Programmes

**Prime number**

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

**void** checkPrime(){

System.***out***.println("enter no");

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

**int** i,m=0,flag=0;

m=n/2;

**if**(n==0||n==1)

{

System.***out***.println("is not prime");

}

**else** {

**for**(i=2;i<=m;i++)

{

**if**(n%i==0) {

System.***out***.println("is not prime");

flag=1;

**break**;

}

}

**if**(flag==0) {

System.***out***.println("is prime");

**count digit**

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

**public** **void** cv()

{

System.***out***.println("enter num");

**long** n=scan.nextLong();

**int** count=0;

**while**(n!=0)

{

n=n/10;

count++;

}

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

**Reverse string**

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

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

String num = *sc*.next();

**char**[] a=num.toCharArray();

**for**(**int** i=a.length-1;i>=0;i--) {

System.***out***.print(a[i]);

}

**Reverse string**

String name="prakash";

String reverse="";

**for**(**int** i=name.length()-1;i>=0;i--) {

reverse=reverse+name.charAt(i);

}

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

**First & Last digit**

  int number = 502356997;

        int firstDigit = 0;

        int lastDigit = 0;

        lastDigit = number%10;

        System.out.println("Last digit: "+lastDigit);

        while(number!=0) {

            firstDigit = number%10;

            number /= 10;

        }

        System.out.println("First digit: "+firstDigit);

    }

**Factorial**

1. **int** i,fact=1;
2. **int** number=5;//It is the number to calculate factorial
3. **for**(i=1;i<=number;i++){
4. fact=fact\*i;
5. }
6. System.out.println("Factorial of "+number+" is: "+fact);
7. }

**Fibonaci**

**int** n1=0,n2=1,i,n3,count=10;

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

**for**(i=2;i<count;++i)

{

n3=n1+n2;

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

n1=n2;

n2=n3;

}

**Odd &even array**

1. **int** a[]={1,2,5,6,3,2};
2. System.out.println("Odd Numbers:");
3. **for**(**int** i=0;i<a.length;i++){
4. **if**(a[i]%2!=0){
5. System.out.println(a[i]);
6. }
7. }
8. System.out.println("Even Numbers:");
9. **for**(**int** i=0;i<a.length;i++){
10. **if**(a[i]%2==0){
11. System.out.println(a[i]);
12. }
13. }
14. }

**Ascending order**

1. **int** [] arr = **new** **int** [] {5, 2, 8, 7, 1};
2. **int** temp = 0;
4. //Displaying elements of original array
5. System.out.println("Elements of original array: ");
6. **for** (**int** i = 0; i < arr.length; i++) {
7. System.out.print(arr[i] + " ");
8. }
10. //Sort the array in ascending order
11. **for** (**int** i = 0; i < arr.length; i++) {
12. **for** (**int** j = i+1; j < arr.length; j++) {
13. **if**(arr[i] > arr[j]) {
14. temp = arr[i];
15. arr[i] = arr[j];
16. arr[j] = temp;
17. }
18. }
19. }
21. System.out.println();
23. //Displaying elements of array after sorting
24. System.out.println("Elements of array sorted in ascending order: ");
25. **for** (**int** i = 0; i < arr.length; i++) {
26. System.out.print(arr[i] + " ");
27. }
28. }

**Alphabets a 2 z**

class Main {

public static void main(String[] args) {

char c;

for(c = 'A'; c <= 'Z'; ++c)

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

}

}

**Perfectnumber**

    long n,sum=0;

                 Scanner sc=new Scanner(System.in);

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

                   n=sc.nextLong();

    int i=1;

    while(i<=n/2)

    {

       if(n%i==0)

       {

sum+=i;

       }

      i++;

    }

if(sum==n)

{

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

               }

else

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

}

**Strong number**

**WordCount**

**public** **void** wordCount()

{

String sentence = "hi prakash";

**int** wordCount = 0;

**for**(**int** i = 0; i < sentence.length()-1; i++) {

//Counts all the spaces present in the string

//It doesn't include the first space as it won't be considered as a word

**if**(sentence.charAt(i) == ' ' && Character.*isLetter*(sentence.charAt(i+1)) && (i > 0)) {

wordCount++;

}

}

wordCount++;

System.***out***.println("Total number of words in the given string: " + wordCount);

}

**Armstrongnumber**

int a,arm=0,n,temp;

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

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

             n = Integer.parseInt(in.readLine());

temp=n;

while(n!=0)

{

a=n%10;

arm=arm+(a\*a\*a);

n=n/10;

}

if(arm==temp)

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

else

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

}

**Duplicate element in array**

1. **int** [] arr = **new** **int** [] {1, 2, 3, 4, 2, 7, 8, 8, 3};
2. System.out.println("Duplicate elements in given array: ");
3. //Searches for duplicate element
4. **for**(**int** i = 0; i < arr.length; i++) {
5. **for**(**int** j = i + 1; j < arr.length; j++) {
6. **if**(arr[i] == arr[j])
7. System.out.println(arr[j]);
8. }
9. }

**Smallest number**

1. **public** **class** SmallestElement\_array {
2. **public** **static** **void** main(String[] args) {
4. //Initialize array
5. **int** [] arr = **new** **int** [] {25, 11, 7, 75, 56};
6. //Initialize min with first element of array.
7. **int** min = arr[0];
8. //Loop through the array
9. **for** (**int** i = 0; i < arr.length; i++) {
10. //Compare elements of array with min
11. **if**(arr[i] <min)
12. min = arr[i];
13. }
14. System.out.println("Smallest element present in given array: " + min);
15. }
16. }

**Vowelse in sentencebk**

int count = 0m

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

Scanner sc = new Scanner(System.in);

String sentence = sc.nextLine();

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

char ch = sentence.charAt(i);

if(ch == 'a'|| ch == 'e'|| ch == 'i' ||ch == 'o' ||ch == 'u'||ch == ' '){

count ++;

}

}

System.out.println("Number of vowels in the given sentence is "+count);

}

|  |  |
| --- | --- |
| **Class** | **Description** |
| LocalDate | Represents a date (year, month, day (yyyy-MM-dd)) |
| LocalTime | Represents a time (hour, minute, second and nanoseconds (HH-mm-ss-ns)) |
| LocalDateTime | Represents both a date and a time (yyyy-MM-dd-HH-mm-ss-ns) |
| DateTimeFormatter | Formatter for displaying and parsing date-time objects |

**Example**

public class Main {

public static void main(String[] args) {

LocalDate myObj = LocalDate.now(); // Create a date object

System.out.println(myObj); // Display the current date

}

}

LocalDateTime myObj = LocalDateTime.now();

System.out.println(myObj);

LocalTime myObj = LocalTime.now();

System.out.println(myObj);

public class Main {

public static void main(String[] args) {

LocalDateTime myDateObj = LocalDateTime.now();

System.out.println("Before formatting: " + myDateObj);

DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

String formattedDate = myDateObj.format(myFormatObj);

System.out.println("After formatting: " + formattedDate);

}

}

The output will be:

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Return Type** |
| [charAt()](https://www.w3schools.com/java/ref_string_charat.asp) | Returns the character at the specified index (position) | char |
| [codePointAt()](https://www.w3schools.com/java/ref_string_codepointat.asp) | Returns the Unicode of the character at the specified index | int |
| [codePointBefore()](https://www.w3schools.com/java/ref_string_codepointbefore.asp) | Returns the Unicode of the character before the specified index | int |
| [codePointCount()](https://www.w3schools.com/java/ref_string_codepointcount.asp) | Returns the Unicode in the specified text range of this String | int |
| [compareTo()](https://www.w3schools.com/java/ref_string_compareto.asp) | Compares two strings lexicographically | int |
| [compareToIgnoreCase()](https://www.w3schools.com/java/ref_string_comparetoignorecase.asp) | Compares two strings lexicographically, ignoring case differences | int |
| [concat()](https://www.w3schools.com/java/ref_string_concat.asp) | Appends a string to the end of another string | String |
| [contains()](https://www.w3schools.com/java/ref_string_contains.asp) | Checks whether a string contains a sequence of characters | boolean |
| [contentEquals()](https://www.w3schools.com/java/ref_string_contentequals.asp) | Checks whether a string contains the exact same sequence of characters of the specified CharSequence or StringBuffer | boolean |
| [copyValueOf()](https://www.w3schools.com/java/ref_string_copyvalueof.asp) | Returns a String that represents the characters of the character array | String |
| [endsWith()](https://www.w3schools.com/java/ref_string_endswith.asp) | Checks whether a string ends with the specified character(s) | boolean |
| [equals()](https://www.w3schools.com/java/ref_string_equals.asp) | Compares two strings. Returns true if the strings are equal, and false if not | boolean |
| [equalsIgnoreCase()](https://www.w3schools.com/java/ref_string_equalsignorecase.asp) | Compares two strings, ignoring case considerations | boolean |
| format() | Returns a formatted string using the specified locale, format string, and arguments | String |
| getBytes() | Encodes this String into a sequence of bytes using the named charset, storing the result into a new byte array | byte[] |
| getChars() | Copies characters from a string to an array of chars | void |
| [hashCode()](https://www.w3schools.com/java/ref_string_hashcode.asp) | Returns the hash code of a string | int |
| [indexOf()](https://www.w3schools.com/java/ref_string_indexof.asp) | Returns the position of the first found occurrence of specified characters in a string | int |
| intern() | Returns the canonical representation for the string object | String |
| [isEmpty()](https://www.w3schools.com/java/ref_string_isempty.asp) | Checks whether a string is empty or not | boolean |
| [lastIndexOf()](https://www.w3schools.com/java/ref_string_lastindexof.asp) | Returns the position of the last found occurrence of specified characters in a string | int |
| [length()](https://www.w3schools.com/java/ref_string_length.asp) | Returns the length of a specified string | int |
| matches() | Searches a string for a match against a regular expression, and returns the matches | boolean |
| offsetByCodePoints() | Returns the index within this String that is offset from the given index by codePointOffset code points | int |
| regionMatches() | Tests if two string regions are equal | boolean |
| [replace()](https://www.w3schools.com/java/ref_string_replace.asp) | Searches a string for a specified value, and returns a new string where the specified values are replaced | String |
| replaceFirst() | Replaces the first occurrence of a substring that matches the given regular expression with the given replacement | String |
| replaceAll() | Replaces each substring of this string that matches the given regular expression with the given replacement | String |
| split() | Splits a string into an array of substrings | String[] |
| [startsWith()](https://www.w3schools.com/java/ref_string_startswith.asp) | Checks whether a string starts with specified characters | boolean |
| subSequence() | Returns a new character sequence that is a subsequence of this sequence | CharSequence |
| substring() | Returns a new string which is the substring of a specified string | String |
| toCharArray() | Converts this string to a new character array | char[] |
| [toLowerCase()](https://www.w3schools.com/java/ref_string_tolowercase.asp) | Converts a string to lower case letters | String |
| toString() | Returns the value of a String object | String |
| [toUpperCase()](https://www.w3schools.com/java/ref_string_touppercase.asp) | Converts a string to upper case letters | String |
| [trim()](https://www.w3schools.com/java/ref_string_trim.asp) | Removes whitespace from both ends of a string | String |
| valueOf() | Returns the string representation of the specified value | String |

**String methods**

All String Methods

The String class has a set of built-in methods that you can use on strings.

**Smallest number**

1. **int** [] arr = **new** **int** [] {25, 11, 7, 75, 56};
2. //Initialize min with first element of array.
3. **int** min = arr[0];
4. //Loop through the array
5. **for** (**int** i = 0; i < arr.length; i++) {
6. //Compare elements of array with min
7. **if**(arr[i] <min)
8. min = arr[i];
9. }
10. System.out.println("Smallest element present in given array: " + min);
11. }

**Largest number**

**Duplicate remove**

public class Demo{

   public static void main(String[] args){

      String my\_str = "This is a is sample a sample only.";

      my\_str = Arrays.stream(my\_str.split("\\s+")).distinct().collect(Collectors.joining(" "));

      System.out.println(my\_str);

   }

}

**Remove white space in string**

public class RemovingWhiteSpaces {

   public static void main(String args[]) {

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

      Scanner sc = new Scanner(System.in);

      String str = sc.nextLine();

      System.out.println(str.replaceAll(" ", ""));

   }

}

**Final**

* It is a keyword.
* It is used to apply restrictions on classes, methods and variables.
* It can’t be inherited.
* It can’t be overridden.
* Final methods can’t be inherited by any class.
* It is needed to initialize the final variable when it is being declared.
* Its value, once declared, can’t be changed or re-initialized.

**Finally**

* It is a block.
* It is used to place important code in this block.
* It gets executed irrespective of whether the exception is handled or not.

**Finalize**

* It is a method.
* It is used to perform clean up processing right before the object is collected by garbage collector.

**Missing alphabets**

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

System.***out***.println("Enter value");

String s1=s.nextLine();

s.close();

**for**(**char** c='a'; c<='z'; c++)

{

**if**(s1.toLowerCase().indexOf(c)==-1)

{

System.***out***.print(c);

}

}

**Palindrome**

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

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

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

**int** r,sum=0;

**int** temp=num;

**while**(num>0)

{

r=num%10;

sum=(sum\*10)+r;

num=num/10;

}

**if**(temp==sum)

System.***out***.println("The entered number "+temp+" is a palindrome number ");

**else**

System.***out***.println("The entered number "+temp+" is not a palindrome");

**Power number**

1. **public** **class** PowerOfNumberExample1
2. {
3. //function to find the power of a number
4. **static** **int** power(**int** base, **int** exponent)
5. {
6. **int** power = 1;
7. //increment the value of i after each iteration until the condition becomes false
8. **for** (**int** i = 1; i <= exponent; i++)
9. //calculates power
10. power = power \* base;
11. //returns power
12. **return** power;
13. }
14. //driver code
15. **public** **static** **void** main(String args[])
16. {
17. **int** base, exponent;
18. Scanner sc=**new** Scanner(System.in);
19. System.out.print("Enter the base: ");
20. base=sc.nextInt();
21. System.out.print("Enter the exponent: ");
22. exponent=sc.nextInt();
23. //calling function
24. **int** pow=power(base, exponent);
25. //prints the result
26. System.out.println(base +" to the power " +exponent + " is: "+pow);
27. }
28. }

**Sum of odd number array**

private static Scanner sc;

public static void main(String[] args)

{

int Size, i, OddSum = 0;

sc = new Scanner(System.in);

System.out.print(" Please Enter Number of elements in an array : ");

Size = sc.nextInt();

int [] a = new int[Size];

System.out.print(" Please Enter " + Size + " elements of an Array : ");

for (i = 0; i < Size; i++)

{

a[i] = sc.nextInt();

}

for(i = 0; i < Size; i++)

{

if(a[i] % 2 != 0)

{

OddSum = OddSum + a[i];

}

}

System.out.println("\n The Sum of Odd Numbers in this Array = " + OddSum);

}

}

**Sum of even number array**

private static Scanner sc;

public static void main(String[] args)

{

int number, i, evenSum = 0, oddSum = 0;

sc = new Scanner(System.in);

System.out.print(" Please Enter any Number : ");

number = sc.nextInt();

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

{

if(i % 2 == 0)

{

evenSum = evenSum + i;

}

else

{

oddSum = oddSum + i;

}

}

System.out.println("\n The Sum of Even Numbers upto " + number + " = " + evenSum);

System.out.println("\n The Sum of Odd Numbers upto " + number + " = " + oddSum);

}

}

**Duplicate remove in hashmap**

 static void removeDups(int[] arr, int n)

    {

        HashMap<Integer,

                Boolean> mp = new HashMap<>();

        for (int i = 0; i < n; ++i)

        {

            // Print the element if it is not

            // there in the hash map

            if (mp.get(arr[i]) == null)

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

            // Insert the element in the hash map

            mp.put(arr[i], true);

        }

    }

    // Driver Code

    public static void main(String[] args)

    {

        int[] arr = { 1, 2, 5, 1, 7, 2, 4, 2 };

        int n = arr.length;

        removeDups(arr, n);

    }

}

**Remove first char**

private static Scanner sc;

public static void main(String[] args) {

String delFirstCharStr;

char del\_ch;

sc= new Scanner(System.in);

System.out.print("\nPlease Enter String to Delete First Character = ");

delFirstCharStr = sc.nextLine();

System.out.print("\nEnter the Character to Delete = ");

del\_ch = sc.next().charAt(0);

StringBuilder sb = new StringBuilder(delFirstCharStr);

int i = delFirstCharStr.indexOf(del\_ch);

sb.deleteCharAt(i);

System.out.println("\nThe Final String after Deleting " + del\_ch + " = " + sb);

}

**Count char using hash map**

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

System.***out***.println("enter sentence");

String name=s.nextLine();

**char**[] c=name.toCharArray();

HashMap<Integer, Character> hm=**new** HashMap<Integer,Character>();

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

hm.put(i,c[i]);

}

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

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

**In Java, there are three ways by using which, we can take input from the console.**

1. BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(System.in));
2. String name = reader.readLine();
3. Scanner scanner = **new** Scanner(system.in);
4. scanner.next;
5. Console c=System.console();
6. String n=c.readLine();

**Secound largest number**

**public** **class** SecoundLargestNumber {

**public** **int** num(**int** []a,**int** total) {

Arrays.*sort*(a);

**return** a[total-1];

}

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

SecoundLargestNumber s=**new** SecoundLargestNumber();

**int** a[]= {2,4,1,4,8};

**int** b[]= {45,65,87,9};

System.***out***.println(s.num(a,5));

}

**Sorted number**

public static void main(String[] args)

{

int[] arr = { 13, 7, 6, 45, 21, 9, 101, 102 };

Arrays.sort(arr);

System.out.println("Modified arr[] : %s",

Arrays.toString(arr));

}

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

Integer a[]={2,3,5,7,2};

Arrays.*sort*(a,Collections.*reverseOrder*());

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

}

**Reverse string in ascending order**

String name="anbu";

String rev="";

**for**(**int** i=name.length()-1;i>=0;i--) {

rev=rev+name.charAt(i);

}

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

**char** c[]=rev.toCharArray();

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

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

}

Arrays.*sort*(c);

System.***out***.println(Arrays.*toString*(c));

}

        System.out.println("Original Array : " +

                                Arrays.toString(arr));

        Collections.reverse(Arrays.asList(arr));

        System.out.println("Modified Array : " +

                                Arrays.toString(arr));

**Reverse collection**

List<String>  mylist = new ArrayList<String>();

        mylist.add("practice");

        mylist.add("code");

        mylist.add("quiz");

        mylist.add("geeksforgeeks");

        System.out.println("Original List : " + mylist);

        Collections.reverse(mylist);

        System.out.println("Modified List: " + mylist);

static void reverse(Integer myArray[])

{

Collections.reverse(Arrays.asList(myArray));

System.out.println("Reversed Array:" + Arrays.asList(myArray));

}

public static void main(String[] args)

{

Integer [] myArray = {1,3,5,7,9};

System.out.println("Original Array:" + Arrays.asList(myArray));

reverse(myArray);

}

}

Repeated words

1. String string = "Big black bug bit a big black dog on his big black nose";
2. **int** count;
4. //Converts the string into lowercase
5. string = string.toLowerCase();
7. //Split the string into words using built-in function
8. String words[] = string.split(" ");
10. System.out.println("Duplicate words in a given string : ");
11. **for**(**int** i = 0; i < words.length; i++) {
12. count = 1;
13. **for**(**int** j = i+1; j < words.length; j++) {
14. **if**(words[i].equals(words[j])) {
15. count++;
16. //Set words[j] to 0 to avoid printing visited word
17. words[j] = "0";
18. }
19. }
21. //Displays the duplicate word if count is greater than 1
22. **if**(count > 1 && words[i] != "0")
23. System.out.println(words[i]);
24. }

Dropdown options

public class TestDrpdownSorted {

@Test

public void runTestOnDocker() throws Exception {

String driverPath = "D:PATHchromedriver.exe";

System.setProperty("webdriver.chrome.driver", driverPath);

WebDriver driver = new ChromeDriver();

driver.get("https://chercher.tech/practice/practice-dropdowns-selenium-webdriver");

WebElement element = driver.findElement(By.xpath("//select[@id='animals']"));

Select se = new Select(element);

List<String> originalList = new ArrayList();

for (WebElement e : se.getOptions()) {

originalList.add(e.getText());

}

//----logic block starts

List<String> tempList= originalList;

Collections.sort(tempList);

Assert.assertEquals(tempList, originalList);

//----logic ends starts

}

}