Java Coding

1. Find all character count in a string.

String input = "ABCABCDABCABCEABCABCFPQRPQRGSTUSTUH";

char chr[] = input.toCharArray();

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

for(Character ch : chr) {

if(map.containsKey(ch)) {

map.put(ch, map.get(ch)+1);

}

else {

map.put(ch, 1);

}

}

for(Character c : map.keySet()) {

if(map.get(c) > 0) {

System.out.print(c + "-" + map.get(c) + " ");

}

}

// Output: A-6 B-6 C-6 D-1 E-1 F-1 G-1 H-1 P-2 Q-2 R-2 S-2 T-2 U-2

1. Find duplicate character in a string.

String input = "ABCABCDABCABCEABCABCFPQRPQRGSTUSTUH";

char chr[] = input.toCharArray();

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

for(Character ch : chr) {

if(map.containsKey(ch)) {

map.put(ch, map.get(ch)+1);

}

else {

map.put(ch, 1);

}

}

for(Character c : map.keySet()) {

if(map.get(c) > 1) {

System.out.print(c + "-" + map.get(c) + " ");

}

}

// A-6 B-6 C-6 P-2 Q-2 R-2 S-2 T-2 U-2

1. First non-repeated character in string.

String input = "ABCABCDABCABCEABCABCFPQRPQRGSTUSTUH";

char chr[] = input.toCharArray();

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

for(Character ch : chr) {

if(map.containsKey(ch)) {

map.put(ch, map.get(ch)+1);

}

else {

map.put(ch, 1);

}

}

for(Character c : map.keySet()) {

if(map.get(c) == 1) {

System.out.println("First non-repeated character is: " + c);

break;

}

}

// First non-repeated character is: D

1. Sort an Array in ascending and descending order.

String input[] = {"java", "selenium", "cucumber", "rest", "soap"};

System.out.println("Un-sorted array: " + Arrays.toString(input));

Arrays.sort(input);

System.out.println("Sorted in ascending order: " + Arrays.toString(input));

Arrays.sort(input, Collections.reverseOrder());

System.out.println("Sorted in reverse order: " + Arrays.toString(input));

// Un-sorted array: [java, selenium, cucumber, rest, soap]

// Sorted in ascending order: [cucumber, java, rest, selenium, soap]

// Sorted in reverse order: [soap, selenium, rest, java, cucumber]

1. Sort a list in ascending and descending order.

String input[] = {"java", "selenium", "cucumber", "rest", "soap"};

List<String> list = new ArrayList<>(Arrays.asList(input));

System.out.println("Un-sorted list: " + list);

Collections.sort(list);

System.out.println("Sorted list in natural order: " + list);

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

System.out.println("Sorted list in descending order: " + list);

// Un-sorted list: [java, selenium, cucumber, rest, soap]

// Sorted list in natural order: [cucumber, java, rest, selenium, soap]

// Sorted list in descending order: [soap, selenium, rest, java, cucumber]

1. Sort a set in ascending and descending order

//Step 1: Set cannot be sorted by Collections.sort() method.

//Step 2: We need to convert set to a list.

//Step 3: Linked HashSet maintain insertion order

//Step 4: Store elements in LinkedHashSet

//Step 5: Convert set object to ArrayList using ArrayList<String> list = new ArrayList<>(set);

//Step 6: Sort the list in natural order using Collections.sort(list).

//Step 7: Reverse the list using Collections.sort(list, Collections.reverseOrder());

//Step 8: Convert the list to set using set = new LinkedHashSet<>(list);

Set<String> set = new LinkedHashSet<>();

set.add("java");

set.add("selenium");

set.add("cucumber");

set.add("rest");

set.add("soap");

System.out.println("Un-sorted set: " + set);

ArrayList<String> list = new ArrayList<>(set);

System.out.println("Un-sorted list: " + list);

Collections.sort(list);

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

set = new LinkedHashSet<>(list);

System.out.println("Sorted set in ascending order: " + set);

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

System.out.println("Sorted list in reverse order: " + list);

set = new LinkedHashSet<>(list);

System.out.println("Sorted set in reverse order: " + set);

/\*

\* Un-sorted set: [java, selenium, cucumber, rest, soap]

Un-sorted list: [java, selenium, cucumber, rest, soap]

Sorted list: [cucumber, java, rest, selenium, soap]

Sorted set in ascending order: [cucumber, java, rest, selenium, soap]

Sorted list in reverse order: [soap, selenium, rest, java, cucumber]

Sorted set in reverse order: [soap, selenium, rest, java, cucumber]

\*/

1. Sort a map using keyset.

// HashMap does not maintain insertion order

// TreeMap is auto sorted in natural/ ascending order by Key

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

map.put(30, "cucumber");

map.put(11, "java");

map.put(404, "rest");

map.put(55, "soap");

map.put(212, "selenium");

System.out.println("Map auto sorted by key: " + map);

// Map auto sorted by key: {11=java, 30=cucumber, 55=soap, 212=selenium, 404=rest}

1. Sort a map using custom objects.

public class Education {

private int id;

private String subject;

public Education(int id, String subject) {

this.id = id;

this.subject = subject;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getSubject() {

return subject;

}

public void setSubject(String subject) {

this.subject = subject;

}

@Override

public String toString() {

return "Education [id=" + id + ", subject=" + subject + "]";

}

}

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

public class Education\_Sorting\_by\_Subject implements Comparator<Education> {

// @Override

// public int compare(Education o1, Education o2) {

// // natural/ ascending order

// return o1.getSubject().compareTo(o2.getSubject());

// }

@Override

public int compare(Education o1, Education o2) {

// descending order

return o2.getSubject().compareTo(o1.getSubject());

}

}

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

public class Education\_Sorting\_by\_Id implements Comparator<Education>{

// @Override

// public int compare(Education o1, Education o2) {

// // natural/ ascending order

// return o1.getId() - o2.getId();

// }

@Override

public int compare(Education o1, Education o2) {

// descending order

return o2.getId() - o1.getId();

}

}

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

public class Education\_Test\_Subject\_Sorting {

public static void main(String[] args) {

ArrayList<Education> list = new ArrayList<Education>();

list.add(new Education(30, "cucumber"));

list.add(new Education(212, "selenium"));

list.add(new Education(404, "rest"));

list.add(new Education(55, "soap"));

list.add(new Education(11, "java"));

System.out.println("Un-sorted list by subject: ");

Iterator<Education> itr = list.iterator();

while(itr.hasNext()) {

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

}

Collections.sort(list, new Education\_Sorting\_by\_Subject());

System.out.println("Sorted list by subject: ");

itr = list.iterator();

while(itr.hasNext()) {

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

}

}

}

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

public class Education\_Test\_ID\_Sorting {

public static void main(String[] args) {

ArrayList<Education> list = new ArrayList<Education>();

list.add(new Education(30, "cucumber"));

list.add(new Education(212, "selenium"));

list.add(new Education(404, "rest"));

list.add(new Education(55, "soap"));

list.add(new Education(11, "java"));

System.out.println("Un-sorted list: ");

Iterator<Education> itr = list.iterator();

while(itr.hasNext()) {

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

}

Collections.sort(list, new Education\_Sorting\_by\_Id());

System.out.println("Sorted list natural order: ");

itr = list.iterator();

while(itr.hasNext()) {

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

}

}

}

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

1. Reverse an integer.

int number = 12345;

int reminder;

int reverse = 0;

while(number > 0) {

reminder = number % 10;

reverse = reverse \* 10 + reminder;

number = number / 10;

}

System.out.println(reverse);

1. Palindrome number.

int number = 5225;

int reminder;

int reverse = 0;

int temp = number;

while(number > 0) {

reminder = number % 10;

reverse = reverse \* 10 + reminder;

number = number / 10;

}

if(temp == reverse) {

System.out.println(temp + " is a Palindrome number");

}

else {

System.out.println(temp + " is not a Palindrome number");

}

1. Armstrong number.

int number = 370;

int reminder;

int cube = 0;

int temp = number;

while(number>0) {

reminder = number % 10;

cube = cube + (reminder \* reminder \* reminder);

number = number/10;

}

if(temp == cube) {

System.out.println(cube + " is a Armstrong number");

}

else {

System.out.println(cube + " is not a Armstrong number");

}

1. Prime number.

public static void main(String[] args) {

for(int i=0; i<=100; i++) {

checkPrimeNumber(i);

}

}

public static void checkPrimeNumber(int number) {

int flag = 0;

if(number <=1) {

System.out.println(number + " is not a Prime number");

}

else {

for(int i=2; i<=number/2; i++) {

if(number%i==0) {

System.out.println(number + " is not a Prime number");

flag = 1;

break;

}

}

if(flag == 0) {

System.out.println(number + " is a Prime number");

}

}

}

1. Factorial Number.

int number = 7;

int fact = 1;

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

fact = fact \* i;

}

System.out.println("Factorial of " + number + " is: "+ fact);

}

1. Fibonacci Series.

int number = 7;

int n1 = 0;

int n2 = 1;

int n3 = 0;

System.out.print(n1 + " " + n2);

for(int i=2; i<=number; i++) {

n3 = n1 + n2;

System.out.print(" " + n3);

n1 = n2;

n2 = n3;

}

1. Highest and lowest number in array.

int number[] = {1,9,2,8,3,7,4,6,5};

int max = number[0];

int min = number[0];

for(int i=1; i<number.length; i++) {

if(number[i] > max) {

max = number[i];

}

else if(number[i] < min){

min = number[i];

}

}

System.out.println("Lergest number is: " + max);

System.out.println("Lowest number is: " + min);

}

1. Swap number.

int a = 10;

int b = 20;

a = a+b;

b = a-b;

a = a-b;

System.out.println("New value: a = " + a + ", b = " +b);

int m = 30;

int n = 40;

int temp=m;

m=n;

n=temp;

System.out.println("New value: m = " + m + ", n = " +n);

1. Leap year.

int year = 2021;

if(year%4==0 && (year%400==0 || year%100 !=0)) {

System.out.println(year + " is a leap year");

}

else {

System.out.println(year + " is not a leap year");

}