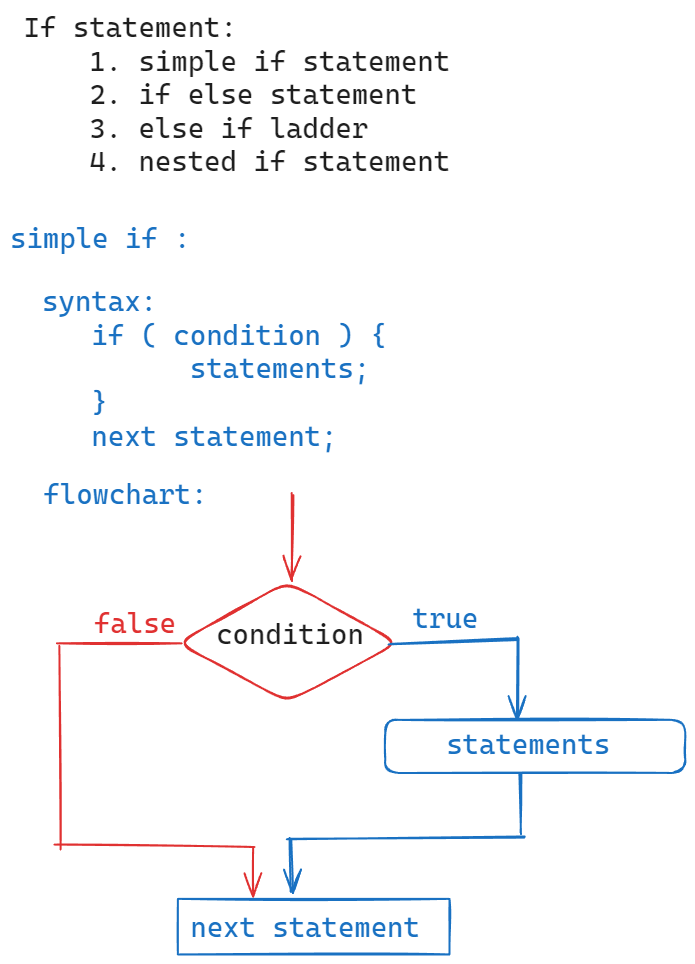
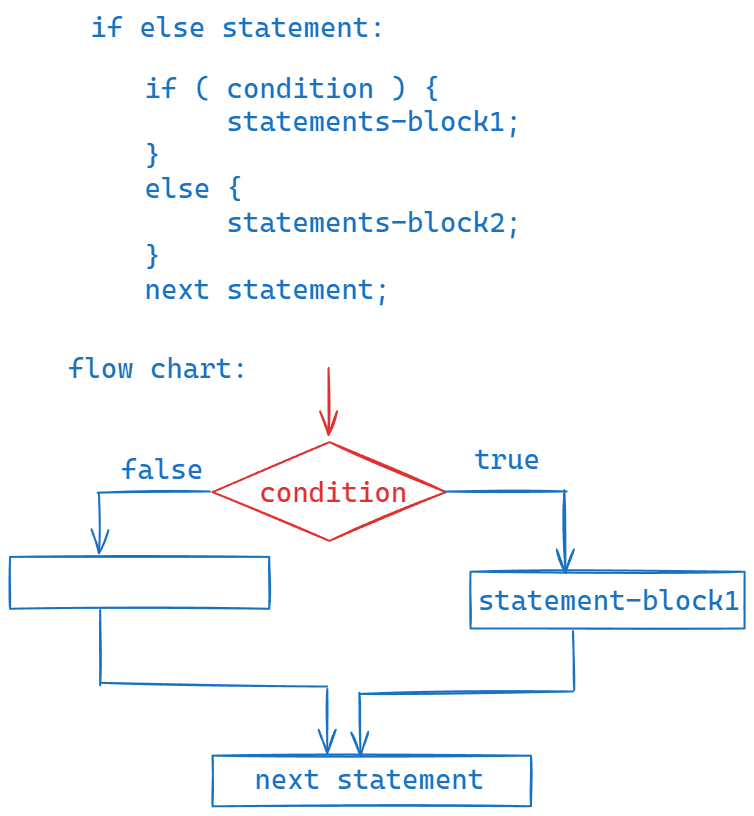
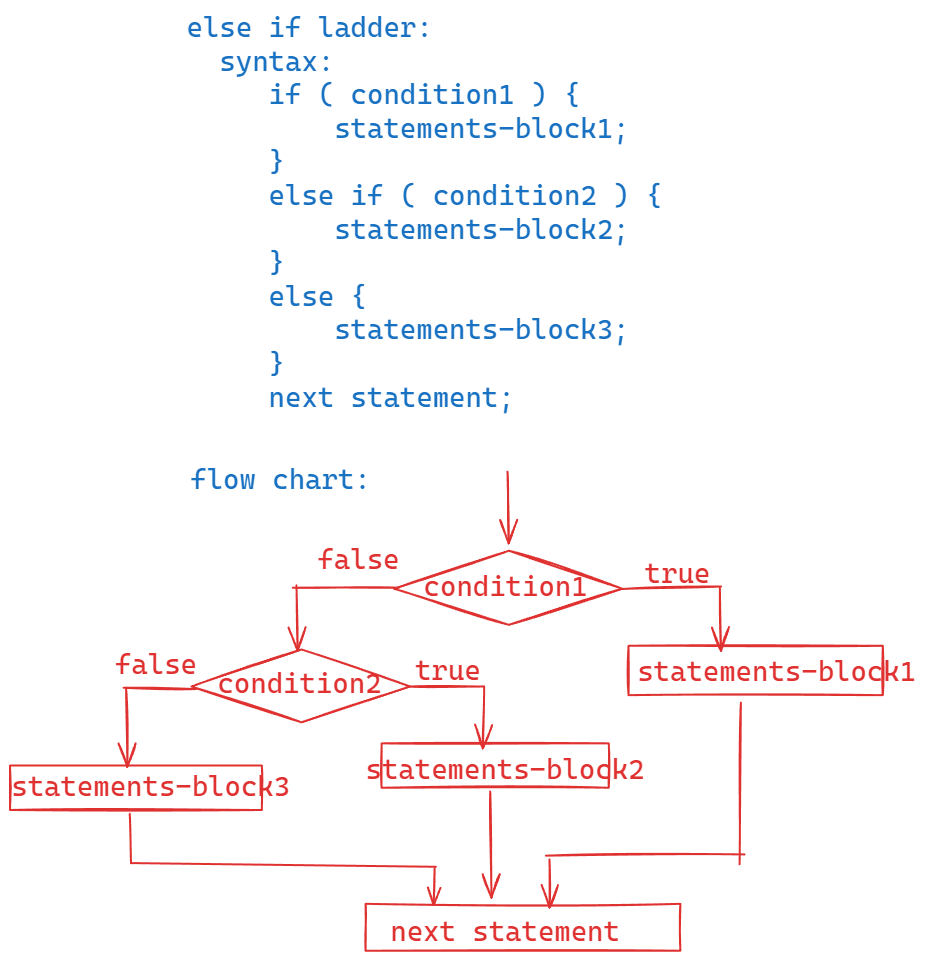
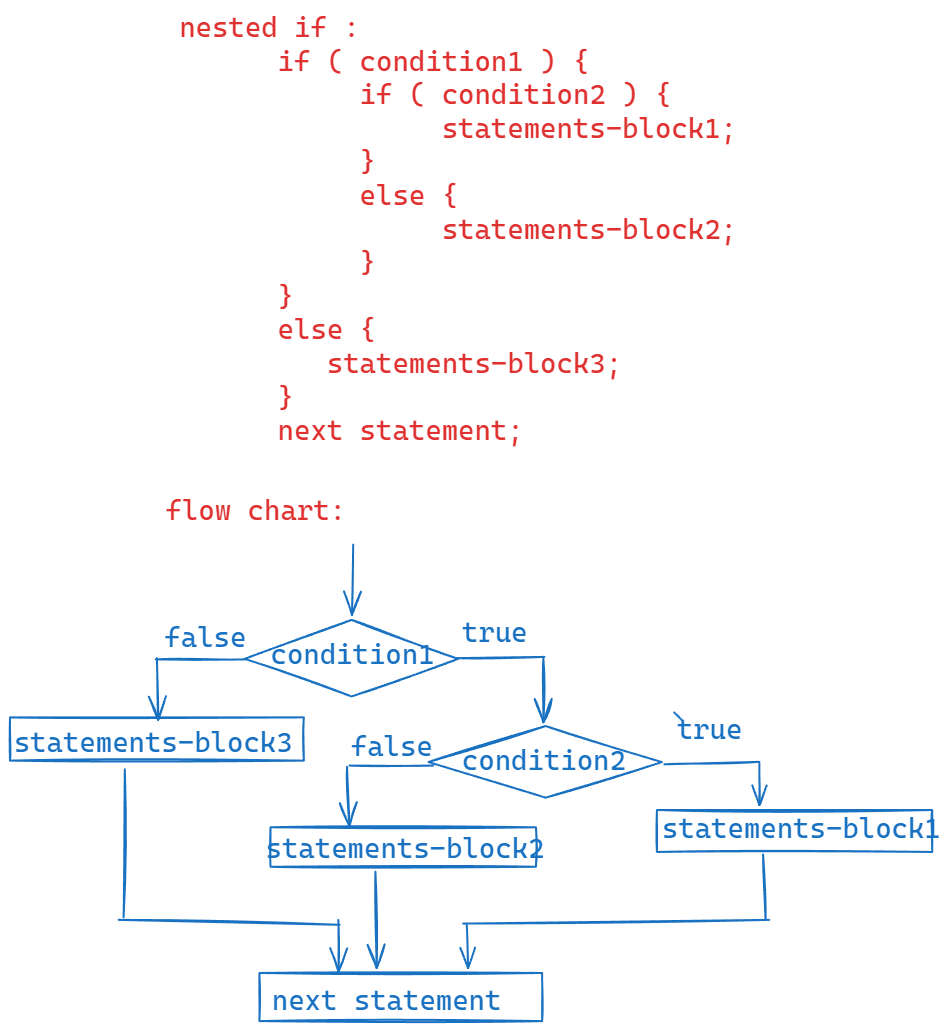
Control statements

1. Selection control statements
2. if statement
3. switch statement
4. Iteration control statements
5. For loop
6. While loop
7. Do while loop
8. Jump control statements
9. Break
10. Continue
11. Return
12. Exit









Comments:

1. Single line comment
2. Multiline comment
3. Documentation comment

// single line comment

/\*

Multiline comment

\*/

/\*\*

Documentation comment

\*/

/\*

\* This program uses if else statement

\* to check whether a given number is

\* even or odd.

\*/

**import** java.util.Scanner;

**public** **class** EvenOddTest {

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

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

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

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

*checkEvenOrOdd*(number);

}

**private** **static** **void** checkEvenOrOdd(**int** n) {

**if** ( n % 2 == 0 )

System.***out***.println(n + " is even number");

**else**

System.***out***.println(n + " is odd number");

}

}

/\*

\* Write a program to take two integers

\* as input and if they are same find

\* the sum, if they are different

\* then print double of the sum.

\* example:

\* sample input expected output

\* 5, 5 10

\* 6, 5 22

\*/

**import** java.util.Scanner;

**public** **class** Main {

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

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

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

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

System.***out***.println("Enter second number");

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

*findSum*(a, b);

}

**private** **static** **void** findSum(**int** a, **int** b) {

**if** ( a == b ) {

**int** c = a + b;

System.***out***.println("Sum = " + c);

}

**else** {

**int** d = (a + b) \* 2;

System.***out***.println("Sum = " + d);

}

}

}

/\*

\* Write a program to take an integer

\* as input and do the following.

\* 1. if it is divisible by 3 then display "zip".

\* 2. if it is divisible by 5 then display "zap".

\* 3. if it is divisible by 3 & 5 then display "hurray!".

\* 4. otherwise, display "not divisible by 3 or 5".

\*/

**import** java.util.Scanner;

**public** **class** TestClass {

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

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

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

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

*doAnswer*(n);

}

**private** **static** **void** doAnswer(**int** n) {

**if** ( n % 3 == 0 && n % 5 == 0 ) {

System.***out***.println("hurray!");

}

**else** **if** ( n % 3 == 0 ) {

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

}

**else** **if** ( n % 5 == 0 ) {

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

}

**else** {

System.***out***.println("not divisible by 3 or 5");

}

}

}

/\*

\* write a program to find the biggest

\* of three integers using nested if

\* statement.

\*/

**import** java.util.Scanner;

**public** **class** MainClass {

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

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

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

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

System.***out***.println("Enter second number");

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

System.***out***.println("Enter third number");

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

*findBigOfThree*(a, b, c);

}

**private** **static** **void** findBigOfThree(**int** a, **int** b, **int** c) {

**if**( a == b && b == c ) {

System.***out***.println("The three numbers are equal");

**return**;

}

**if** ( a > b ) {

**if** ( a > c ) {

System.***out***.println(a + " is big");

}

**else** {

System.***out***.println(c + " is big");

}

}

**else** {

**if**( b > c) {

System.***out***.println(b + " is big");

}

**else** {

System.***out***.println(c + " is big");

}

}

}

}

/\*

\* write a program to take username

\* and password as input.

\* if username is "scott"(in any case) and password

\* is "tiger" (in any case) then display

\* "Login successful". Otherwise, display

\* "Login failed."

\*/

**import** java.util.Scanner;

**public** **class** TestClass {

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

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

System.***out***.println("Please enter username");

String username = scan.nextLine();

System.***out***.println("Please enter password");

String password = scan.nextLine();

*checkUser*(username, password);

}

**private** **static** **void** checkUser(String username, String password) {

**if** ( username.equalsIgnoreCase("scott") && password.equalsIgnoreCase("tiger")) {

System.***out***.println("Login successful");

}

**else** {

System.***out***.println("Login failed.");

}

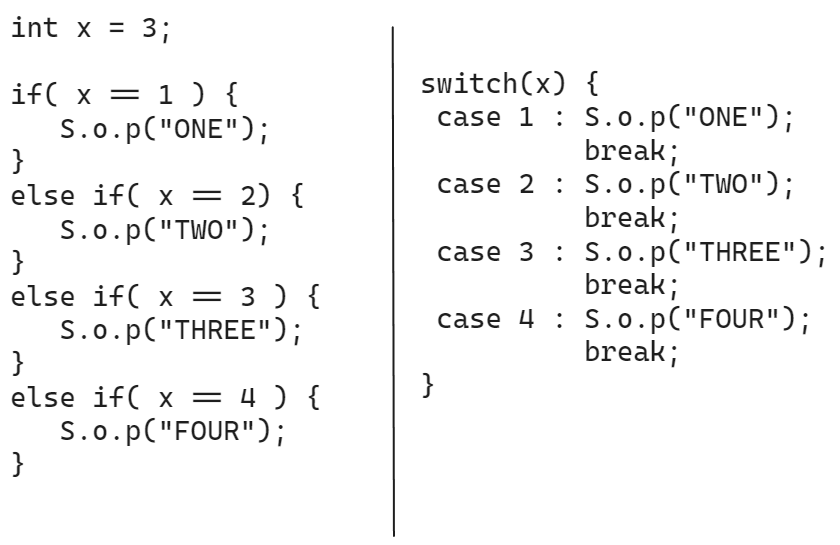
}

}

switch statement:

* When you want to write multiple conditions on the same variable and the condition is equality condition then we use switch statement.

For ex:



* In the above example, we have if statement at left side and switch statement at right side and both are same.
* Compared with if statement at left side, the switch statement at right side will improve the code readability and reduces the code complexity.
* That’s why switch statement is better in this case.

Syntax:

switch (variable/expression) {

case value1: statement-1;

break;

case value2: statement-2;

break;

case valuen: statement-n;

break;

default: statement;

}

* break statement is optional in a case.
* Default case is also optional.
* If no case value is matched with switch variable/expression then default case will be executed.
* In switch statement, the variable must be integer type variable(byte/short/int/long) or a character type variable or a string variable.
* In case statement, value must be integer type value or character type value or string value.

Ex1:

**int** d = 1;

**switch**(d) {

**case** 1.1: System.***out***.println("hello");

**break**;

**case** 1.2: System.***out***.println("bye");

**break**;

}

* This code has compile-time error. Because case value is double value/decimal point value.

Ex2:

**double** d = 1;

**switch**(d) {

**case** 1: System.***out***.println("hello");

**break**;

**case** 2: System.***out***.println("bye");

**break**;

}

* This code has compile-time error. Because switch statement has double variable.

Ex3:

**int** x = 5;

**switch**( x \* 4) {

**case** 10: System.***out***.println("hello");

**break**;

**case** 20: System.***out***.println("bye");

**break**;

}

Output: bye

Ex4:

**int** x = 5;

**switch**( x \* 2) {

**case** 10: System.***out***.println("hello");

**case** 20: System.***out***.println("bye");

**break**;

}

Output: hello

Bye

* In the first case, break statement is missing.
* If a case don’t have a break statement, then

Next cases also will be executed until break statement occurs.

Ex5:

**int** x = 1;

**switch** ( x ) {

**default**: System.***out***.print("Welcome ");

**case** 10: System.***out***.print("To ");

**case** 20: System.***out***.println("AshokIT!");

}

Output:

Welcome To AshokIT!

* Default case we can write anywhere in switch statement.
* If no case is matched then control executes default case and further cases also until break statement occurs.

Ex6:

**char** ch = 'i';

**switch** ( ch ) {

**case** 'a':

**case** 'e':

**case** 'i':

**case** 'o':

**case** 'u': System.***out***.println("Vowel");

**break**;

**default**: System.***out***.println("Consonent");

}

Output: Vowel

Ex7:

//This line gives me system day

String today = LocalDate.*now*().getDayOfWeek().toString();

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

**switch**(today) {

**case** "MONDAY": System.***out***.println("Monday is a lazy day");

**break**;

**case** "TUESDAY":

**case** "WEDNESDAY":

**case** "THURSDAY": System.***out***.println("The day is so-so");

**break**;

**case** "FRIDAY": System.***out***.println("Waiting for weekend");

**break**;

**case** "SATURDAY":

**case** "SUNDAY": System.***out***.println("Enjoy the weekend");

}

Output: FRIDAY

Waiting for weekend.

Iteration control statements:

* When you want to execute the same statements

For more than once then you have to use a loop.

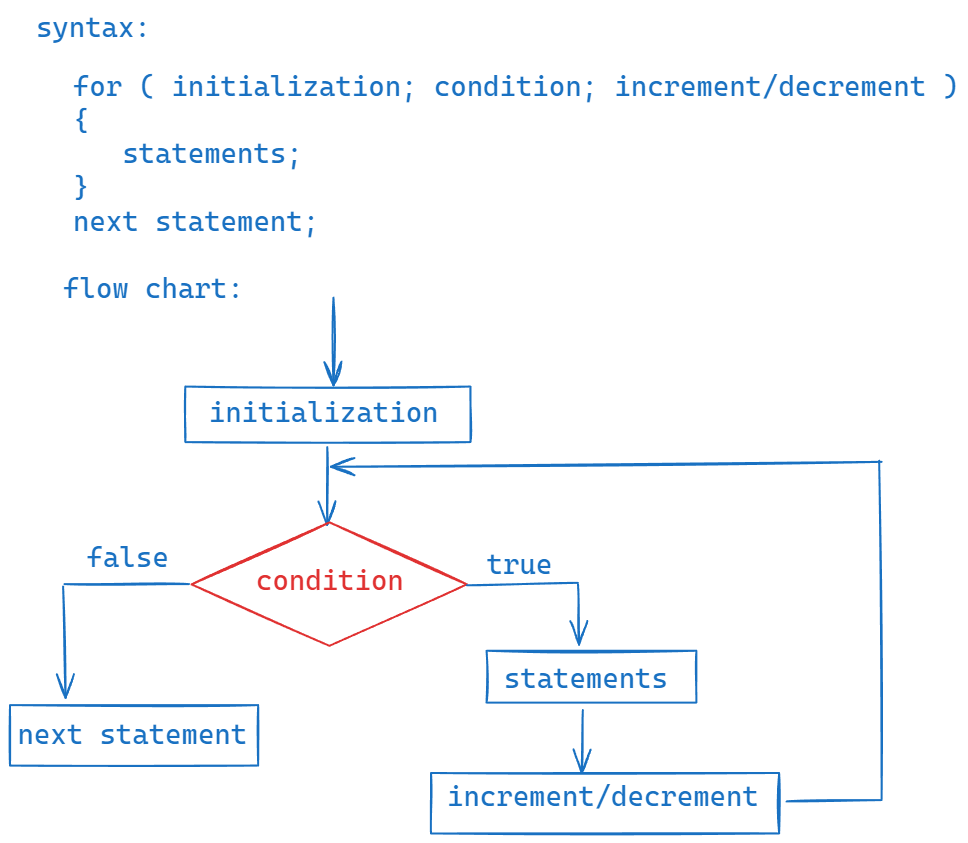
* Loops are 3 types.

1. for loop
2. while loop
3. do while loop

* In java, for loop is of 2 types.

1. for loop
2. for each loop/enhanced for loop.

for loop:



* when control enters into for statement, first the control goes to initialization part.
* Next, the control goes to the condition.
* If the condition is true, then control goes to the statements of the for loop.
* After executing statement, the control goes to the increment/decrement part.
* After inc/dec, then control goes back to the condition.
* If the condition is true, again the control goes statements of the for loop and executes them.
* This process will repeat until the given condition becomes false.
* If the condition is false, then the control goes to next statement after the for loop.

Ex1:

for( int i = 1; i <= 3; i++ ) {

S.o.p(“ i = “ + i);

}

Output:

i = 1

i = 2

i = 3

ex2:

for(int i=1; i<=5; i++); {

S.o.p(“hello”);

}

Output: hello

Ex3:

for(int i = 1; i <= 5; i++) {

S.o.p(i);

}

S.o.p(i);

Output: error

Ex4:

int i = 1;

for(int i = 1; i <= 5; i++) {

S.o.p(i);

}

S.o.p(i);

Output: error

Ex5:

int i=1;

for(i=10; i<=50; i=i+10) {

S.o.p(i);

}

S.o.p(i);

Output:

10

20

30

40

50

60

|| Date: 15th July, 24 ||

Ex6:

for(int i=1, j=10; i <= j; i++, j--) {

System.out.println( i+j );

}

Output:

11

11

11

11

11

Ex7:

int i = 1;

for( ; i <= 3;) {

S.o.p(i);

i++;

}

Output: 1

2

3

Ex8:

int i = 1;

for(;;) {

if( i >= 4) {

break;

}

S.o.p(i);

i++;

}

Ouput: 1

2

3

/\*

\* write a program to find the sum

\* of n natural numbers

\*/

**import** java.util.Scanner;

**public** **class** Example {

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

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

System.***out***.println("Enter n value to find the sum of n natural numbers");

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

*findSumOfNatural*(n);

}

**private** **static** **void** findSumOfNatural(**int** n) {

**int** sum = 0;

**for**(**int** i = 1; i <= n; i++) {

sum = sum + i;

}

System.***out***.println("sum = " + sum);

}

}

/\*

\* write a program to find the sum of

\* even and odd numbers separately in

\* n natural numbers.

\*/

**import** java.util.Scanner;

**public** **class** Example2 {

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

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

System.***out***.println("Enter n value to find the sum of even and odd numbers separately");

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

*findSumOfEvenOdd*(n);

}

**private** **static** **void** findSumOfEvenOdd(**int** n) {

**int** evenSum = 0, oddSum = 0;

**for**(**int** i = 1; i <= n; i++ ) {

**if** ( i % 2 == 0 ) {

evenSum += i; //evenSum = evenSum + i;

}

**else** {

oddSum += i; // oddSum = oddSum + i;

}

}

System.***out***.println("Even sum = " + evenSum);

System.***out***.println("Odd sum = " + oddSum);

}

}

/\*

\* write a program to print the multiplication

\* table of a given number

\* for example, n = 5

\* 5 \* 1 = 5

\* 5 \* 2 =10

\* .......

\* 5 \* 10=50

\*/

**import** java.util.Scanner;

**public** **class** Example3 {

**private** **static** **void** printTable(**int** n) {

**for**(**int** i=1; i <= 10; i++) {

System.***out***.println( n + " \* " + i +" = "+ (n\*i));

}

}

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

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

System.***out***.println("Enter n value to print multiplication table");

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

*printTable*(n);

}

}

|| Date: 16th July, 2024 ||

Prime number:

A whole number which has only two factors i.e., 1 and itself is a prime number.

* A whole number which has more than two factors is called a composite number.
* Prime numbers and composite numbers starts from 2.
* 0 and 1 are neither prime numbers nor composite numbers.

/\*

\* write a program to check whether a given

\* number is a prime number or not.

\*/

**import** java.util.Scanner;

**public** **class** PrimeNumberTest {

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

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

System.***out***.println("Please enter a number");

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

**boolean** flag = *isPrime*(n);

**if**( flag == **true** ) {

System.***out***.println( n + " : is a prime number");

}

**else** {

System.***out***.println( n + " : is not a prime number");

}

}

**private** **static** **boolean** isPrime(**int** n) {

**boolean** b = **true**;

**for**( **int** i = 2; i <= Math.*sqrt*(n); i++) {

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

b = **false**;

**break**;

}

}

**return** b;

}

}

Perfect number:

If sum of the factors of a number, excluding that number is equal to the given number then it is a perfect number.

Example: 6 and 28.

/\*

\* write a program to check whether

\* a given number is perfect number or not.

\*/

**import** java.util.Scanner;

**public** **class** PerfectNumberTest {

**private** **static** **boolean** isPerfect(**int** n) {

**int** sum = 0;

**for**(**int** i=1; i <= n/2; i++) {

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

sum = sum + i;

}

}

**if**(sum == n)

**return** **true**;

**else**

**return** **false**;

}

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

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

System.***out***.println("Please enter a number");

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

**boolean** status = *isPerfect*(n);

**if**(status == **true**) {

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

}

**else** {

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

}

}

}

Fibonacci series:

* The first two terms of the Fibonacci series are 0 and 1
* Next term is a sum of the previous two terms.

Ex: Fibonacci series of 5 terms.

0

1

1

2

3

Fibonacci series of 10 terms.

0

1

1

2

3

5

8

13

21

34

|| DATE : 17th July, 2024 ||

/\*

\* write a program to print fibonacci series

\* of n terms.

\*/

**import** java.util.Scanner;

**public** **class** FibonacciSeries {

**private** **static** **void** printSeries(**int** number) {

**int** firstTerm = 0, secondTerm = 1;

**if**(number == 1) {

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

**return**;

}

**if**(number == 2) {

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

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

**return**;

}

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

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

**for**(**int** i = 3; i <= number; i++) {

**int** nextTerm = firstTerm + secondTerm;

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

firstTerm = secondTerm;

secondTerm = nextTerm;

}

}

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

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

System.***out***.println("Enter number of terms of the fibonacci series ");

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

*printSeries*(number);

}

}

===========================================================

|| DATE: 18th July 2024 ||

Nested for loop programs

/\*

\* write a program to print the primes

\* with in given range

\*/

**import** java.util.Scanner;

**public** **class** PrimesInRange {

**private** **static** **void** findPrimes(**int** x, **int** y) { //definition point

System.***out***.println("The prime numbers between : " + x + " and : "+ y);

**for** ( **int** i = x; i <= y; i++) {

**boolean** flag = **true**;

**for** ( **int** j = 2; j <= Math.*sqrt*(i); j++) {

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

flag = **false**;

**break**;

}

} // end inner loop

**if** ( flag == **true** )

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

} //end outer loop

}

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

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

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

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

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

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

*findPrimes*(x, y); //calling point

}

}

==========================================================

/\*

\* write a program to print perfect numbers

\* with in given range

\*/

**import** java.util.Scanner;

**public** **class** PerfectNumbersRange {

**private** **static** **void** findPerfects(**int** x, **int** y) {

System.***out***.println("The perfect numbers between : " + x + " and : " + y);

**for** ( **int** i = x; i <= y; i++) {

**int** sum = 0;

**for** ( **int** j = 1; j <= i/2; j++) {

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

sum = sum + j;

}

} //end inner loop

**if** ( sum == i )

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

} //end outer loop

} //end findPerfects method

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

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

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

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

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

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

*findPerfects*(x, y);

}

}

**Pattern programs**:

1. Star patterns
2. Number patterns
3. Character patterns.

Star pattern:

\*

\* \*

\* \* \*

Number pattern:

1

1 2

1 2 3

Character pattern:

A

A B

A B C

/\*

\* write a program to display the

\* below pattern.

\* if n = 5,

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

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

\*

\*/

**import** java.util.Scanner;

**public** **class** Example {

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

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

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

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

*printPattern*(n);

}

**private** **static** **void** printPattern(**int** n) {

**for** ( **int** i = 1; i <= n; i++ )

{

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

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

}

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

}

}

}

==================================================================

|| DATE : 20th July, 2024 ||

/\*

\* write a program to print the

\* below pattern.

\* if n = 5,

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

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*/

**import** java.util.Scanner;

**public** **class** PatternExample {

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

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

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

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

*printPattern*(n);

}

**private** **static** **void** printPattern(**int** n) {

**for** ( **int** i = 1; i <= n; i++ )

{

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

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

}

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

}

}

}

/\*

\* write a program to display

\* the below pattern.

\* if n = 5

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

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

\*/

**import** java.util.Scanner;

**public** **class** PatternExample {

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

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

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

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

*printPattern*(n);

}

**private** **static** **void** printPattern(**int** n) {

**for** ( **int** i = 1; i <= n; i++ ) {

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

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

}

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

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

}

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

}

}

}

/\*

\* write a program to display the

\* below pattern.

\* if n = 5,

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

\* \* \*

\* \* \*

\* \* \*

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

\*/

import java.util.Scanner;

public class PatternExample {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

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

int n = scan.nextInt();

printPattern(n);

}

private static void printPattern(int n) {

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

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

if ( i == 1 || i == n || j == 1 || j == n )

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

else

System.out.print(" ");

}

System.out.println();

}

}

}

/\*

\* write a program to display the

\* below pattern

\* if n = 5,

\* \*

\* \* \* \*

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

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

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

\*/

import java.util.Scanner;

public class PatternExample {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

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

int n = scan.nextInt();

printPattern(n);

}

private static void printPattern(int n) {

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

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

System.out.print(" ");

}

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

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

}

System.out.println();

}

}

}

/\*

\* write a program to display

\* the star pattern for left and

\* right diagonal.

\*/

import java.util.Scanner;

public class PatternExample {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

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

int n = scan.nextInt();

printPattern(n);

}

private static void printPattern(int n) {

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

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

if( i == j || i + j == n + 1 )

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

else

System.out.print(" ");

}

System.out.println();

}

}

}

/\*

\* write a program to display the

\* below pattern.

\* if n = 5

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

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*/

import java.util.Scanner;

public class PatternExample {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

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

int n = scan.nextInt();

printPattern(n);

}

private static void printPattern(int n) {

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

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

System.out.print(" ");

}

for( int k = 1; k <= n - i + 1; k++) {

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

}

System.out.println();

}

}

}

/\*

\* write a program to display the

\* below pattern

\* if n = 5,

\* \*

\* \* \* \*

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

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

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

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

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

\* \* \* \*

\* \*

\*/

import java.util.Scanner;

public class PatternExample {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

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

int n = scan.nextInt();

printPattern(n);

}

private static void printPattern(int n) {

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

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

System.out.print(" ");

}

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

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

}

System.out.println();

}

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

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

System.out.print(" ");

}

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

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

}

System.out.println();

}

}

}

|| DATE: 22nd July, 2024 ||

/\*

\* write a program to display the below pattern.

\* if n = 5,

\* 1

\* 0 1

\* 1 0 1

\* 0 1 0 1

\* 1 0 1 0 1

\*/

**import** java.util.Scanner;

**public** **class** PatternExample {

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

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

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

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

*printPattern*(n);

}

**private** **static** **void** printPattern(**int** n) {

**int** start;

**for** ( **int** i = 1; i <=n; i++ ) {

**if** ( i % 2 == 0 )

start = 0;

**else**

start = 1;

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

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

start = 1 - start;

}

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

}

}

}

/\*

\* write a program to display

\* pascal triangle.

\* if n = 5,

\* 1

\* 1 1

\* 1 2 1

\* 1 3 3 1

\* 1 4 6 4 1

\*

\*/

**import** java.util.Scanner;

**public** **class** PascalTriangle {

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

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

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

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

*printPattern*(n);

}

**private** **static** **void** printPattern(**int** n) {

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

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

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

}

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

System.***out***.print(*nCr*(i, k)+" ");

}

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

}

}

**private** **static** **int** nCr(**int** n, **int** r) {

**return** *factorial*(n) / ( *factorial*(r) \* *factorial*(n-r) ) ;

}

**private** **static** **int** factorial(**int** n) {

**if**( n <= 1 )

**return** 1;

**int** fact = 1;

**for** ( **int** i = 1; i <=n; i++) {

fact = fact \* i;

}

**return** fact;

}

}

|| DATE: 23rd July, 2024 ||

while loop:

* This loop is also called as entry-control loop.
* If you know exactly how many times to iterate a group of statements then you use for loop.
* If you don’t know how many times to iterate the group of statements, but until a given condition becomes false then use while loop.

Syntax:

while ( condition ) {

statements;

}

next statement;

/\*

\* write a program to find the

\* sum of the digits of a given

\* number.

\* ex: number = 123

\* op: 6

\*/

**import** java.util.Scanner;

**public** **class** Main {

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

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

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

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

*findSumOfDigits*(number);

}

**private** **static** **void** findSumOfDigits(**int** number) {

**int** sum = 0;

**while** ( number > 0 ) {

**int** d = number % 10;

sum = sum + d;

number = number / 10;

}

System.***out***.println("Sum of the digits of this number : " + sum);

}

}

/\*

\* write a program to check whether

\* a given number is armstrong number

\* or not.

\* armstrong number: if sum of the nth power of

\* each digit of a number is equal to the same number

\* then it is called an armstrong number.

\* ex: number = 153

\* 3 3 3

\* = 1 + 5 + 3

\* = 1 + 125 + 27

\* = 153

\*

\* number = 1634

\* 4 4 4 4

\* = 1 + 6 + 3 + 4

\* = 1 + 1296 + 81 + 256

\* = 1634

\*/

**import** java.util.Scanner;

**public** **class** ArmstrongTest {

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

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

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

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

*checkArmstrongNumber*(number);

}

**private** **static** **void** checkArmstrongNumber(**int** number) {

**int** temp = number;

String str = String.*valueOf*(number);

**int** len = str.length();

**int** sum = 0;

**while**(number > 0) {

**int** d = number % 10;

sum = sum + (**int**) Math.*pow*(d, len);

number = number / 10;

}

**if**( sum == temp ) {

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

}

**else** {

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

}

}

}

|| DATE : 25th July, 2024 ||

/\*

write a program to find the sum

of even position digits and odd

position digits separately

ex: number = 23197

evenPositionSum = 12

oddPositionSum = 10

\*/

**import** java.util.Scanner;

**public** **class** EvenOddPosition {

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

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

System.***out***.println("Please enter a number");

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

*findEvenOddPositionSum*(number);

}

**private** **static** **void** findEvenOddPositionSum(**int** number) {

String str = String.*valueOf*(number);

**int** len = str.length();

**int** evenPositionSum = 0;

**int** oddPositionSum = 0;

**boolean** isEvenPosition = **false**;

**if** ( len % 2 == 0 ) {

isEvenPosition = **true**;

}

**while**( number > 0 ) {

**int** d = number % 10;

**if**(isEvenPosition) {

evenPositionSum = evenPositionSum + d;

isEvenPosition = !isEvenPosition;

}

**else** {

oddPositionSum = oddPositionSum + d;

isEvenPosition = !isEvenPosition;

}

number = number / 10;

}

System.***out***.println("Sum of even position digits : " + evenPositionSum);

System.***out***.println("Sum of odd position digits : " + oddPositionSum);

}

}

do while loop:

do {

statements;

} while(condition);

* Use do while loop when you want to execute the statements of a loop

For at least once.

* Do while loop is also called exit control loop. Because, the loop condition is defined after the body of the loop.
* Do while loop is mostly used in menu driven programs.

**import** java.util.Random;

**import** java.util.Scanner;

**public** **class** MainClass {

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

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

**int** number;

**do** {

System.***out***.println("Guess a number between 0 to 10");

number = scan.nextInt();

} **while**( number < 0 || number > 10);

**int** temp = **new** Random().nextInt(11);

**if**(temp == number) {

System.***out***.println("Your guess : "+ number);

System.***out***.println("Random number : " + temp);

System.***out***.println("Your guess is correct!");

}

**else** {

System.***out***.println("Your guess : "+ number);

System.***out***.println("Random number : " + temp);

System.***out***.println("Your guess is wrong!");

}

}

}

===================================================================

Jumping control statements:

1. break
2. continue
3. return
4. exit

ex1:

int x = 5;

if ( x % 2 != 0 ) {

break;

}

* error. Because, we can’t use break statement without a loop or a switch case.

Ex2:

for(int i=1; i<=5; i++) {

if ( i>3 )

break;

S.o.p(i);

}

Output: 1

2

3

Ex3:

for(int i=1; i<=3; i++)

{

for(int j=1; j<=3; j++)

{

if ( i+j > 3 )

break;

S.o.println(j);

}

S.o.println(i);

}

Output: 1

2

1

1

2

3

Note: If you have break statement in inner loop, then it breaks the execution of inner loop only.

Ex4:

one: for(int i=1; i<=3; i++)

{

two: for(int j=1; j<=3; j++)

{

if( i+j > 3 )

break one;

S.o.println(j);

}

S.o.println(i);

}

Output: 1

2

continue:

* continue statement, will skip the remaining statements of the loop and moves the control to the next iteration.

ex:

int sum = 0;

for(int i=1; i<=7; i++)

{

if ( i % 2 == 0 )

continue;

sum = sum + i;

}

S.o.println(sum);

Output: 16

return :

* return statement we can write in a method and if the return type of that method is not void.

Ex1:

public int m1() {

-------

-------

return true; //error

}

Ex2:

public void m2() {

------

------

return 1; //error

}

Ex3:

public Employee findEmployee(int empno) { //factory method

----------

----------

return new Employee(101, “Akash”); //valid

}

Ex4:

public double getPrice() {

retun 83.7; //valid

}

factory method: It is a method which returns an object.

Ex5:

public void m1() {

----

----

return; // just returns the control without a value.

}

==============================================================

ARRAYS

* Suppose, we want to store multiple values then we need multiple variables.
* If we take multiple variables in a program, it will become a complex program.
* The memory allocation will be done for the variables at different places in JVM.
* So, to store the data or to fetch the data of a variable, JVM has to search for that variable in the entire memory.
* This will lead to poor performance of the program.
* The solution for these problems is use arrays.
* With arrays concept, we can store multiple values with a single variable.
* The memory will be allocated sequentially to the values. So fetching the data or storing the data becomes easy for the JVM.

How to create an array?

datatype[] variablename = new datatype[size];

(or)

datatype variablename[] = new datatype[size];

(or)

datatype[] variablename; //array declaration

variablename = new datatype[size]; //array creation

ex:

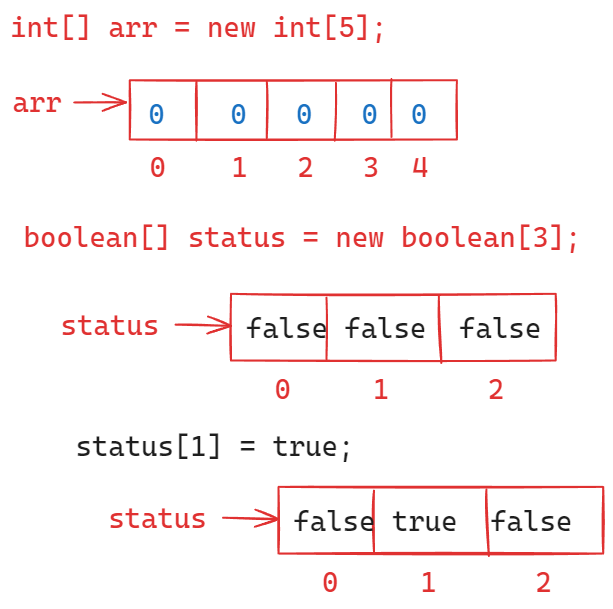
int[] arr = new int[]; //error

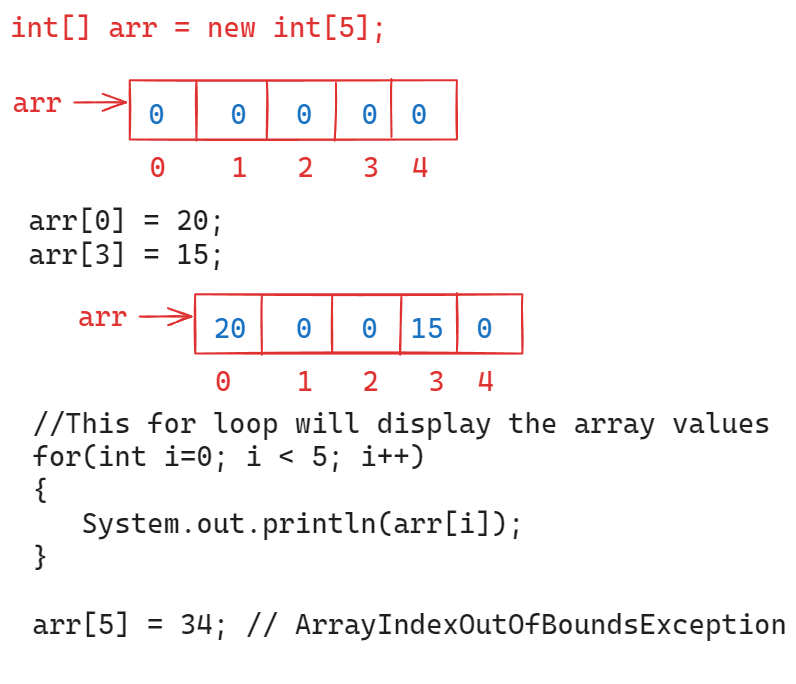
int[] arr = new int[5]; //valid

int[] arr;

arr = new int[5]; //valid

* When an array is created, the array locations are filled with default values of the data type.
* If you want to access any array location, you have to use index.
* Suppose, if an array size is n then indexes are from 0 to n-1.





Finding the length of an array:

* To find the size of an array, we can use length attribute.

For ex:

int[] arr = new int[5];

S.o.println( arr.length ); // 5

\* In Java, we have length() method also, and it is used to find the size of a string value.

String str = “Ashokit”;

S.o.println( str.length ); //error

S.o.println( str.length() ); // 7

For ex:

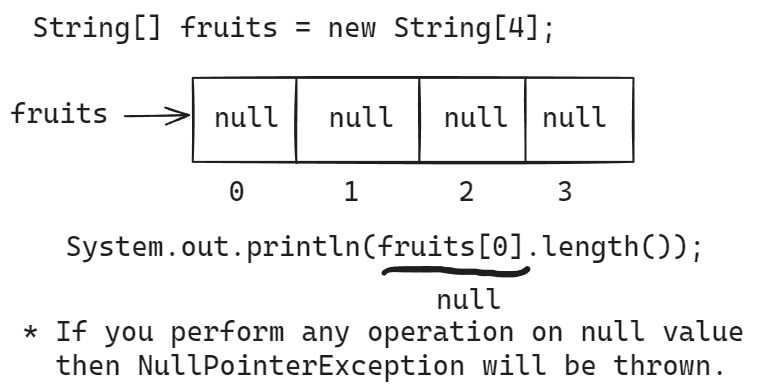
String[] fruits = new String[4];

S.o.println( fruits.length ); //correct op: 4

S.o.println( fruits.length() ); // error

S.o.println( fruits[0].length ); //error

S.o.println( fruits[0].length() ); //NullPointerException



For ex:

String str1 = null;

String str2 = “null”;

S.o.println( str1.length() ); //NullPointerException

S.o.println( str2.length() ); // 4

Direct initialization/static initialization of array:

int[] ar = new int[] { 10, 34, -8, 12 };

(or)

int[] ar = { 10, 34, -8, 12 };

======================================================

Creating an array by taking array size from the user:

-----------------------------------------------------

Scanner scan = new Scanner(System.in);

System.out.println(“Enter the size of an array”);

int size = scan.nextInt();

int[] arr = new int[size];

/\*

\* write a program to find the sum of

\* array elements, by taking the size and

\* elements of an array from the user.

\*/

**import** java.util.Scanner;

**public** **class** SumOfArrayElements {

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

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

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

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

**int**[] arr = **new** **int**[n];

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

System.***out***.println("Enter the element at index : " + i);

arr[i] = scan.nextInt();

}

*findSumOfElements*(arr);

}

**private** **static** **void** findSumOfElements(**int**[] arr) {

**int** sum = 0;

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

sum = sum + arr[i];

}

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

}

}

/\*

\* write a program to find the second

\* largest element in the given array.

\*/

**import** java.util.Arrays;

**public** **class** SecondLargestInArray {

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

**int**[] arr = { 13, 46, 24, 52, 20, 9 };

**int** n = arr.length;

*findSecondLargeElement*(arr, n);

}

**private** **static** **void** findSecondLargeElement(**int**[] arr, **int** n) {

**int** largest = arr[0];

**int** second\_largest = Integer.***MIN\_VALUE***;

**for** ( **int** i = 1; i < n; i++ ) {

**if** ( arr[i] > largest ) {

second\_largest = largest;

largest = arr[i];

}

**else** **if** ( arr[i] != largest && arr[i] > second\_largest) {

second\_largest = arr[i];

}

}

System.***out***.println("Largest element : " + largest);

System.***out***.println("Second Largest element : " + second\_largest);

}

}

Date: 30th July, 2024

/\*

\* write a program to search for an element

\* in an array using binary search.

\*

\* binary search:

\* --> For binary search, the array elements must be in

\* ascending sorting order.

\* --> 1. find the mid point of the array ( low + high ) / 2.

\* --> 2. if element at mid point is equal to the searching element

\* then element is found. Stop searching.

\* 3. If element at mid point is greater than searching element

\* then search for the element at left side of the mid point

\* ( high = mid - 1)

\* 4. if element at mid point is less than searching element

\* then search for the element at right side of the mid point

\* (low = mid + 1)

\* --> repeat the steps 1 thru 4 until low > high

\*/

**import** java.util.Arrays;

**public** **class** BinarySearch {

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

**int**[] arr = { 13, 56, 46, 11, 9, 34 };

**int** k = 34;

*binarySearch*(arr, k);

}

**private** **static** **void** binarySearch(**int**[] arr, **int** k) {

Arrays.*sort*(arr);

**int** low = 0;

**int** high = arr.length - 1;

**boolean** found = **false**;

**while** ( low <= high ) {

**int** mid = ( low + high ) / 2;

**if**( arr[mid] == k ) {

System.***out***.println("Element is found");

found = **true**;

**break**;

}

**else** **if** ( arr[mid] > k ) {

high = mid - 1;

}

**else** {

low = mid + 1;

}

}

**if**(found == **false**) {

System.***out***.println("Element is not found");

}

}

}

/\*

\* write a program to remove the duplicate

\* elements from the given array.

\* ex:

\* int[] arr = { 3, 1, 4, 2, 1, 3, 5}

\* output : { 1, 2, 3, 4, 5 }

\*/

**import** java.util.Arrays;

**public** **class** RemoveDuplicates {

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

**int**[] arr = { 3, 1, 4, 2, 1, 3, 5};

*removeDuplicates*(arr);

}

**private** **static** **void** removeDuplicates(**int**[] arr) {

Arrays.*sort*(arr);

**int** i = 0;

**for** ( **int** j = 1; j < arr.length; j++ ) {

**if** ( arr[i] != arr[j] ) {

i++;

arr[i]= arr[j];

}

}

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

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

}

}

}

/\*

\* write a program to display max consecutively

\* occurred element in the given array.

\*

\* ex :

\* int[] arr = { 1, 1, 0, 2, 2, 2, 1, 1 }

\* output:

\* Element : 2

\* max consecutively occurred : 3 times

\*/

**public** **class** MaxOccurance {

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

**int**[] arr = { 1, 1, 0, 2, 2, 2, 1, 1, 5, 5, 5, 5, 1, 1 };

*findMaxConsecutiveOccurred*(arr);

}

**private** **static** **void** findMaxConsecutiveOccurred(**int**[] arr) {

**int** count = 1;

**int** k=0;

**int** maxi = 0;

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

**if** ( arr[i] == arr[i+1] ) {

count++;

}

**else** {

count = 1;

}

**if**( count > maxi ) {

maxi = count;

k = arr[i];

}

}

System.***out***.println("Element : " + k);

System.***out***.println("Max consecutive times occurred : " + maxi);

}

}

|| DATE: 31st July, 2024 ||

selection sort:

---------------

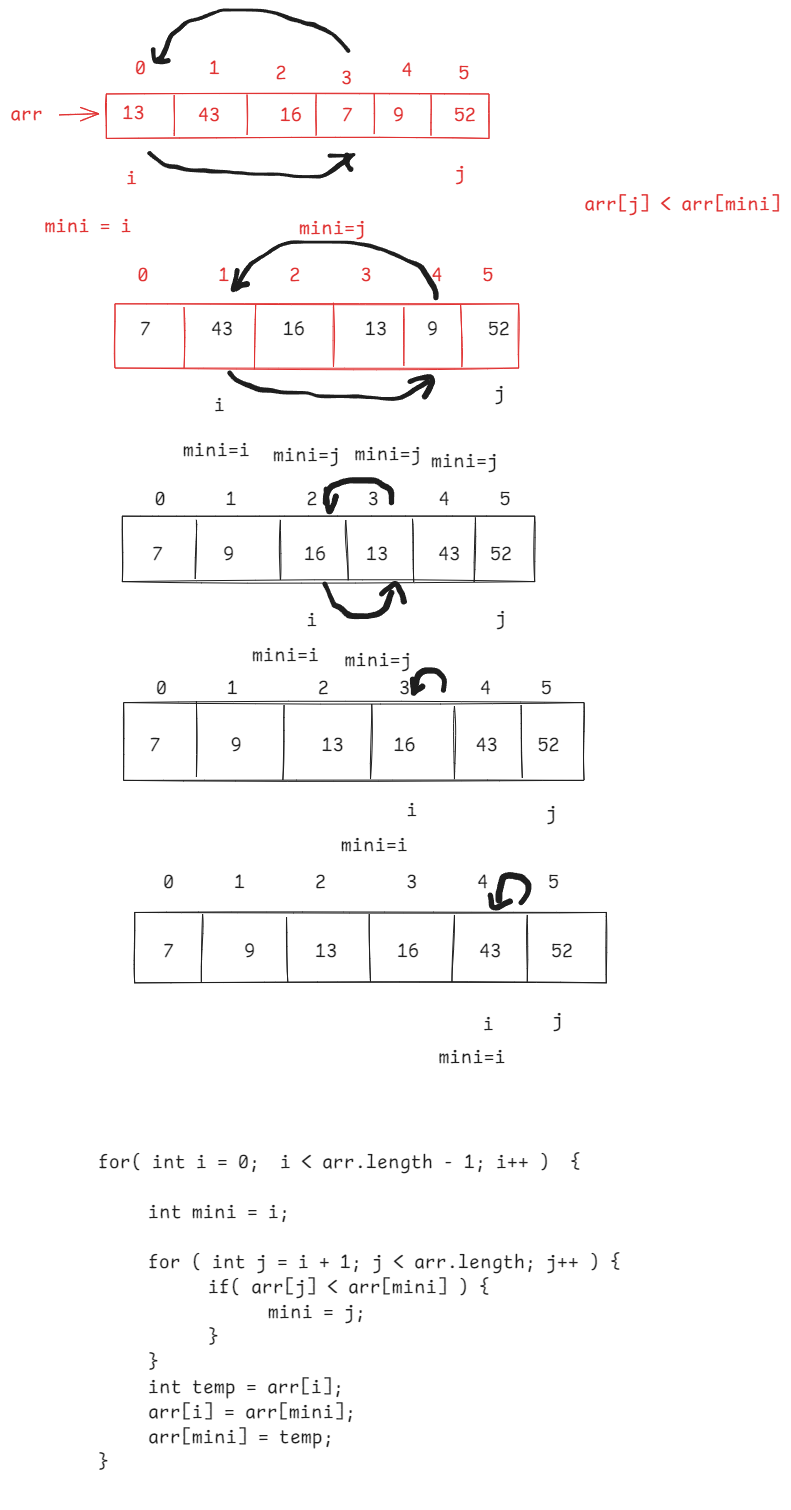
* In selection sort algorithm, first we need to find the

smallest element of the array, place it in the first position.

* next find the second smallest element of the array and place it

in the second position.

* continue this algorithm until n-1 elements are sorted.



/\*

\* write a program to sort the elements of an array

\* in ascending order with selection sort algorithm.

\*/

**public** **class** SelectionSortTest {

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

**int**[] arr = { 23, 35, 12, 67, 9 };

*selectionSort*(arr);

}

**private** **static** **void** selectionSort(**int**[] arr) {

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

**int** mini = i;

**for** ( **int** j = i + 1; j < arr.length; j++ ) {

**if**( arr[j] < arr[mini])

mini = j;

}

**int** temp = arr[i];

arr[i] = arr[mini];

arr[mini] = temp;

}

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

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

}

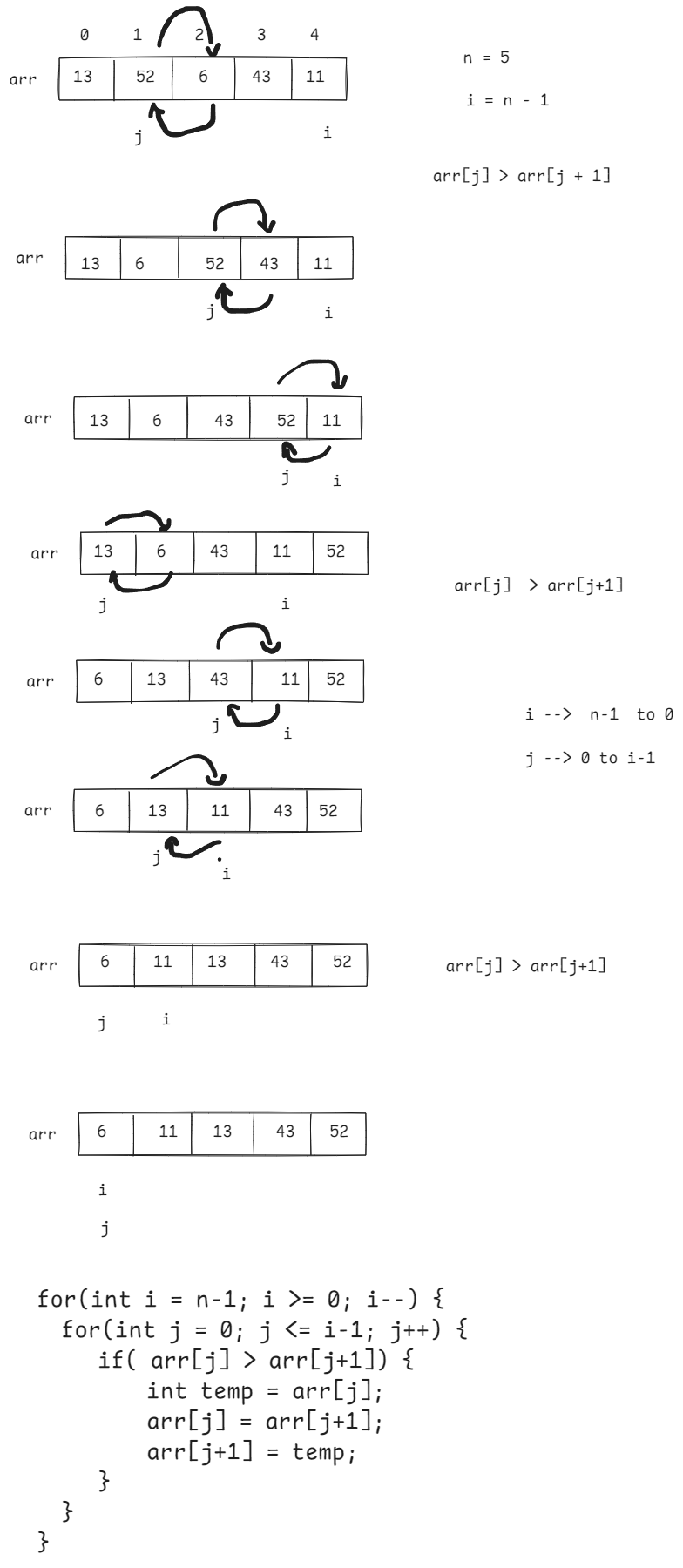
}

}

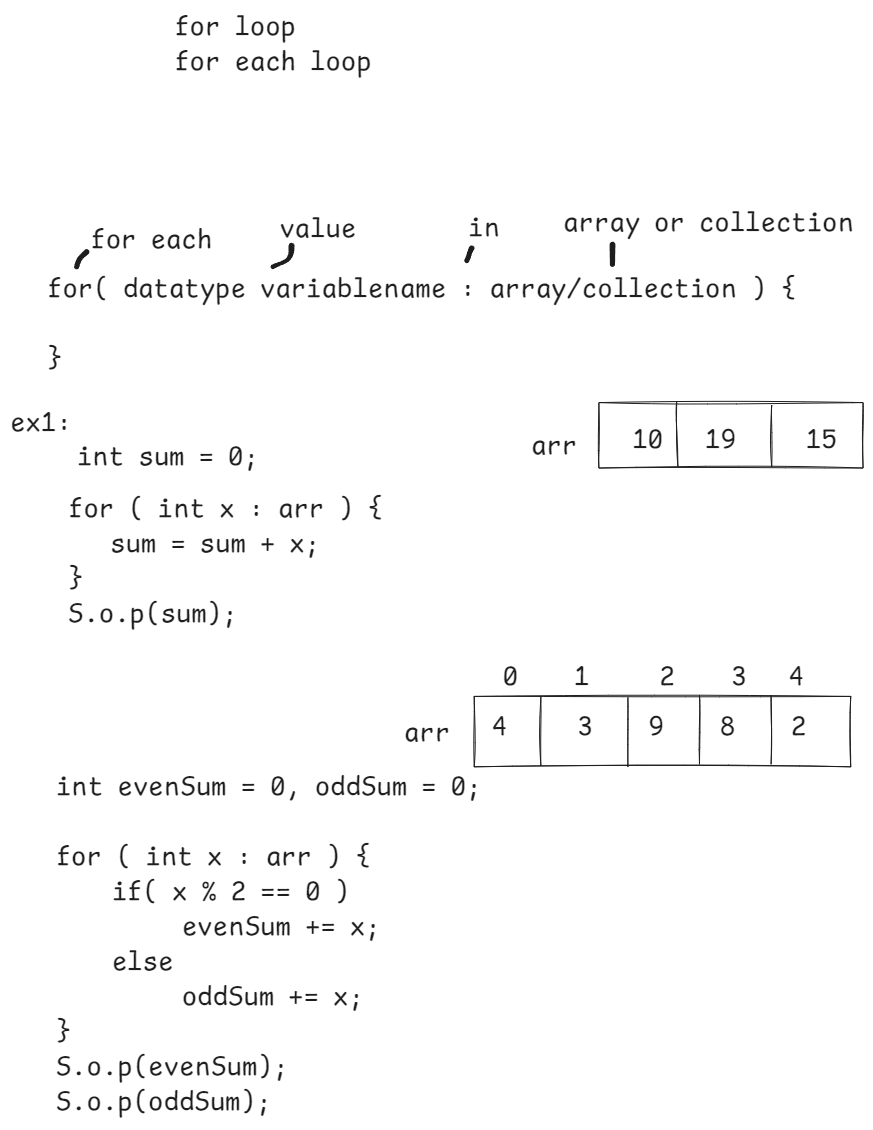
Bubble sort:

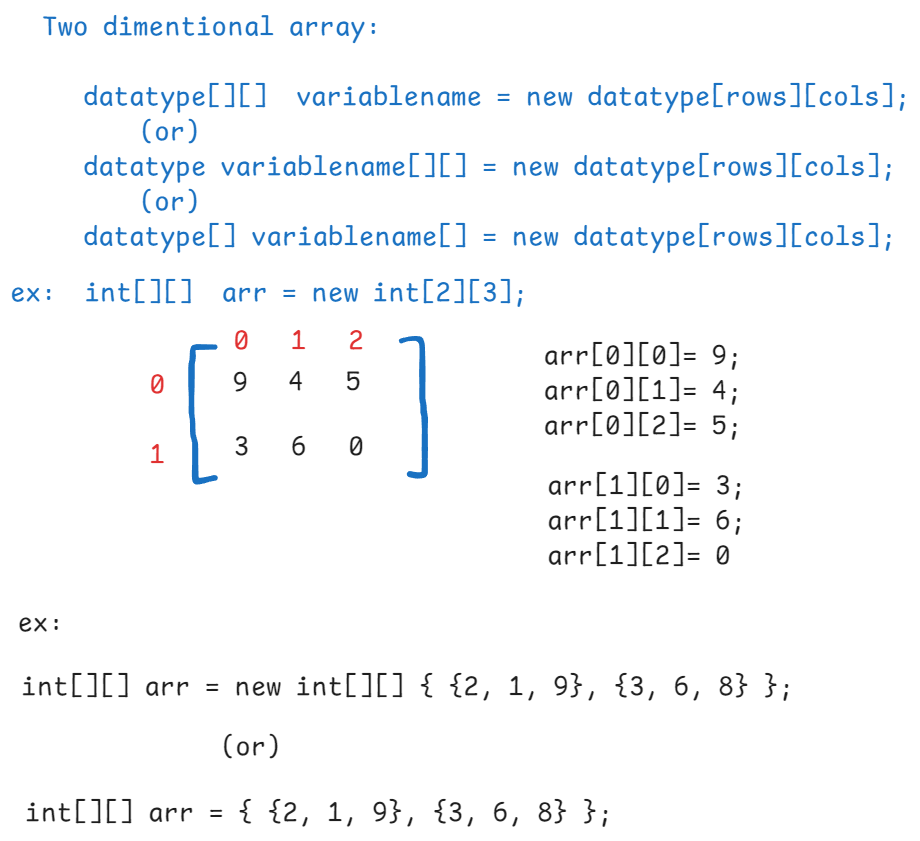
-----------

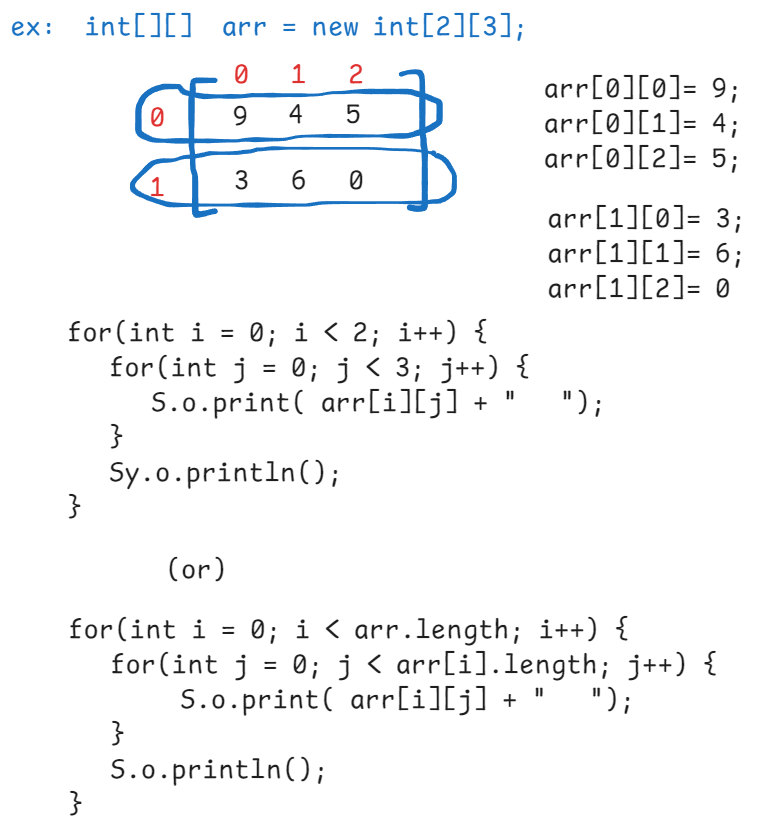
* In bubble sort, first find the highest element in array and place it at the last index.
* Next find the second highest element in array and place it at the second from last position.
* continue this, until all the elements are sorted.



|| DATE: 1st Aug, 2024 ||







/\*

\* write a program to find the sum of even

\* and odd elements of a matrix separately.

\*/

**import** java.util.Scanner;

**public** **class** MatrixEvenOddSum {

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

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

System.***out***.println("Enter the rows");

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

System.***out***.println("Enter the cols");

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

**int**[][] arr = **new** **int**[rows][cols];

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

**for**(**int** j = 0; j < arr[i].length; j++ ) {

System.***out***.println("Enter the element at ( "+ i +" , "+ j + " )");

arr[i][j] = scan.nextInt();

}

}

*findEvenOddSum*(arr);

}

**private** **static** **void** findEvenOddSum(**int**[][] arr) {

**int** evenSum = 0, oddSum = 0;

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

**for** (**int** j = 0; j < arr[i].length; j++ ) {

**if** ( arr[i][j] % 2 == 0 ) {

evenSum += arr[i][j];

}

**else** {

oddSum += arr[i][j];

}

}

}

System.***out***.println("Even sum : " + evenSum );

System.***out***.println("Odd sum : " + oddSum);

}

}

/\*

\* write a program to add the two matrices

\*/

**public** **class** MatrixAddition {

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

**int**[][] matrix1 = { {2, 3} , {0, 2}, {9, 1} };

**int**[][] matrix2 = { {2, 0} , {7, 3}, {8, 5} };

*add*(matrix1, matrix2);

}

**private** **static** **void** add(**int**[][] matrix1, **int**[][] matrix2) {

**int**[][] matrix3 = **new** **int**[matrix1.length][matrix1[0].length];

**for**(**int** i = 0; i < matrix1.length; i++) {

**for**(**int** j = 0; j < matrix1[i].length; j++) {

matrix3[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

**for**(**int** i = 0; i < matrix3.length; i++) {

**for**(**int** j = 0; j < matrix3[i].length; j++) {

System.***out***.print( matrix3[i][j] + " ");

}

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

}

}

}

=========================================================

|| DATE: 2nd Aug, 2024 ||

/\*

\* write a program to find the sum of left

\* and right diagonal elements of a matrix.

\*/

**import** java.util.Scanner;

**public** **class** DiagonalSum {

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

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

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

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

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

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

**if**( rows != cols ) {

System.***out***.println("Diagonal can't exist for this matrix");

System.*exit*(0);

}

**int**[][] matrix = **new** **int**[rows][cols];

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

**for** (**int** j=0; j < matrix[i].length; j++) {

System.***out***.println("Element for ("+i+" , "+ j +")");

matrix[i][j]= scan.nextInt();

}

}

*findDiagonalSum*(matrix);

}

**private** **static** **void** findDiagonalSum(**int**[][] matrix) {

**int** leftSum = 0, rightSum = 0;

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

**for** (**int** j=0; j < matrix[i].length; j++) {

**if**( i == j) {

leftSum += matrix[i][j];

}

**if**( i+j == matrix.length - 1) {

rightSum += matrix[i][j];

}

}

}

System.***out***.println("Left diagonal sum : " + leftSum);

System.***out***.println("Right diagonal sum : "+ rightSum);

}

}

String handling

-----------------

* A string is a sequence of characters which are enclosed with in double quotes (“ “).
* String is a class from java.lang package.
* java.lang is the default package, which is automatically imported into a java program.
* A String object can be created in two ways.

1. string literal
2. string object

ex:

String str1 = “Hello”; //literal

String str2 = new String(“hello”); // object

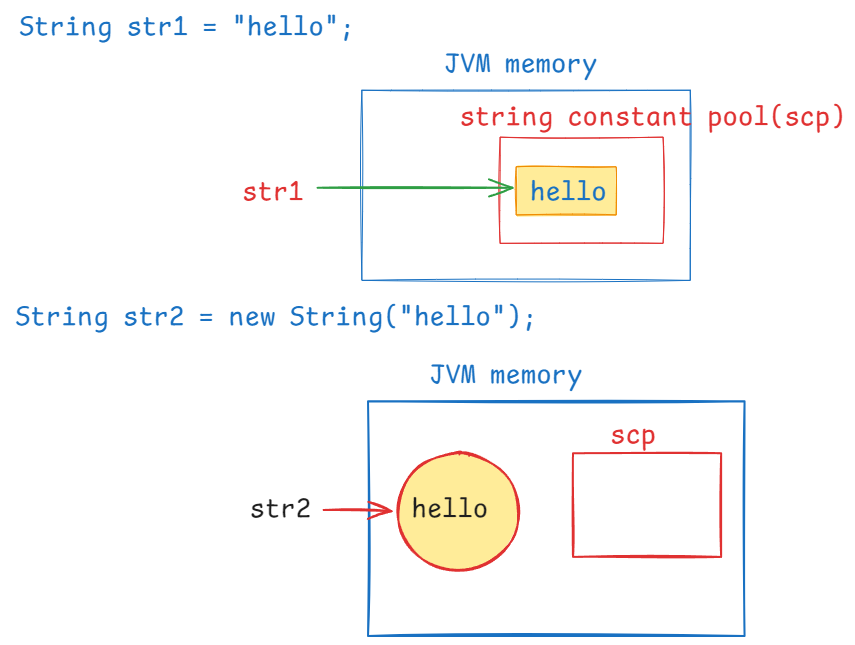
* If a string object is created in literal form, then

the object is created in string constant pool area of JVM’s memory.

* If a string object is created in object form(using new keyword) then the object is created in directly\

in JVM’s memory, but not in scp.

ex:



|| DATE: 5th Aug, 2024 ||

ex:

String str1 = "hello";

String str2 = **new** String("hello");

// == operator checks whether the two

// references are pointing to the same

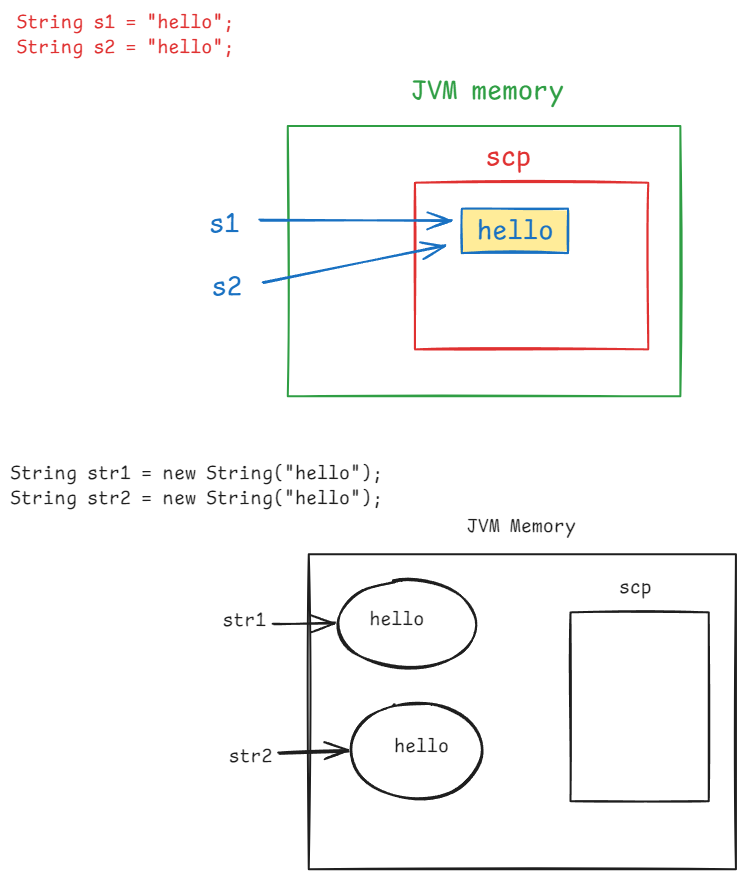
// object or not.

// if yes, returns true. Otherwise, returns false.

System.***out***.println(str1 == str2);

op: false.

ex:



String s1 = "hello";

String s2 = "hello";

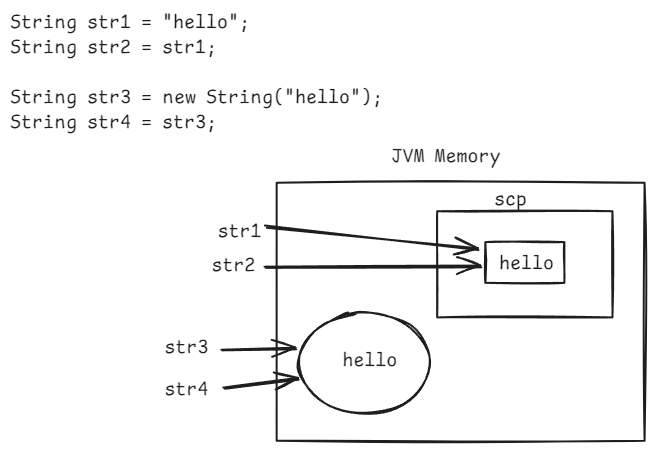
System.***out***.println(s1 == s2); // true

String str1 = **new** String("hello");

String str2 = **new** String("hello");

System.***out***.println(str1 == str2); //false

ex:



String str1 = "hello";

String str2 = str1;

String str3 = **new** String("hello");

String str4 = str3;

System.***out***.println(str1 == str2); //true

System.***out***.println(str3 == str4); //true

System.***out***.println(str1 == str3); // false

System.***out***.println(str2 == str4); //false

comparing two strings:

* Two strings can be compared in 4 ways.

1. == operator
2. equals() method
3. equalsIgnoreCase() method
4. compareTo() method.

* equals() method checks for the content is same or not. But it doesn’t check for the two variables are pointing to the same object or not.

ex:

String str1 = “hello”;

String str2 = new String(“hello”);

System.out.println(str1 == str2); //false

System.out.println( str1.equals(str2) ); //true

ex:

String s1 = “ashokit”;

String s2 = “Ashokit”;

System.out.println( s1 == s2 ); //false

System.out.println( s1.equals(s2) ); //false

* equals() method checks the content in case-sensitive manner.
* equalsIgnoreCase() method checks the content in case-insensitive manner.

System.out.println(s1.equalsIgnoreCase(s2));//true

ex:

String username = “ashokit”;

if( username.equals(“ashokit”)) {

S.o.p(“Login success”);

}

ex:

String username = null; // null value

if( username.equals(“ashokit”) ) {

S.o.p(“Login success”);

}

else {

S.o.p(“Login failed”);

}

op: NullPointerException

ex:

String username = “null”; // null string

if( username.equals(“ashokit”) ) {

S.o.p(“Login success”);

}

else {

S.o.p(“Login failed”);

}

op: Login failed

ex:

compareTo() method returns integer value.

if the value is 0 then the two strings are equal. Otherwise, not equal.

String str1 = “ashokit”;

String str2 = “ashokit”;

if(str1.compareTo(str2) == 0) {

S.o.p(“strings are equal”);

}

else {

S.o.p(“strings are not equal”);

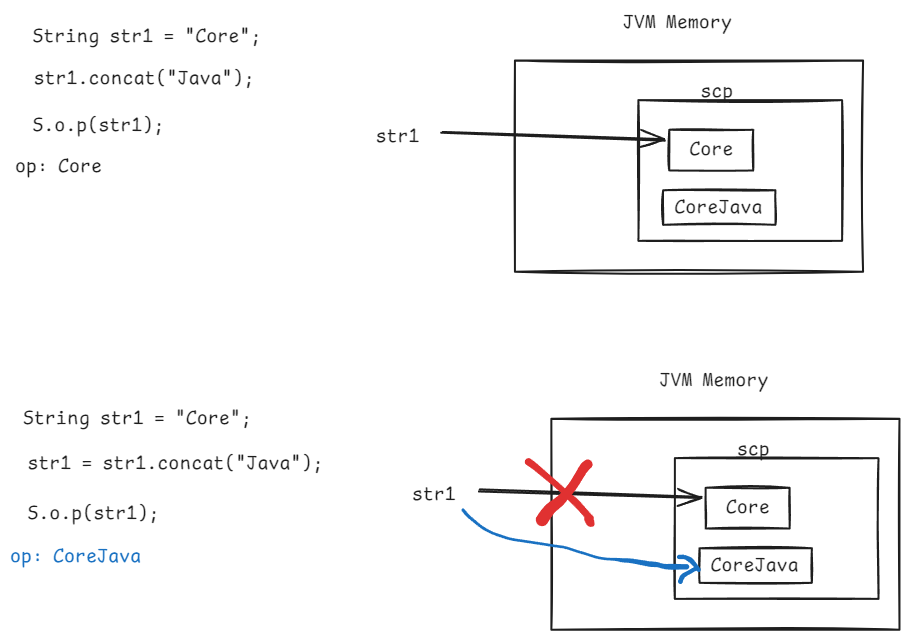
}

op: strings are equal

strings are immutable:

----------------------

* immutable object: an object which doesn’t allow to modify its content, after creating the object.
* mutable object: an object which allows to modify its content, after creating the object.
* String objects are immutable objects.



fetching a portion of a string:

* substring() method can be used to fetch a portion of a given string.
* substring() method can be used in 2 ways.

1. substring(beginIndex)
2. substring(beginIndex, endIndex)

ex:

String str1 = “Spring Framework”;

System.out.println( str1.substring(7) ); //Framework

System.out.println( str1.substring(0, 6));//Spring

==================================================

|| DATE: 6th Aug, 2024 ||

replace in string:

1. replace(oldChar, newChar)
2. replace(olString, newString)
3. replaceFirst(oldString, newString)
4. replaceAll(regex, newString)

ex1:

String str1 = “hello world”;

String str2 = str1.replace(‘l’, ‘x’);

S.o.p(str2);

op: hexxo worxd

ex2:

String str1 = “The cat sat on mat. My cat is cute”;

String str2 = str1.replace(“cat”, “dog”);

S.o.p(str2);

op: The dog sat on mat. My dog is cute

ex3:

String str1 = “The cat sat on mat. My cat is cute”;

String str2 = str1.replaceFirst(“cat”, “dog”);

S.o.p(str2);

op: The dog sat on mat. My cat is cute

ex4:

String str1 = “The cat sat on mat”;

String str2 = str1.replaceAll(“\\s”, “#”);

S.o.p(str2);

op: The#cat#sat#on#mat.

note: [\\s](file:///\\s) represents a whitespace.

ex5:

String str1 = “abc123def456ghi”;

String str2 = str1.replaceAll(“\\d”, “@”);

S.o.p(str2);

op: abc@@@def@@@ghi

note: [\\d](file:///\\d) represents a digit.

ex6:

String str1 = “example”;

String str2 = str1.replace(null, “abc”);

S.o.p(str2);

output:

NullPointerException.

splitting & joining a string:

* split() method will split the one string into multiple strings at a given character.
* join() method will join multiple strings together into a single string.

ex1:

String str1 = “The cat sat on mat”;

String str2[] = str1.split(“\\s”);

for ( String s : str2 ) {

System.out.println(s);

}

output:

The

cat

sat

on

mat

ex2:

String[] str3 = { “Spring”, “Boot”, “Microservices”);

String str4 = String.join(“--", str3);

S.o.p(str4);

output:

Spring--Boot--Microservices

charAt() and indexOf() methods:

-------------------------------

charAt(): returns character at the given index.

indexOf(): returns the index of a given character.

lastIndexOf() : returns the index of last occurrence of a given character.

* In charAt() method, if you provide an index which is greater than equals to the length of a string then StringIndexOutOfBoundsException will be thrown.
* indexOf() method, returns the index of the first occurrence of a character. If the given character doesn’t exist then result is -1.
* If you want to get the index of a character which is not the first occurrence and not the last occurrence then you call indexOf() method with character and fromIndex.

ex:

String str1 = "helollo";

System.***out***.println(str1.charAt(3));

System.***out***.println(str1.indexOf('l'));

System.***out***.println(str1.indexOf('l', 3));

System.***out***.println(str1.lastIndexOf('p'));

String str2 = "The cat sat on mat. My cat is cute";

System.***out***.println(str2.indexOf("cat"));

System.***out***.println(str2.lastIndexOf("cat"));

output:

o

2

4

-1

4

23

trim() :

This method removes the white space characters before start of the first character or after the end of last character in a string.

Ex1:

String str1 = “ Core Java”;

System.out.println( str1.trim() );

output:Core Java

Ex2:

String str = “The cat sat on mat”;

S.o.p( str.trim() );

output:

The cat sat on mat

Ex3:

String str = “ “; // a string with one white space character

S.o.p(str.length()); // 1

S.o.p(str.trim()); // no output

note: String str = “ “; //It is not an empty string

String str = “”; // It is an empty string

toCharArray():

* It converts a string into character array.

String str = “apple”;

char ch[] = str.toCharArray();

valueOf() :

* It converts a primitive type data into a String type.
* It is a static method, so we can call it with the class name.

ex:

Sop(String.valueOf(10) + String.valueOf(20));

output: 1020

ex:

Sop(String.valueOf(true) + String.valueOf(false));

output: truefalse

/\*

\* write a program to count the vowels and

\* consonents in a given String.

\*/

**import** java.util.Scanner;

**public** **class** VowelConsonentCount {

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

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

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

String str = scan.nextLine();

*findVowelConsonentCount*(str);

}

**private** **static** **void** findVowelConsonentCount(String str) {

**int** vCount = 0, cCount = 0;

str = str.toLowerCase();

**char** ch[] = str.toCharArray();

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

**if** ( ch[i] == 'a' || ch[i] == 'e' ||

ch[i] == 'i' || ch[i] == 'o' || ch[i] == 'u' ) {

vCount++;

}

**else** **if** ( !Character.*isDigit*(ch[i])) {

cCount++;

}

}

System.***out***.println("Vowels count = " + vCount);

System.***out***.println("Consoenents count = " + cCount);

}

}

|| DATE: 8-Aug-24 ||

/\*

\* write a program to check whether a

\* given string is palindrome or not.

\* ex: String str = "madam"

\* It is palindrome

\*/

**import** java.util.Scanner;

**public** **class** StringPalindromeTest {

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

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

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

String str = scan.nextLine();

**boolean** status = *isPalindrome*(str);

**if** ( status == **false** ) {

System.***out***.println( str + " : is palindrome");

}

**else** {

System.***out***.println( str + " : is not a palindrome");

}

scan.close();

}

**private** **static** **boolean** isPalindrome(String str) {

**if**( str == **null** ) {

System.***out***.println("entered string value is null");

System.*exit*(0);

}

**if** ( str.isEmpty() ) {

**return** **true**;

}

str = str.toLowerCase();

**char**[] ch = str.toCharArray();

**boolean** flag = **false**;

**for** ( **int** i = 0, j = ch.length - 1; i < ch.length / 2; i++, j--) {

**if** ( ch[i] != ch[j] ) {

flag = **true**;

**break**;

}

}

**return** flag;

}

}

/\*

\* write a program to swap the two strings

\* without using a third variable.

\*

\* ex:

\* str1 = "ashok";

\* str2 = "it";

\* S.o.p(str1); // it

\* S.o.p(str2); // ashok

\*/

**public** **class** SwapStrings {

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

String str1 = "Ashok";

String str2 = "Shekher";

str1 = str1 + str2;

str2 = str1.substring(0, str1.length() - str2.length());

str1 = str1.substring(str2.length());

System.***out***.println(str1);

System.***out***.println(str2);

}

}

/\*

\* Write a program to check whether two strings

\* are anagrams or not.

\* anagram : If one string is a permutation of another string

\* then they are called anagrams.

\* (or)

\* If the two strings have same characters, but order

\* could be any then they are called anagrams.

\* ex: str1 = "listen"

\* str2 = "silent"

\*/

**import** java.util.Arrays;

**public** **class** AnagramsTest {

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

String str1 = "ttes";

String str2 = "test";

*checkAnagrams*(str1, str2);

}

**private** **static** **void** checkAnagrams(String str1, String str2) {

**if** ( str1.length() != str2.length() ) {

System.***out***.println("The strings are not anagrams");

**return**;

}

str1 = str1.toLowerCase();

str2 = str2.toLowerCase();

**char** ch1[] = str1.toCharArray();

**char** ch2[] = str2.toCharArray();

Arrays.*sort*(ch1);

Arrays.*sort*(ch2);

**boolean** flag = **true**;

**for** ( **int** i = 0; i < ch1.length; i++ ) {

**if** ( ch1[i] != ch2[i] ) {

flag = **false**;

**break**;

}

}

**if**( flag == **true** ) {

System.***out***.println("The strings are anagrams");

}

**else** {

System.***out***.println("The strings are not anagrams");

}

}

}

|| DATE : 9-Aug-24 ||

/\*

\* write a program to remove a character

\* from a given index from a given string.

\*/

**import** java.util.Scanner;

**public** **class** Main {

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

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

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

String str = scan.nextLine();

System.***out***.println("enter the index to remove the character");

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

**if** (index >= 0 && index <= str.length() - 1 ) {

System.***out***.println( str.substring(0, index) + str.substring(index+1));

}

**else** {

System.***out***.println("index is invalid");

}

}

}

|| DATE: 10-Aug-24 ||

/\*

\* write a program to print the

\* occurrence of each character in a

\* given string.

\*

\* ex: String str = "LIRIL";

\* output: L - 2

\* I - 2

\* R - 1

\*/

import java.util.Arrays;

import java.util.Scanner;

public class MaxOccurance {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

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

String str = scan.nextLine();

findFrequency(str);

}

private static void findFrequency(String str) {

char ch[] = str.toCharArray();

Arrays.sort(ch);

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

int count = 0;

for(int j = i; j < ch.length; j++ ) {

if ( ch[i] == ch[j] ) {

count++;

i = j;

}

else {

//i = j;

break;

}

}

System.out.println( ch[i] + " - " + count);

}

}

}

/\*

\* write a program to reverse each word

\* of a given string.

\* ex:

\* String str = "This is decent";

\* output: siht si tneced

\*/

**import** java.util.Scanner;

**public** **class** MainClass {

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

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

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

String str = scan.nextLine();

*reverseWords*(str);

}

**private** **static** **void** reverseWords(String str) {

String[] words = str.split(" ");

String newStr = "";

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

String temp = words[i];

String rev ="";

**for** ( **int** j = temp.length()-1; j >= 0; j--) {

rev = rev + temp.charAt(j);

}

newStr = newStr + rev + " ";

}

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

}

}

/\*

\* write a program to check whether one string

\* is a rotation of another or not.

\* ex:

\* str1 = "ABCD"

\* str2 = "CDAB"

\* OUTPUT: str2 is a rotation of str1

\*/

**import** java.util.Scanner;

**public** **class** RotationTest {

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

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

System.***out***.println("enter string1");

String str1 = scan.nextLine();

System.***out***.println("enter string2");

String str2 = scan.nextLine();

**if**( str1.length() == str2.length() ) {

**if** ( (str1 + str1 ).indexOf(str2) != -1 ) {

System.***out***.println("str2 is a rotation of str1");

}

**else**

System.***out***.println("str2 is not a rotation of str1");

}

**else**

System.***out***.println("str2 is not a rotation of str1");

}

}

Q) why String class is given as immutable?

A) To achieve thread-safety and security.

**StringBuffer class:**

* It is a mutable and thread-safe class.
* StringBuffer objects are created directly in the JVM’s heap memory. But not in the string constant pool(Scp).
* A StringBuffer object can be created only with new keyword. We can’t create a StringBuffer object in literal form.

StringBuffer sb = “hello”; //error

StringBuffer sb = new StringBuffer(“hello”); //yes

* When a StringBuffer object is created, it allocates additional space for 16 characters to allow the modifications.
* In the above StringBuffer object, the length of the StringBuffer object is 5, and capacity of the object is 21.

S.o.p(sb.length()); // 5

S.o.p(sb.capacity()); //21

\*

\* If the length of the StringBuffer object exceeds the capacity then the capacity will be double of the length.

* Some of the useful methods of StringBuffer are,

1. append() : appends a given value to the StringBuffer
2. delete(startIndex, endIndex): deletes the characters from startindex to endindex-1.
3. insert(index, value) : inserts a value at a given index.
4. reverse(): reverse the value of a StringBuffer object.
5. substring(start,end): returns a string from start to end-1.
6. toString(): returns a string of StringBuffer.

//example, to check a string is palindrome or not

//using reverse() method of StringBuffer class.

StringBuffer sb = **new** StringBuffer("madam");

String s1 = sb.toString();

sb.reverse();

String s2 = sb.toString();

**if** ( s1.equals(s2)) {

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

}

**else** {

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

}

Q) How to compare two StringBuffer objects?

A)

ex1:

StringBuffer sb1 = new StringBuffer(“hello”);

StringBuffer sb2 = sb1;

if( sb1 == sb2 ) {

S.o.p(“both are equal”);

}

else {

S.o.p(“both are not equal”);

}

output: both are equal.

ex2:

StringBuffer sb1 = new StringBuffer(“hello”);

StringBuffer sb2 = new StringBuffer(“hello”);

if ( sb1.equals(sb2) ) {

S.o.p(“both are equal”);

}

else {

S.o.p(“both are not equal”);

}

output: both are not equal.

ex3:

StringBuffer sb1 = new StringBuffer(“hello”);

StringBuffer sb2 = new StringBuffer(“hello”);

if ( sb1.toString().equals(sb2.toString()) ) {

S.o.p(“both are equal”);

}

else {

S.o.p(“both are not equal”);

}

output: both are equal.

Q) How does a StringBuffer object is a thread-safe object?

A) The methods of StringBuffer class are synchronized methods, which means, only one thread can perform an operation at a time on a StringBuffer object. So it is a thread-safe object?

Q) What is the difference between String class and StringBuffer class?

A) String is an immutable and thread-safe object.

StringBuffer is a mutable and thread-safe object.

Q) what is the difference between StringBuffer and StringBuilder classes?

A) StringBuffer is mutable and thread-safe object.

StringBuilder is mutable, but not a thread-safe object.

Q) why do we say that StringBuilder is not a thread-safe object?

A) Because, the methods of StringBuilder class are not a

synchronized methods.