Collection coding questions:

1.Write a Java program to create a new array list, add some colors (string) and print out the collection.

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

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

colors.add("Red");

colors.add("Green");

colors.add("Blue");

colors.add("Yellow");

System.out.println("Colors in the collection:");

for (String color : colors) {

System.out.println(color);

}

}

}

O/P:

Colors in the collection:

Red

Green

Blue

Yellow

2.Write a Java program to insert an element into the array list at the first position.

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

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

colors.add("Red");

colors.add("Green");

colors.add("Blue");

colors.add("Yellow");

colors.add(0, "Orange");

System.out.println("Colors in the collection after inserting at the first position:");

for (String color : colors) {

System.out.println(color);

}

}

}

O/P:

Colors in the collection after inserting at the first position:

Orange

Red

Green

Blue

Yellow

3.Write a Java program to retrieve an element (at a specified index) from a given array list.

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

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

colors.add("Red");

colors.add("Green");

colors.add("Blue");

colors.add("Yellow");

int index = 2;

String element = colors.get(index);

System.out.println("Element at index " + index + ": " + element);

}

}

O/P:

Element at index 2: Blue

4.Write a Java program to sort a given array list.

import java.util.ArrayList;

import java.util.Collections;

public class Main {

public static void main(String[] args) {

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

fruits.add("Apple");

fruits.add("Orange");

fruits.add("Banana");

fruits.add("Grapes");

Collections.sort(fruits);

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

for (String fruit : fruits) {

System.out.println(fruit);

}

}

}

O/P:

Sorted fruits:

Apple

Banana

Grapes

Orange

5.Write a Java program to reverse elements in a array list

import java.util.ArrayList;

import java.util.Collections;

public class Main {

public static void main(String[] args) {

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

colors.add("Red");

colors.add("Green");

colors.add("Blue");

colors.add("Yellow");

// Reverse the ArrayList

Collections.reverse(colors);

// Print out the reversed ArrayList

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

for (String color : colors) {

System.out.println(color);

}

}

}

O/P:Reversed colors:

Yellow

Blue

Green

Red

6.Write a Java program of swap two elements in an array list.

import java.util.ArrayList;

import java.util.Collections;

public class Main {

public static void main(String[] args) {

// Create a new ArrayList to store integers

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

// Add some numbers to the ArrayList

numbers.add(1);

numbers.add(2);

numbers.add(3);

numbers.add(4);

numbers.add(5);

// Print the ArrayList before swapping

System.out.println("Before swapping:");

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

// Swap elements at index 1 and index 3

int index1 = 1;

int index2 = 3;

Collections.swap(numbers, index1, index2);

// Print the ArrayList after swapping

System.out.println("After swapping:");

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

}

}

O/P:

Before swapping:

Numbers: [1, 2, 3, 4, 5]

After swapping:

Numbers: [1, 4, 3, 2, 5]

7.Write a Java program to print all the elements of a ArrayList using the position of the elements

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

// Create a new ArrayList to store integers

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

// Add some numbers to the ArrayList

numbers.add(10);

numbers.add(20);

numbers.add(30);

numbers.add(40);

numbers.add(50);

// Print all elements of the ArrayList along with their positions

System.out.println("Elements of the ArrayList with their positions:");

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

System.out.println("Position " + i + ": " + numbers.get(i));

}

}

}

O/P:

Elements of the ArrayList with their positions:

Position 0: 10

Position 1: 20

Position 2: 30

Position 3: 40

Position 4: 50

8.Write a Java program to append the specified element to the end of a linked list.

import java.util.LinkedList;

public class Main {

public static void main(String[] args) {

// Create a new LinkedList to store integers

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

// Add some numbers to the LinkedList

numbers.add(10);

numbers.add(20);

numbers.add(30);

// Print the LinkedList before appending

System.out.println("LinkedList before appending: " + numbers);

// Append a specified element to the end of the LinkedList

int elementToAppend = 40;

numbers.addLast(elementToAppend);

// Print the LinkedList after appending

System.out.println("LinkedList after appending: " + numbers);

}

}

O/P:

LinkedList before appending: [10, 20, 30]

LinkedList after appending: [10, 20, 30, 40]

9.Write a Java program to insert the specified element at the specified position in the linked list.

import java.util.LinkedList;

public class Main {

public static void main(String[] args) {

// Create a new LinkedList to store integers

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

// Add some numbers to the LinkedList

numbers.add(10);

numbers.add(20);

numbers.add(30);

// Print the LinkedList before inserting

System.out.println("LinkedList before inserting: " + numbers);

// Insert a specified element (15) at a specified position (index 1)

int elementToInsert = 15;

int position = 1;

numbers.add(position, elementToInsert);

// Print the LinkedList after inserting

System.out.println("LinkedList after inserting: " + numbers);

}

}

O/P:

LinkedList before inserting: [10, 20, 30]

LinkedList after inserting: [10, 15, 20, 30]

10.Write a Java program to insert elements into the linked list at the first and last position.

import java.util.LinkedList;

public class Main {

public static void main(String[] args) {

// Create a new LinkedList to store integers

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

// Add some numbers to the LinkedList

numbers.add(20);

numbers.add(30);

// Print the LinkedList before insertion

System.out.println("LinkedList before insertion: " + numbers);

// Insert element at the first position

numbers.addFirst(10);

// Insert element at the last position

numbers.addLast(40);

// Print the LinkedList after insertion

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

}

}

O/P:

LinkedList before insertion: [20, 30]

LinkedList after insertion: [10, 20, 30, 40]

11.Write a Java program to display the elements and their positions in a linked list

import java.util.LinkedList;

public class Main {

public static void main(String[] args) {

// Create a new LinkedList to store integers

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

// Add some numbers to the LinkedList

numbers.add(10);

numbers.add(20);

numbers.add(30);

numbers.add(40);

numbers.add(50);

// Display the elements and their positions in the LinkedList

System.out.println("Elements of the LinkedList with their positions:");

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

System.out.println("Position " + i + ": " + numbers.get(i));

}

}

}

O/P:

Elements of the LinkedList with their positions:

Position 0: 10

Position 1: 20

Position 2: 30

Position 3: 40

Position 4: 50

12.Write a Java program to check if a particular element exists in a linked list

import java.util.LinkedList;

public class Main {

public static void main(String[] args) {

// Create a new LinkedList to store integers

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

// Add some numbers to the LinkedList

numbers.add(10);

numbers.add(20);

numbers.add(30);

numbers.add(40);

numbers.add(50);

// Element to check

int elementToCheck = 30;

// Check if the element exists in the LinkedList

boolean exists = numbers.contains(elementToCheck);

// Print the result

if (exists) {

System.out.println("The element " + elementToCheck + " exists in the LinkedList.");

} else {

System.out.println("The element " + elementToCheck + " does not exist in the LinkedList.");

}

}

}

O'P:

The element 30 exists in the LinkedList.

13.Write a Java program to compare two linked lists.

import java.util.LinkedList;

public class Main {

public static void main(String[] args) {

// Create the first LinkedList

LinkedList<Integer> list1 = new LinkedList<>();

list1.add(10);

list1.add(20);

list1.add(30);

// Create the second LinkedList

LinkedList<Integer> list2 = new LinkedList<>();

list2.add(10);

list2.add(20);

list2.add(30);

// Compare the two LinkedLists

boolean isEqual = list1.equals(list2);

// Print the result

if (isEqual) {

System.out.println("The two linked lists are equal.");

} else {

System.out.println("The two linked lists are not equal.");

}

}

}

O/P:

The two linked lists are equal.

14.Write a Java program to replace an element in a linked list.

import java.util.LinkedList;

public class Main {

public static void main(String[] args) {

// Create a LinkedList

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

// Add some elements to the LinkedList

linkedList.add("Apple");

linkedList.add("Banana");

linkedList.add("Orange");

linkedList.add("Grapes");

// Print the LinkedList before replacement

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

// Replace an element at a specified index

int indexToReplace = 1; // Index of "Banana"

String replacement = "Mango";

linkedList.set(indexToReplace, replacement);

// Print the LinkedList after replacement

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

}

}

O/P:

LinkedList before replacement: [Apple, Banana, Orange, Grapes]

LinkedList after replacement: [Apple, Mango, Orange, Grapes]

15.Write a Java program to iterate through all elements in a hash list.

import java.util.HashSet;

import java.util.Iterator;

public class Main {

public static void main(String[] args) {

// Create a HashSet to store integers

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

// Add some integers to the HashSet

hashSet.add(10);

hashSet.add(20);

hashSet.add(30);

hashSet.add(40);

hashSet.add(50);

// Iterate through all elements in the HashSet

System.out.println("Elements in the HashSet:");

Iterator<Integer> iterator = hashSet.iterator();

while (iterator.hasNext()) {

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

}

}

}

o/p:

Elements in the HashSet:

40

10

50

20

30

16.Write a Java program to empty an hash set.

import java.util.HashSet;

public class Main {

public static void main(String[] args) {

// Create a HashSet to store integers

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

// Add some integers to the HashSet

hashSet.add(10);

hashSet.add(20);

hashSet.add(30);

hashSet.add(40);

hashSet.add(50);

// Print the HashSet before emptying

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

// Empty the HashSet

hashSet.clear();

// Print the HashSet after emptying

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

}

}

o/p:

HashSet before emptying: [40, 10, 50, 20, 30]

HashSet after emptying: []

17.Write a Java program to convert a hash set to an array.

import java.util.HashSet;

public class Main {

public static void main(String[] args) {

// Create a HashSet to store integers

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

// Add some integers to the HashSet

hashSet.add(10);

hashSet.add(20);

hashSet.add(30);

hashSet.add(40);

hashSet.add(50);

// Convert the HashSet to an array

Integer[] array = hashSet.toArray(new Integer[hashSet.size()]);

// Print the array

System.out.println("Array from HashSet:");

for (Integer num : array) {

System.out.println(num);

}

}

}

o/p:

Array from HashSet:

40

10

50

20

30

18.Write a Java program to compare two sets and retain elements which are same on both sets.

import java.util.HashSet;

import java.util.Set;

public class Main {

public static void main(String[] args) {

// Create two sets

Set<Integer> set1 = new HashSet<>();

Set<Integer> set2 = new HashSet<>();

// Add elements to the first set

set1.add(10);

set1.add(20);

set1.add(30);

set1.add(40);

// Add elements to the second set

set2.add(30);

set2.add(40);

set2.add(50);

set2.add(60);

// Retain only the common elements between the two sets

set1.retainAll(set2);

// Print the elements in the first set after retaining common elements

System.out.println("Elements retained in set1 after comparison: " + set1);

}

}

o/p:

Elements retained in set1 after comparison: [40, 30]

19.Write a Java program to create a new tree set, add some colors (string) and print out the tree set.

import java.util.TreeSet;

public class Main {

public static void main(String[] args) {

// Create a new TreeSet to store colors

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

// Add some colors to the TreeSet

treeSet.add("Red");

treeSet.add("Green");

treeSet.add("Blue");

treeSet.add("Yellow");

// Print out the TreeSet

System.out.println("Colors in the TreeSet:");

for (String color : treeSet) {

System.out.println(color);

}

}

}

o/p:

Colors in the TreeSet:

Blue

Green

Red

Yellow

20.Write a Java program to find the numbers less than 7 in a tree set.

import java.util.TreeSet;

public class Main {

public static void main(String[] args) {

// Create a TreeSet to store integers

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

// Add some integers to the TreeSet

treeSet.add(5);

treeSet.add(8);

treeSet.add(3);

treeSet.add(12);

treeSet.add(1);

treeSet.add(6);

treeSet.add(9);

// Find numbers less than 7 in the TreeSet

System.out.println("Numbers less than 7 in the TreeSet:");

for (Integer num : treeSet) {

if (num < 7) {

System.out.println(num);

}

}

}

}

o/p:

Numbers less than 7 in the TreeSet:

1

3

5

6

21.Write a Java program to remove all the elements from a priority queue.

import java.util.PriorityQueue;

public class Main {

public static void main(String[] args) {

// Create a PriorityQueue to store integers

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

// Add some integers to the PriorityQueue

priorityQueue.add(10);

priorityQueue.add(20);

priorityQueue.add(30);

priorityQueue.add(40);

priorityQueue.add(50);

// Print the PriorityQueue before removal

System.out.println("PriorityQueue before removal: " + priorityQueue);

// Remove all elements from the PriorityQueue

priorityQueue.clear();

// Print the PriorityQueue after removal

System.out.println("PriorityQueue after removal: " + priorityQueue);

}

}

o/p:

PriorityQueue before removal: [10, 20, 30, 40, 50]

PriorityQueue after removal: []

22.Write a Java program to count the number of key-value (size) mappings in a map

import java.util.HashMap;

import java.util.Map;

public class Main {

public static void main(String[] args) {

// Create a HashMap to store key-value pairs

Map<String, Integer> map = new HashMap<>();

// Add some key-value mappings to the map

map.put("A", 1);

map.put("B", 2);

map.put("C", 3);

map.put("D", 4);

map.put("E", 5);

// Count the number of key-value mappings in the map

int size = map.size();

// Print the number of key-value mappings

System.out.println("Number of key-value mappings in the map: " + size);

}

}

o/p:

Number of key-value mappings in the map: 5

23. Write a Java program to convert a priority queue to an array containing all of the elements of the queue

import java.util.PriorityQueue;

import java.util.Arrays;

public class Main {

public static void main(String[] args) {

// Create a PriorityQueue to store integers

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

// Add some integers to the PriorityQueue

priorityQueue.add(10);

priorityQueue.add(20);

priorityQueue.add(30);

priorityQueue.add(40);

priorityQueue.add(50);

// Convert the PriorityQueue to an array

Integer[] array = priorityQueue.toArray(new Integer[priorityQueue.size()]);

// Print the array

System.out.println("Array from PriorityQueue:");

System.out.println(Arrays.toString(array));

}

}

24. Write a Java program to check whether a map contains key-value mappings (empty) or not

import java.util.HashMap;

import java.util.Map;

public class Main {

public static void main(String[] args) {

// Create a HashMap to store key-value pairs

Map<String, Integer> map = new HashMap<>();

// Add some key-value mappings to the map

map.put("A", 1);

map.put("B", 2);

map.put("C", 3);

map.put("D", 4);

map.put("E", 5);

// Check if the map is empty

boolean isEmpty = map.isEmpty();

// Print the result

if (isEmpty) {

System.out.println("The map is empty.");

} else {

System.out.println("The map contains key-value mappings.");

}

}

}

o/p:

The map contains key-value mappings.

25. Write a Java program to get the value of a specified key in a map

import java.util.HashMap;

import java.util.Map;

public class Main {

public static void main(String[] args) {

// Create a HashMap to store key-value pairs

Map<String, Integer> map = new HashMap<>();

// Add some key-value mappings to the map

map.put("A", 1);

map.put("B", 2);

map.put("C", 3);

map.put("D", 4);

map.put("E", 5);

// Specify the key to get the value

String key = "C";

// Get the value corresponding to the specified key

Integer value = map.get(key);

// Print the value

if (value != null) {

System.out.println("Value of key '" + key + "': " + value);

} else {

System.out.println("Key '" + key + "' does not exist in the map.");

}

}

}

o/p:

Value of key 'C': 3

Logical Coding Question:

1)

\*

\* \*

\* \*

\* \*

\*\*\*\*\*\*\*\*\*

public class Pattern {

public static void main(String[] args) {

int rows = 5;

// Outer loop for number of rows

for (int i = 1; i <= rows; i++) {

// Inner loop 1 for spaces

for (int j = i; j < rows; j++) {

System.out.print(" ");

}

// Inner loop 2 for stars

for (int k = 1; k <= (2 \* i - 1); k++) {

if (k == 1 || k == (2 \* i - 1)) {

System.out.print("\*");

} else {

System.out.print(" ");

}

}

// Move to the next line after each row

System.out.println();

}

}

}

2)

\*\*\*\*\*\*\*\*\*

\* \*

\* \*

\* \*

\*

public class Pattern {

public static void main(String[] args) {

int rows = 5;

// Outer loop for number of rows

for (int i = 1; i <= rows; i++) {

// Inner loop 1 for spaces

for (int j = 1; j < i; j++) {

System.out.print(" ");

}

// Inner loop 2 for stars

for (int k = 1; k <= (2 \* (rows - i + 1) - 1); k++) {

if (k == 1 || k == (2 \* (rows - i + 1) - 1)) {

System.out.print("\*");

} else {

System.out.print(" ");

}

}

// Move to the next line after each row

System.out.println();

}

}

}

\*\*\*\*\*\*\*\*\*

\* \*

\* \*

\* \*

\*

3)

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

public class Pattern {

public static void main(String[] args) {

int rows = 5;

int count = 1;

// Outer loop for number of rows

for (int i = 1; i <= rows; i++) {

// Inner loop for printing numbers

for (int j = 1; j <= i; j++) {

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

count++;

}

// Move to the next line after each row

System.out.println();

}

}

}

4)

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

public class Pattern {

public static void main(String[] args) {

int rows = 5;

// Outer loop for number of rows

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

// Inner loop 1 for spaces

for (int j = i; j < rows - 1; j++) {

System.out.print(" ");

}

// Inner loop 2 for printing numbers

int number = 1;

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

System.out.printf("%4d", number);

number = number \* (i - k) / (k + 1);

}

// Move to the next line after each row

System.out.println();

}

}

}

25)

1 2 3 4 5 6 7

2 3 4 5 6 7 1

3 4 5 6 7 1 2

4 5 6 7 1 2 3

5 6 7 1 2 3 4

6 7 1 2 3 4 5

7 1 2 3 4 5 6

public class NumberPattern {

public static void main(String[] args) {

int n = 7; // Number of rows

int[][] pattern = new int[n][n];

// Fill the pattern

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

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

pattern[i][j] = (i + j + 1) % n + 1;

}

}

// Print the pattern

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

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

System.out.print(pattern[i][j] + " ");

}

System.out.println();

}

}

}