<Student4> stdlst=**new** ArrayList<>();

stdlst.add(**new** Student4(101,"Ravi",98.0d));

stdlst.add(**new** Student4(102,"Raju",95.0d));

stdlst.add(**new** Student4(101,"Ramesh",90.0d));

stdlst.add(**new** Student4(101,"Rajiv",95.6d));

System.***out***.println("----Student Details-------");

**for**(Student4 student:stdlst) {

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

}

}

}

Output; ----Student Details-------

ID:101, Name:Ravi, Marks:98.0

ID:102, Name:Raju, Marks:95.0

ID:101, Name:Ramesh, Marks:90.0

ID:101, Name:Rajiv, Marks:95.6

**10. Copy One ArrayList to Another**

Write a program to:

* Create an ArrayList with some elements.
* Create a second ArrayList.
* Copy all elements from the first to the second using addAll() method.

Code; **package** MAp\_collections;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** Copyonearraylisttoanother {

**public** **static** **void** main(String[] args) {

ArrayList<Integer> a1= **new** ArrayList<Integer>(List.*of*(4,9,1,3,2,5,0));

ArrayList<Integer> a2=**new** ArrayList<>();

System.***out***.println("list of elements before copying :"+a1);

a2.addAll(a1);

System.***out***.println("list of elements after copying :"+a2);

}

}

Output; list of elements before copying :[4, 9, 1, 3, 2, 5, 0]

list of elements after copying :[4, 9, 1, 3, 2, 5, 0]

List(LinkedList)

**1. Create and Display a LinkedList**

Write a program to:

* Create a LinkedList of Strings.
* Add five colors to it.
* Display the list using a for-each loop.

Code; **package** MAp\_collections;

**import** java.util.LinkedList;

**import** java.util.List;

**public** **class** LinkedList\_example {

**public** **static** **void** main(String[] args) {

LinkedList<String> l1= **new** LinkedList<String>();

l1.add("Red");

l1.add("Yellow");

l1.add("orange");

l1.add("white");

l1.add("Blue");

l1.add("violet");

**for**(String lt:l1) {

//System.out.println(l1);

}

System.***out***.println(l1);

}

}

Output; [Red, Yellow, orange, white, Blue, violet]

**2. Add Elements at First and Last Position**

Write a program to:

* Create a LinkedList of integers.
* Add elements at the beginning and at the end.
* Display the updated list.

Code; **package** MAp\_collections;

**import** java.util.LinkedList;

**import** java.util.List;

**public** **class** Add\_elements\_at\_first\_and\_last {

**public** **static** **void** main(String[] args) {

LinkedList<Integer> l1=**new** LinkedList<Integer>(List.*of*(2,0,4,8,6,5));

System.***out***.println("List of elements before adding elements :"+l1);

l1.add(0, 1);

l1.add(l1.size(),7);

System.***out***.println("List of elements after adding elements :"+l1);

}

}

Output; List of elements before adding elements :[2, 0, 4, 8, 6, 5]

List of elements after adding elements :[1, 2, 0, 4, 8, 6, 5, 7]

**3. Insert Element at Specific Position**

Write a program to:

* Create a LinkedList of names.
* Insert a name at index 2.
* Display the list before and after insertion.

Code; **package** MAp\_collections;

**import** java.util.LinkedList;

**import** java.util.List;

**public** **class** Insert\_elements\_at\_specific\_position {

**public** **static** **void** main(String[] args) {

LinkedList<String> Names=**new** LinkedList<String>();

Names.add("Jagadeesh");

Names.add("suresh");

Names.add("Ramesh");

Names.add("Rajesh");

Names.add("Somesh");

System.***out***.println("List of names before adding a element at specific position:"+Names);

Names.add(2, "Naresh");

System.***out***.println("List of names after adding a element at specific position:"+Names);

}

}

Output; List of names before adding a element at specific position:[Jagadeesh, suresh, Ramesh, Rajesh, Somesh]

List of names after adding a element at specific position:[Jagadeesh, suresh, Naresh, Ramesh, Rajesh, Somesh]

**4. Remove Elements**

Write a program to:

* Create a LinkedList of animal names.
* Remove the first and last elements.
* Remove a specific element by value.
* Display the list after each removal.

Code; **package** MAp\_collections;

**import** java.util.LinkedList;

**import** java.util.List;

**public** **class** Remove\_elements {

**public** **static** **void** main(String[] args) {

LinkedList<String> an=**new** LinkedList<String>();

an.add("Tiger");

an.add("Dog");

an.add("Lion");

an.add("Elepant");

an.add("deer");

an.add("fox");

System.***out***.println("List of animals before removing :"+an);

an.remove(0);

**int** s=an.size();

an.remove(s-1);

System.***out***.println("List of animals after removing :"+an);

}

}

Output; List of animals before removing :[Tiger, Dog, Lion, Elepant, deer, fox]

List of animals after removing :[Dog, Lion, Elepant, deer]

**5. Search for an Element**

Write a program to:

* Create a LinkedList of Strings.
* Ask the user for a string to search.
* Display if the string is found or not.

Code: **package** MAp\_collections;

**import** java.util.LinkedList;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** Search\_for\_element {

**public** **static** **void** main(String[] args) {

LinkedList<String> l1=**new** LinkedList<String>(List.*of*("mango","apple","graphes","kiwi","banana"));

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("Enter a String:");

String n=sc.nextLine();

n.toLowerCase();

**if**(l1.contains(n)) {

System.***out***.println("The entered String is exists in the list.");

}**else** {

System.***out***.println("The entered String is not exists in the list:");

}

}

}

Output; Enter a String:

jagadeesh

The entered String is not exists in the list:

**6. Iterate using ListIterator**

Write a program to:

* Create a LinkedList of cities.
* Use ListIterator to display the list in both forward and reverse directions.

Code; **package** MAp\_collections;

**import** java.util.LinkedList;

**import** java.util.ListIterator;

**import** java.util.Iterator;

**public** **class** CityListIterator {

**public** **static** **void** main(String[] args) {

LinkedList<String> city=**new** LinkedList<String>();

city.add("Hyderabad");

city.add("Mumbai");

city.add("Pune");

city.add("Jaipur");

city.add("Chennai");

ListIterator<String> forwarditerator=city.listIterator();

System.***out***.println("List in forward :");

**while**(forwarditerator.hasNext()) {

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

}

System.***out***.println("List in reverse order :");

**while**(forwarditerator.hasPrevious()) {

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

}

}

}

Output; List in forward :

Hyderabad

Mumbai

Pune

Jaipur

Chennai

List in reverse order :

Chennai

Jaipur

Pune

Mumbai

Hyderabad

**7. Sort a LinkedList**

Write a program to:

* Create a LinkedList of integers.
* Add unsorted numbers.
* Sort the list using Collections.sort().
* Display the sorted list.

Code; **package** MAp\_collections;

**import** java.util.LinkedList;

**import** java.util.Collections;

**import** java.util.List;

**public** **class** Sort\_Linkedlist {

**public** **static** **void** main(String[] args) {

LinkedList<Integer> l1=**new** LinkedList<Integer>(List.*of*(4,9,2,7,1,5,8));

Collections.*sort*(l1);

System.***out***.println("Sorted list is :"+l1);

}

}

Output; Sorted list is :[1, 2, 4, 5, 7, 8, 9]

**8. Convert LinkedList to ArrayList**

Write a program to:

* Create a LinkedList of Strings.
* Convert it into an ArrayList.
* Display both the LinkedList and ArrayList.

Code; **package** MAp\_collections;

**import** java.util.LinkedList;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** Linkedlist\_to\_arraylist {

**public** **static** **void** main(String[] args) {

LinkedList<String> l1=**new** LinkedList<String>(List.*of*("Lion","Tiger","Fox","elephant","Deer"));

ArrayList<String> l11=**new** ArrayList<String>(l1);

System.***out***.println("LinkedList is :"+l1);

System.***out***.println("ArrayList is :"+l11);

}

}

Output; LinkedList is :[Lion, Tiger, Fox, elephant, Deer]

ArrayList is :[Lion, Tiger, Fox, elephant, Deer]

**9. Store Custom Objects in LinkedList**

Write a program to:

* Create a class Book with fields: id, title, and author.
* Create a LinkedList of Book objects.
* Add 3 books and display their details using a loop.

Code; **package** MAp\_collections;

**import** java.util.LinkedList;

**class** Book{

**int** id;

String author;

String title;

Book(**int** id,String author,String title){

**this**.id=id;

**this**.author=author;

**this**.title=title;

}

**public** **void** displayDetails() {

System.***out***.println("Book ID: "+id+", Title:" +title+", Author:"+author);

}

}

**public** **class** Store\_custom\_objects\_in\_LinkedList {

**public** **static** **void** main(String[] args) {

LinkedList<Book> b1=**new** LinkedList<>();

b1.add(**new** Book(1,"The Alchemist","Paulo Coelho"));

b1.add(**new** Book(2,"1984","George Orwell"));

b1.add(**new** Book(3,"To kill a Mockingbird","Harper Lee"));

**for**(Book book:b1) {

book.displayDetails();

}

}

}

Output; Book ID: 1, Title:Paulo Coelho, Author:The Alchemist

Book ID: 2, Title:George Orwell, Author:1984

Book ID: 3, Title:Harper Lee, Author:To kill a Mockingbird

**10. Clone a LinkedList**

Write a program to:

* Create a LinkedList of numbers.
* Clone it using the clone() method.
* Display both original and cloned lists.

Code; **package** MAp\_collections;

**import** java.util.LinkedList;

**import** java.util.Scanner;

**public** **class** Clone\_a\_LinkedList {

**public** **static** **void** main(String[] args) {

Scanner scanner=**new** Scanner(System.***in***);

LinkedList<Integer> l2=**new** LinkedList<Integer>();

System.***out***.print("Enter the number of elements: ");

**int** numElements = scanner.nextInt();

scanner.nextLine(); // Consume the newline character after nextInt()

System.***out***.println("Enter " + numElements + " elements:");

**for** (**int** i = 0; i < numElements; i++) {

**int** element = scanner.nextInt();

l2.add(element);

}

System.***out***.println("Original linkedlist is :"+l2);

l2.clone();

System.***out***.println("Cloned linkedlist is:"+l2);

}

}

Output; Enter the number of elements: 5

Enter 5 elements:

6 9 0 4 7

Original linkedlist is :[6, 9, 0, 4, 7]

Cloned linkedlist is:[6, 9, 0, 4, 7]

Vector

* **Create a Vector of integers** and perform the following operations:
* Add 5 integers to the Vector.
* Insert an element at the 3rd position.
* Remove the 2nd element.
* Display the elements using Enumeration.

Code; **package** Collection\_framework;

**import** java.util.\*;

**public** **class** integer\_vector {

**public** **static** **void** main(String[] args) {

Vector<Integer> v=**new** Vector<Integer>();

v.add(4);

v.add(9);

v.add(5);

v.add(3);

v.add(1);

v.add(6);

v.add(3, 0);

v.remove(2);

Enumeration<Integer> en=v.elements();

**while**(en.hasMoreElements()) {

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

}

}

}

Output; 4

9

0

3

1

6

* **Create a Vector of Strings** and:
* Add at least 4 names.
* Check if a specific name exists in the vector.
* Replace one name with another.
* Clear all elements from the vector.
* **Write a program** to:
* Copy all elements from one Vector to another Vector.
* Compare both vectors for equality.

Code; **package** Collection\_framework;

**import** java.util.\*;

**public** **class** String\_vector {

**public** **static** **void** main(String[] args) {

Vector<String> v=**new** Vector<String>();

v.add("Raju");

v.add("Ravi");

v.add("Ramesh");

v.add("Rajesh");

System.***out***.println("2. Added at least 4 names (Raju,Ravi,Ramesh,Rajesh).");

System.***out***.print("Current names: ");

**for** (String name : v) {

System.***out***.print(name + " ");

}

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

String searchName = "Charlie";

**if** (v.contains(searchName)) {

System.***out***.println("3. '" + searchName + "' found in the Vector.");

} **else** {

System.***out***.println("3. '" + searchName + "' not found in the Vector.");

}

String oldName = "David";

String newName = "Grace";

**int** indexToReplace = v.indexOf(oldName); // Find the index of the old name

**if** (indexToReplace != -1) { // If the old name exists

v.set(indexToReplace, newName); // Use set() to replace at a specific index

System.***out***.println("4. Replaced '" + oldName + "' with '" + newName + "'.");

} **else** {

System.***out***.println("4. '" + oldName + "' not found for replacement.");

}

System.***out***.print("Names after replacement: ");

**for** (String name : v) {

System.***out***.print(name + " ");

}

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

Vector<String> copiedNames;

copiedNames = **new** Vector<>(v);

System.***out***.println("5. Copied elements from 'names' to 'copiedNames'.");

System.***out***.print("Copied names: ");

**for** (String name : copiedNames) {

System.***out***.print(name + " ");

}

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

**if** (v.equals(copiedNames)) {

System.***out***.println("6. 'names' and 'copiedNames' Vectors are equal.");

} **else** {

System.***out***.println("6. 'names' and 'copiedNames' Vectors are not equal.");

}

v.clear();

System.***out***.println("7. Cleared all elements from 'names' Vector.");

System.***out***.println("Names Vector size after clearing: " + v.size());

}

}

Output; 2. Added at least 4 names (Raju,Ravi,Ramesh,Rajesh).

Current names: Raju Ravi Ramesh Rajesh

3. 'Charlie' not found in the Vector.

4. 'David' not found for replacement.

Names after replacement: Raju Ravi Ramesh Rajesh

5. Copied elements from 'names' to 'copiedNames'.

Copied names: Raju Ravi Ramesh Rajesh

6. 'names' and 'copiedNames' Vectors are equal.

7. Cleared all elements from 'names' Vector.

Names Vector size after clearing: 0

* **Write a method** that takes a Vector<Integer> and returns the **sum of all elements**.

**Code; package** MAp\_collections;

**import** java.util.Vector;

**public** **class** sum\_of\_elements {

**public** **static** **void** main(String[] args) {

Vector<Integer> v1=**new** Vector<Integer>();

v1.add(3);

v1.add(9);

v1.add(7);

v1.add(2);

v1.add(4);

v1.add(1);

**int** sum=0;

**for**(Integer i:v1) {

sum+=i;

}

System.***out***.println("Sum of all elements is :"+sum);

}

}

Output; Sum of all elements is :26

**Stack**

* Understand how to use the Stack class for LIFO (Last In, First Out) operations.
* **Create a Stack of integers** and:
* Push 5 elements.
* Pop the top element.
* Peek the current top.
* Check if the stack is empty.

Code; **package** MAp\_collections;

**import** java.util.Stack;

**import** java.util.Scanner;

**public** **class** Stack\_example{

**public** **static** **void** main(String[] args) {

Stack<Integer> s1=**new** Stack<Integer>();

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Enter the length of stack:");

**int** n=scan.nextInt();

System.***out***.println("Enter a "+n+" element in a stack:");

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

**int** q=scan.nextInt();

s1.add(q);

}

System.***out***.println("Stack is given as:"+s1);

s1.remove(0);

System.***out***.println("Stack is after pop the top element as:"+s1);

System.***out***.println("Stack is empty :"+s1.isEmpty());

}

}

**Output;** Enter the length of stack:

5

Enter a 5 element in a stack:

3 5 7 9 0

Stack is given as:[3, 5, 7, 9, 0]

Stack is after pop the top element as:[5, 7, 9, 0]

Stack is empty :false

**Reverse a string using Stack**:

* Input a string from the user.
* Use a stack to reverse and print the string.

Code; **package** MAp\_collections;

**import** java.util.Stack;

**import** java.util.Scanner;

**public** **class** Reverse\_a\_String\_using\_stack {

**public** **static** **void** main(String[] args) {

Stack<Character> s1=**new** Stack<Character>();

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Enter a string;");

String s=scan.nextLine();

**for** (**char** ch : s.toCharArray()) {

s1.push(ch);

}

StringBuilder reversed = **new** StringBuilder();

**while** (!s1.isEmpty()) {

reversed.append(s1.pop());

}

System.***out***.println("The reversed string is: " + reversed.toString());

}

}

Output; Enter a string;

jagadeesh

The reversed string is: hseedagaj

* **Use Stack to check for balanced parentheses** in an expression.
* Input: (a+b) \* (c-d)
* Output: Valid or Invalid expression
* **Convert a decimal number to binary using Stack**.

**Code**

import java.util.Stack;

import java.util.Scanner;

public class ReverseStringUsingStack {

public static void main(String[] args) {

Stack<Character> stack = new Stack<>();

Scanner scan = new Scanner(System.in);

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

String input = scan.nextLine();

// Push all characters of the string into the stack

for (char ch : input.toCharArray()) {

stack.push(ch);

}

// Pop all characters from the stack and append to result

StringBuilder reversed = new StringBuilder();

while (!stack.isEmpty()) {

reversed.append(stack.pop());

}

System.out.println("The reversed string is: " + reversed.toString());

}

}

import java.util.\*;

public class StackDemo {

// Check if brackets match

private static boolean isMatching(char open, char close) {

return (open == '(' && close == ')')

|| (open == '{' && close == '}')

|| (open == '[' && close == ']');

}

public static boolean isBalanced(String expr) {

Stack<Character> stack = new Stack<>();

for (char c : expr.toCharArray()) {

if (c == '(' || c == '{' || c == '[') {

stack.push(c);

} else if (c == ')' || c == '}' || c == ']') {

if (stack.isEmpty() || !isMatching(stack.pop(), c)) {

return false;

}

}

}

return stack.isEmpty();

}

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.print("Enter an expression: ");

String expr = scan.nextLine();

System.out.println(isBalanced(expr) ? "Valid expression" : "Invalid expression");

// Then proceed to decimal-to-binary

}

}

Output; Enter a string;

jagadeesh

The reversed string is: hseedagaj

## 2. Decimal‑to‑Binary Conversion Using Stack

import java.util.\*;

public class StackDemo{

public static String decimalToBinary(int num) {

if (num == 0) return "0";

Stack<Integer> stk = new Stack<>();

while (num > 0) {

stk.push(num % 2);

num /= 2;

}

StringBuilder sb = new StringBuilder();

while (!stk.isEmpty()) {

sb.append(stk.pop());

}

return sb.toString();

}

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.print("Enter a decimal number to convert: ");

int n = scan.nextInt();

System.out.println("Binary equivalent: " + decimalToBinary(n));

}

}

HashSet

1. **Create a HashSet of Strings**:
   * Add 5 different city names.
   * Try adding a duplicate city and observe the output.
   * Iterate using an Iterator and print each city.

Code; **package** MAp\_collections;

**import** java.util.HashSet;

**import** java.util.Iterator;

**public** **class** Create\_hashset\_of\_string {

**public** **static** **void** main(String[] args) {

HashSet<String> s1=**new** HashSet<String>();

s1.add("Mumbai");

s1.add("Hyderabad");

s1.add("Chennai");

s1.add("Bangalore");

s1.add("Pune");

System.***out***.println("Given hashset is :"+s1);

**boolean** add=s1.add("Hyderabad");

System.***out***.println("Given hashset after adding duplicate :"+add);

Iterator<String> iterator=s1.iterator();

**for**(String i:s1) {

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

}

}

}

Output; Given hashset is :[Chennai, Pune, Mumbai, Hyderabad, Bangalore]

Given hashset after adding duplicate :false

Chennai

Pune

Mumbai

Hyderabad

Bangalore

1. **Perform operations**:
   * Remove an element.
   * Check if a city exists.
   * Clear the entire HashSet.

Code; **package** MAp\_collections;

**import** java.util.HashSet;

**import** java.util.Iterator;

**public** **class** Create\_hashset\_of\_string {

**public** **static** **void** main(String[] args) {

HashSet<String> s1=**new** HashSet<String>();

s1.add("Mumbai");

s1.add("Hyderabad");

s1.add("Chennai");

s1.add("Bangalore");

s1.add("Pune");

System.***out***.println("Given hashset is :"+s1);

**boolean** add=s1.add("Hyderabad");

System.***out***.println("Given hashset after adding duplicate :"+add);

Iterator<String> iterator=s1.iterator();

**for**(String i:s1) {

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

}

s1.remove("Bangalore");

System.***out***.println("Hashset after removing element:"+s1);

**boolean** cons=s1.contains("Chennai");

System.***out***.println("Hashset containing the element :"+cons);

s1.clear();

System.***out***.println("Hashset after clearing the hashset:"+s1);

}

}

Output; Given hashset is :[Chennai, Pune, Mumbai, Hyderabad, Bangalore]

Given hashset after adding duplicate :false

Chennai

Pune

Mumbai

Hyderabad

Bangalore

Hashset after removing element:[Chennai, Pune, Mumbai, Hyderabad]

Hashset containing the element :true

Hashset after clearing the hashset:[]

1. **Write a method** that takes a HashSet<Integer> and returns the maximum element.

**Code; package** MAp\_collections;

**import** java.util.HashSet;

**import** java.util.Iterator;

**import** java.lang.Iterable;

**import** java.util.Scanner;

**import** java.util.Collections;

**public** **class** Hashset\_integer {

**public** **static** Integer findMax(HashSet<Integer> set) {

**if** (set == **null** || set.isEmpty()) {

**return** **null**;

}

**return** Collections.*max*(set);

\*/

}

**public** **static** **void** main(String[] args) {

HashSet<Integer> s1=**new** HashSet<Integer>();

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Enter the size of hashset:");

**int** s=scan.nextInt();

System.***out***.println("Enter "+s +"elements to hashset:");

**for**(**int** i=0;i<s;i++) {

**int** p=scan.nextInt();

s1.add(p);

}

Integer max = *findMax*(s1);

**if** (max != **null**) {

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

} **else** {

System.***out***.println("HashSet is empty");

}

}

}

Output; Enter the size of hashset:

5

Enter 5elements to hashset:

2 6 8 9 0

Maximum element is: 9

**LinkedHashSet**

**1.Create a LinkedHashSet of Integers**:

* + Add numbers: 10, 5, 20, 15, 5.
  + Print the elements and observe the order.

Code; **package** MAp\_collections;

**import** java.util.LinkedHashSet;

**public** **class** LinkedHashSet\_demo {

**public** **static** **void** main(String[] args) {

LinkedHashSet<Integer> l1=**new** LinkedHashSet<Integer>();

l1.add(10);

l1.add(5);

l1.add(20);

l1.add(15);

l1.add(5);

System.***out***.println("LinkedHashset elements are: "+l1);

}

}

Output; LinkedHashset elements are: [10, 5, 20, 15]

1. **Create a LinkedHashSet of custom objects (e.g., Student with id and name)**:
   * Override hashCode() and equals() properly.
   * Add at least 3 Student objects.
   * Try adding a duplicate student and check if it gets added.

Code; **package** MAp\_collections;

**import** java.util.LinkedHashSet;

**import** java.util.Objects;

**public** **class** Student\_SET {

**int** id;

String name;

Student\_SET(**int** id,String name ){

**this**.id=id;

**this**.name=name;

}

**public** **int** hashCode() {

**return** Objects.*hash*(id,name);

}

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null** || getClass() != obj.getClass())

**return** **false**;

Student\_SET other = (Student\_SET) obj;

**return** id == other.id && Objects.*equals*(name, other.name);

}

**public** String toString() {

**return** "Student{id="+id+",name='" +name+"'}";

}

**public** **static** **void** main(String[] args) {

LinkedHashSet<Student\_SET>s=**new** LinkedHashSet<>();

Student\_SET s1=**new** Student\_SET(22, "Ravi");

Student\_SET s2=**new** Student\_SET(21, "Rajesh");

Student\_SET s3=**new** Student\_SET(23, "Ramesh");

Student\_SET s4=**new** Student\_SET(22, "Ravi");

s.add(s1);

s.add(s2);

s.add(s3);

**boolean** isAdded = s.add(s4); // Attempt to add duplicate

System.***out***.println("Students in LinkedHashSet:");

**for** (Student\_SET student : s) {

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

}

System.***out***.println("\nWas the duplicate student added? " + isAdded);

}

}

Output; Students in LinkedHashSet:

Student{id=22,name='Ravi'}

Student{id=21,name='Rajesh'}

Student{id=23,name='Ramesh'}

Was the duplicate student added? false

1. **Write a program** to:
   * Merge two LinkedHashSets and print the result.

Code; **package** MAp\_collections;

**import** java.util.LinkedHashSet;

**import** java.util.Collections;

**public** **class** Merge\_two\_LinkedHashSet {

**public** **static** **void** main(String[] args) {

LinkedHashSet<Integer> l1=**new** LinkedHashSet<Integer>();

l1.add(7);

l1.add(9);

l1.add(0);

l1.add(5);

System.***out***.println("LinkedHashSet one is:"+l1);

LinkedHashSet<Integer> l2=**new** LinkedHashSet<Integer>();

l2.add(77);

l2.add(98);

l2.add(10);

l2.add(55);

System.***out***.println("LinkedHashSet one is:"+l2);

LinkedHashSet<Integer> l3=**new** LinkedHashSet<Integer>();

l1.addAll(l2);

System.***out***.println("Linkedhashset after merging two linkedHashSets:"+l1);

}

}

Output; LinkedHashSet one is:[7, 9, 0, 5]

LinkedHashSet one is:[77, 98, 10, 55]

Linkedhashset after merging two linkedHashSets:[7, 9, 0, 5, 77, 98, 10, 55]

**TreeSet**

**1. Create a TreeSet of Strings**:

* + Add 5 country names in random order.
  + Print the sorted list of countries using TreeSet.

Code; **package** MAp\_collections;

**import** java.util.TreeSet;

**import** java.util.Collections;

**public** **class** Create\_treeset {

**public** **static** **void** main(String[] args) {

TreeSet<String> t1=**new** TreeSet<String>();

t1.add("Japan");

t1.add("China");

t1.add("India");

t1.add("USA");

t1.add("Russia");

t1.add("England");

System.***out***.println("Sorted list of countries :");

**for**(String s:t1) {

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

}

}

}

Output; Sorted list of countries :

China

England

India

Japan

Russia

USA

1. **Create a TreeSet of Integers**:
   * Add some numbers and print the first and last elements.
   * Find the elements lower than and higher than a given number using lower() and higher() methods.

Code; **package** MAp\_collections;

**import** java.util.TreeSet;

**import** java.util.Scanner;

**public** **class** Integer\_Treeset {

**public** **static** **void** main(String[] args) {

TreeSet<Integer> s=**new** TreeSet<Integer>();

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Enter a size of treeset:");

**int** n =scan.nextInt();

System.***out***.println("Enter "+ n +" elements");

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

**int** m=scan.nextInt();

s.add(m);

}

System.***out***.println("The treeset is :"+s);

**if** (s.isEmpty()) {

System.***out***.println("TreeSet is empty");

} **else** {

System.***out***.println("The first number of treeset is:"+s.first());

System.***out***.println("The last number of treeset is:"+s.last());

}

System.***out***.println("Enter a number:");

**int** w=scan.nextInt();

System.***out***.println("The numbers less than "+ w +" is:"+s.lower(w));

System.***out***.println("The numbers higher than "+ w +" is:"+s.higher(w));

}

}

Output; Enter a size of treeset:

4

Enter 4 elements

2 7 9 6

The treeset is :[2, 6, 7, 9]

The first number of treeset is:2

The last number of treeset is:9

Enter a number:

8

The numbers less than 8 is:7

The numbers higher than 8 is:9

1. **Create a TreeSet with a custom comparator**:
   * Sort strings in **reverse alphabetical order** using Comparator.

Code; **package** MAp\_collections;

**import** java.util.TreeSet;

**import** java.util.\*;

**public** **class** Createacustom\_Compartor {

**public** **static** **void** main(String[] args) {

TreeSet<String> set = **new** TreeSet<>(Comparator.*reverseOrder*());

// Adding some example strings in random order

set.add("Apple");

set.add("Banana");

set.add("Mango");

set.add("Cherry");

set.add("Date");

System.out.println(“Initial Treeset is:”+set);

// Printing the sorted TreeSet

System.***out***.println("TreeSet in reverse alphabetical order:");

**for** (String s : set) {

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

}

}

}

Output; TreeSet in reverse alphabetical order:

Mango

Date

Cherry

Banana

Apple

Queue

1. **Bank Queue Simulation**:
   * Create a queue of customer names using Queue<String>.
   * Add 5 customers to the queue.
   * Serve (remove) customers one by one and print the queue after each removal.

Code; **package** MAp\_collections;

**import** java.util.Collection;

**import** java.util.Iterator;

**import** java.util.LinkedList;

**import** java.util.Queue;

**public** **class** Bank\_Queue {

**public** **static** **void** main(String[] args) {

Queue<String> q=**new** LinkedList<>();

q.add("raju");

q.add("Somesh");

q.add("ravi");

q.add("ramesh");

q.add("rajini");

System.***out***.println("Entered queue is:");

**for**(String s:q) {

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

}

System.***out***.println("Queue after removal one by one:");

**while**(!q.isEmpty()) {

String served=q.poll();

System.***out***.println("Served:"+served);

System.***out***.println("Remaining Queue:"+q);

}

System.out.println(“Queue is empty”);

}

}

Output; Entered queue is:

raju

Somesh

ravi

ramesh

rajini

Queue after removal one by one:

Served:raju

Remaining Queue:[Somesh, ravi, ramesh, rajini]

Served:Somesh

Remaining Queue:[ravi, ramesh, rajini]

Served:ravi

Remaining Queue:[ramesh, rajini]

Served:ramesh

Remaining Queue:[rajini]

Served:rajini

Remaining Queue:[]

1. **Task Manager**:
   * Queue of tasks (String values).
   * Add tasks, peek at the next task, and poll completed tasks.

Code; **package** MAp\_collections;

**import** java.util.LinkedList;

**import** java.util.Queue;

**public** **class** Task\_Manager\_Queue {

**public** **static** **void** main(String[] args) {

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

tasks.add("work Assigned");

tasks.add("work In Progress");

tasks.add("work still In Progress ");

tasks.add("work completed");

tasks.add("Additional assigned work completed");

System.***out***.println("Initial tasks queue is:"+tasks);

// Peek at next task

String next = tasks.peek();

System.***out***.println("Next task to process (peek): " + (next != **null** ? next : "None"));

System.***out***.println("Queue after peeking: " + tasks);

// Poll (process) next task

String processed = tasks.poll();

System.***out***.println("Processed (polled): " + (processed != **null** ? processed : "No task to process"));

System.***out***.println("Queue after poll: " + tasks);

}

}

Output; Initial tasks queue is:[work Assigned, work In Progress, work still In Progress , work completed, Additional assigned work completed]

Next task to process (peek): work Assigned

Queue after peeking: [work Assigned, work In Progress, work still In Progress , work completed, Additional assigned work completed]

Processed (polled): work Assigned

Queue after poll: [work In Progress, work still In Progress , work completed, Additional assigned work completed]

1. **Write a method**:
   * That takes a queue of integers and returns a list of even numbers.

Code; **package** MAp\_collections;

**import** java.util.\*;

**public** **class** Evennumber\_queue {

**public** **static** List<Integer> que1(Queue<Integer> q) {

List<Integer> l=**new** ArrayList<Integer>();

**for**(Integer i:q)

{

**if**(i!=**null** && i%2==0)

{

l.add(i);

}

}

**return** l;

}

**public** **static** **void** main(String[] args) {

Queue<Integer> q=**new** LinkedList<Integer>();

q.add(6);

q.add(7);

q.add(3);

q.add(5);

q.add(10);

q.add(9);

System.***out***.println("Original queue: " + q);

List<Integer> evenList =*que1*(q);

System.***out***.println("List of even numbers: " + evenList);

}

}

Output; Original queue: [6, 7, 3, 5, 10, 9]

List of even numbers: [6, 10]

**PriorityQueue**

1. **Hospital Emergency Queue**:
   * Create a class Patient with fields: name and severityLevel (int).
   * Use PriorityQueue<Patient> with a comparator to serve the most critical patients first (highest severityLevel).

Code; **package** Collection\_framework;

**import** java.util.\*;

**class** Patient{

**int** severitylevel;

String name;

Patient(**int** severitylevel,String name){

**this**.severitylevel=severitylevel;

**this**.name=name;

}

**public** String toString() {

**return** "Patient{name='" + name + "', severityLevel=" + severitylevel + "}";

}

}

**public** **class** PriorityQueue\_Hospital {

**public** **static** **void** main(String[] args) {

PriorityQueue<Patient> p=**new** PriorityQueue<Patient>(**new** Comparator<Patient>() {

**public** **int** compare(Patient p1, Patient p2) {

// Descending order: higher severityLevel comes first

**return** Integer.*compare*(p2.severitylevel, p1.severitylevel);

}

});

p.add(**new** Patient(6,"Ravi"));

p.add(**new** Patient(2,"Raji"));

p.add(**new** Patient(9,"Ramesh"));

p.add(**new** Patient(1,"somesh"));

p.add(**new** Patient(5,"Kalpana"));

**while**(!p.isEmpty()) {

Patient nextPatient = p.poll();

System.***out***.println("Serving:"+nextPatient);

}

}

}

Output; Serving:Patient{name='Ramesh', severityLevel=9}

Serving:Patient{name='Ravi', severityLevel=6}

Serving:Patient{name='Kalpana', severityLevel=5}

Serving:Patient{name='Raji', severityLevel=2}

Serving:Patient{name='somesh', severityLevel=1}

1. **Print Jobs Priority**:
   * Add different print jobs (String) with priority levels.
   * Use PriorityQueue to simulate serving high-priority jobs before others.

Code; **package** Collection\_framework;

**import** java.util.\*;

**class** jobs {

String job\_name;

**int** prioritylevel;

jobs(String job\_name,**int** prioritylevel){

**this**.job\_name=job\_name;

**this**.prioritylevel=prioritylevel;

}

**public** String toString() {

**return** "Jobs{job\_name='"+job\_name+"',priority="+prioritylevel + "}";

}

}

**public** **class** Jobs\_Priority\_Queue {

**public** **static** **void** main(String[] args) {

PriorityQueue<jobs> j=**new** PriorityQueue<jobs>(**new** Comparator<jobs>() {

**public** **int** compare(jobs j1,jobs j2) {

**return** Integer.*compare*(j2.prioritylevel, j1.prioritylevel);

}

});

j.add(**new** jobs("Developer",3));

j.add(**new** jobs("Tester",4));

j.add(**new** jobs("Teamlead",5));

j.add(**new** jobs("Manager",1));

j.add(**new** jobs("General manager",2));

**while**(!j.isEmpty()) {

jobs nextjob=j.poll();

System.***out***.println("Priority job served as:"+nextjob);

}

}

}

Output; Priority job served as:Jobs{job\_name='Teamlead',priority=5}

Priority job served as:Jobs{job\_name='Tester',priority=4}

Priority job served as:Jobs{job\_name='Developer',priority=3}

Priority job served as:Jobs{job\_name='General manager',priority=2}

Priority job served as:Jobs{job\_name='Manager',priority=1}

1. **Write a method**:
   * To merge two PriorityQueue<Integer> and return a sorted merged queue.

Code; **package** Collection\_framework;

**import** java.util.\*;

**public** **class** Merge\_two\_priorityQueues {

**public** **static** PriorityQueue<Integer> mergePriorityQueues(PriorityQueue<Integer> pq1, PriorityQueue<Integer> pq2) {

PriorityQueue<Integer> mergedQueue = **new** PriorityQueue<>(pq1);

mergedQueue.addAll(pq2);

**return** mergedQueue;

}

**public** **static** **void** main(String[] args) {

PriorityQueue<Integer> pq1=**new** PriorityQueue<Integer>();

PriorityQueue<Integer> pq2=**new** PriorityQueue<Integer>();

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Enter size of first queue:");

**int** n=scan.nextInt();

System.***out***.println("Enter "+n+" elements for first queue:");

**if**(n<=0) {

System.***out***.println("Enter a positive number greater than 0: and try again.");

**return**;

}

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

**int** q=scan.nextInt();

pq1.add(q);

}

System.***out***.println("Priorityqueue 1 is: "+pq1);

System.***out***.println("Enter size of second queue:");

**int** p=scan.nextInt();

System.***out***.println("Enter "+p+" elements for second queue:");

**if**(p<=0) {

System.***out***.println("Enter a positive number greater than 0: and try again.");

**return**;

}

**for**(**int** i=0;i<p;i++) {

**int** r=scan.nextInt();

pq2.add(r);

}

System.***out***.println("Priorityqueue 2 is :"+pq2);

PriorityQueue<Integer> mergedQueue=*mergePriorityQueues*(pq1, pq2);

System.***out***.println("Merged priorityqueue is:"+mergedQueue);

}

}

Output; Enter size of first queue:

4

Enter 4 elements for first queue:

23 67 34 56

Priorityqueue 1 is: [23, 56, 34, 67]

Enter size of second queue:

3

Enter 3 elements for second queue:

2 6 5

Priorityqueue 2 is :[2, 6, 5]

Merged priorityqueue is:[2, 23, 5, 67, 56, 34, 6]

**Deque**

1. **Palindrome Checker**:
   * Input a string and check if it is a palindrome using a Deque<Character>.

Code; **package** Collection\_framework;

**import** java.util.\*;

**public** **class** Deque\_palindrome\_checker {

**public** **static** **void** main(String[] args) {

Deque<Character> d=**new** ArrayDeque<Character>() ;

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Enter a string:");

String s=scan.nextLine();

**for**(**int** i=0; i<s.length();i++) {

**if** (Character.*isLetterOrDigit*(s.charAt(i))) {

d.add(s.charAt(i));

}

}

**boolean** isPalindrome = **true**;

**while** (d.size() > 1) {

**if** (!d.pollFirst().equals(d.pollLast())) {

isPalindrome = **false**;

**break**;

}

}

System.***out***.println("Entered string is :"+s);

**if**(isPalindrome) {

System.***out***.println("Entered string is palindrome.");

}**else** {

System.***out***.println("Entered string is not a palindrome.");

}

}

}

Output; Enter a string:

madam

Entered string is :madam

Entered string is palindrome.

1. **Double-ended Order System**:
   * Add items from front and rear.
   * Remove items from both ends.
   * Display contents of the deque after each operation.

Code; **package** Collection\_framework;

**import** java.util.\*;

**public** **class** Double\_ended\_order {

**public** **static** **void** main(String[] args) {

Deque<Integer> dq=**new** ArrayDeque<Integer>() ;

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Enter size of deque:");

**int** s=scan.nextInt();

System.***out***.println("Enter "+s+" elements :");

**if**(s<=0) {

System.***out***.println("Enter a positive number greater than 0 :please try again.");

**return**;

}

**for**(**int** i=0;i<s;i++) {

dq.add(scan.nextInt());

}

System.***out***.println("Entered deque is:"+dq);

System.***out***.println("Enter number to add first:");

dq.addFirst(scan.nextInt());

System.***out***.println("Enter number to add last:");

dq.addLast(scan.nextInt());

System.***out***.println("Deque after adding numbers at first and last is:"+dq);

dq.removeFirst();

dq.removeLast();

System.***out***.println("Deque after removing numbers at first and last is:"+dq);

scan.close();

}

}

Output; Enter size of deque:

5

Enter 5 elements :

2 8 9 5 7

Entered deque is:[2, 8, 9, 5, 7]

Enter number to add first:

1

Enter number to add last:

6

Deque after adding numbers at first and last is:[1, 2, 8, 9, 5, 7, 6]

Deque after removing numbers at first and last is:[2, 8, 9, 5, 7]

1. **Browser History Simulation**:
   * Implement browser back and forward navigation using two deques.

Code; **package** Collection\_framework;

**import** java.util.\*;

**public** **class** Browser\_History\_simulation {

**private** Deque<String> backHistory = **new** ArrayDeque<>();

**private** Deque<String> forwardHistory = **new** ArrayDeque<>();

**private** String currentPage;

**public** Browser\_History\_simulation(String homepage) {

currentPage = homepage;

}

**public** **void** visit(String url) {

backHistory.push(currentPage);

currentPage = url;

forwardHistory.clear();

}

**public** **void** back() {

**if** (!backHistory.isEmpty()) {

forwardHistory.push(currentPage);

currentPage = backHistory.pop();

} **else** {

System.***out***.println("No pages in back history.");

}

}

**public** **void** forward() {

**if** (!forwardHistory.isEmpty()) {

backHistory.push(currentPage);

currentPage = forwardHistory.pop();

} **else** {

System.***out***.println("No pages in forward history.");

}

}

**public** **void** printStatus() {

System.***out***.println("Back history: " + backHistory);

System.***out***.println("Current page: " + currentPage);

System.***out***.println("Forward history: " + forwardHistory);

}

**public** **static** **void** main(String[] args) {

Browser\_History\_simulation bh = **new** Browser\_History\_simulation("google.com");

bh.visit("youtube.com");

bh.visit("facebook.com");

bh.printStatus();

bh.back();

bh.printStatus();

bh.forward();

bh.printStatus();

bh.visit("stackoverflow.com");

bh.printStatus();

}

}

Output; Back history: [youtube.com, google.com]

Current page: facebook.com

Forward history: []

Back history: [google.com]

Current page: youtube.com

Forward history: [facebook.com]

Back history: [youtube.com, google.com]

Current page: facebook.com

Forward history: []

Back history: [facebook.com, youtube.com, google.com]

Current page: stackoverflow.com

Forward history: []