Java FullStack Development Program

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

Trainer : Shekher

(working professional)

Duration : 6 Months

Time : 11AM - 12:30PM

Fee : 30k (With Placement assistance)

(Live class + Class Notes + Backup videos(1 year validity) + Mock interviews + Material)

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

FullStack Development = Front-end Development + Back-end Development + Database + Cloud(AWS) + Real-time Tools.

Front-end Development : HTML

CSS

Java script

Bootstrap

React JS

Back-end Development : Core JAVA

Advanced JAVA(JDBC, Servlet, JSP)

Spring Framework

Spring Boot

Microservices

Database : Oracle

MySQL

MongoDB

Real-time Tools : Maven

JUnit

GitHub

Sonar

Jenkins

Log4J, etc..

Front-end :

\* It is also called User Interface(UI).

\* It is used for creating web pages for displaying

information to the user, or to read the input from the

user.

\* customers/users will interact with the Front-end

application.

Back-end :

\* It will accept the request from the front-end

application.

\* It will interact with the Database, reads the

information from the Database.

\* It will process the information and performs business

logics and provides the response to the front-end.

Database :

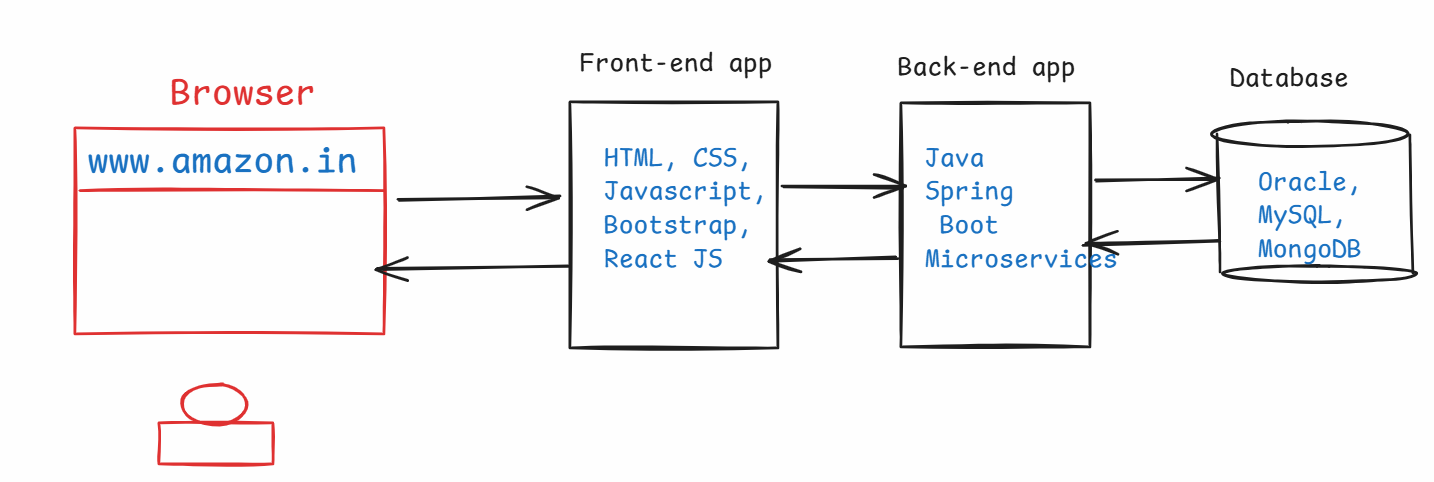
\* It will store the large amount of data in a structured

format.

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

Core JAVA

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



* Java is a high-level and open source programming language.
* Java is an Object Oriented Programming language(OOP).



* It was developed by Sun Microsystems and released in 1995.



* The initial name for the programming language was Oak, later it was renamed to Java in 1995.
* Java was created by a team of 5 persons, headed by James Gosling.
* The primary goal of Java was to develop a platform-independent language, that could run on any operating system or any device.
* Java was released to the public in 1996, with a slogan

“Write Once, Run Anywhere” (WORA).

* Java is widely used for developing Web applications, enterprise applications and mobile applications development(Android).
* Java is also used in IoT(Internet of Things) applications, game development and data analysis.
* Java is extensively used in the financial industry for building banking systems and trading applications, etc.



* In 2010, Oracle Corporation has acquired Sun Microsystems company. So, from 2010, Java software maintainence and releases are handled by Oracle corporation.



Java Editions:

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

* Java is divided into three main editions.

1. Java SE (Java Standard Edition)
2. Java EE (Java Enterprise Edition)
3. Java ME (Java Micro Edition)

* Java SE was formerly called J2SE.
* Java EE was formerly called J2EE.
* Java ME was formerly called J2ME.
* Recently, Java EE has been renamed to Jakarta EE.

Java SE:

-------

* Java SE is the core of the Java Programming language.
* Java SE, we call it as Core JAVA.
* Java SE provides the fundmental libraries for creating general-purpose applications.
* Using Java SE, we can develop Desktop applications, also called stand-alone applications.
* Desktop applications are called single user applications, and a user has to download and install the application in their computer.
* For example, Media players, calculators, Anti-virus, IDE’s, etc.

Java EE:

-------

* Java EE was built on Java SE and it provides additional libraries for developing large-scale applications.
* Java EE provides Servlets and JSP(Java Server Page), JPA(Java Persistence API), RESTful webservices, Messaging, etc..
* With Java EE, we can develop web applications and enterprise applications.
* Both web and enterprise applications runs on a server. So, you no need to install them on your computer.
* For example, social media applications like facebook or twitter are enterprise applications, and you no need to install in your computer. You can access them from your browser.
* web applications are small applications like restaurant applications or online tutorials or hotel booking applications.
* enterprise applications are large applications like banking applications, e-commerce applications, CRM applications,etc..

Java ME:

-------

* Java ME is provided for developing applications for Mobile phones, embedded systems, IoT devices
* Using Java ME, you can develop applications for smart appliances, sensors, disply modules, car infotainment systems, etc.
* For a Full Stack Java Developer, Java SE and Java EE are required, but not Java ME.

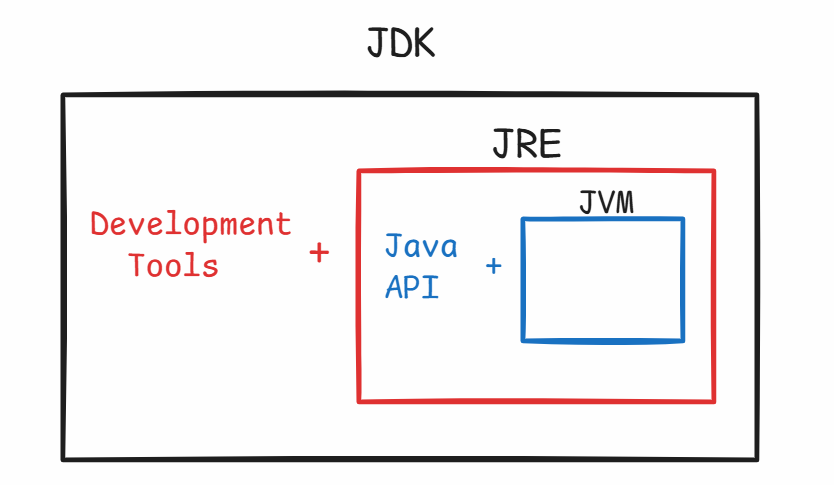
JDK :

----

* JDK stands for Java Development Kit.
* JDK is the name of the Java Software.
* To compile and to execute the Java program in a computer, we have to install the JDK software.
* JDK software provides,
* Development Tools, and
* JRE
* Development tools are, compiler, debugger, profiler, etc..
* JRE stands for Java Runtime Environment.
* JRE provides,
* Java API(Java libraries)
* JVM
* API stands for Application Programming Interface.
* Java API provides system programs(pre-defined programs) for the

programmers to develop the software applications.

* JVM stands for Java Virtual Machine
* JVM is responsible for executing the Java programs.



installing jdk-17:

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

1. visit <https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html>
2. download Windows X64 MSI Installer.
3. A file jdk-17.0.12\_Windows-X64\_bin.msi is downloaded.
4. Now double click on the downloaded file, and click on Next buttons, then close button.
5. Now, jdk-17 is successfully installed.

* To verify the Java software,

Goto C:\Program Files\Java\jdk-17

Java PATH setting:

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

1. Open Windows search and type environment
2. choose Edit system environment variables
3. click on Environment variables
4. Under system variables, choose Path variable and click Edit.
5. click on New button, and enter the value

C:\Program Files\Java\jdk-17\bin

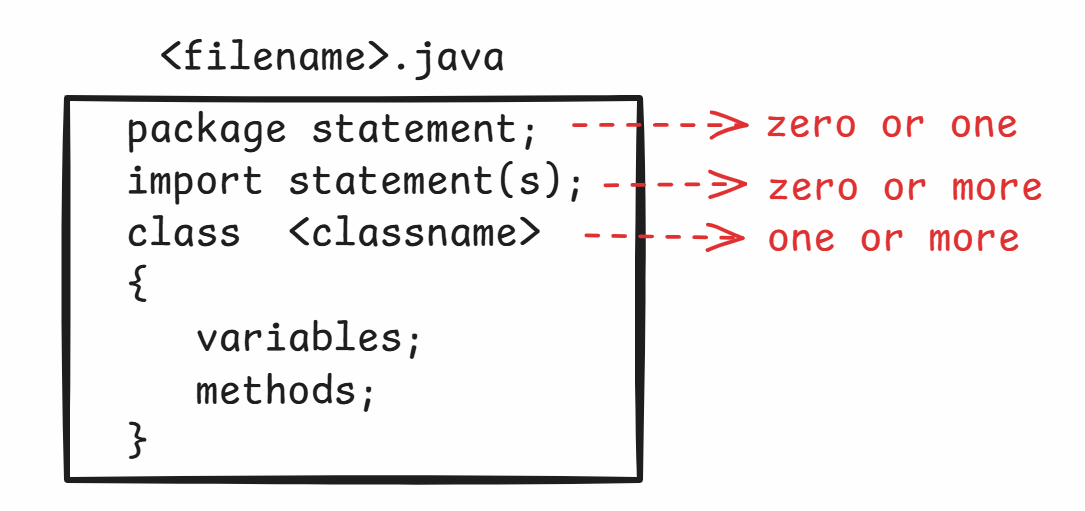
1. click on move up button, place it on top position.
2. click on OK buttons.

* To verify the path setting,

open a command prompt and run the below command.

* java --version

Basic structure of a Java program:



First Java program in Notepad:

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

1. open a notepad
2. enter the folowing code.

class MyFirstJavaProgram

{

public static void main(String[] args)

{

System.out.println("Hello, Welcome To Java");

}

}

1. Click on File menu ---> Save ---> choose D: drive ---> change save as type : All files --->enter file name: MyFirstJavaProgram.java --🡪 Save
2. open command prompt
3. change to D: drive, by typing D: then enter
4. compile the program

D:\>javac MyFirstJavaProgram.java

1. execute the program.

D:\>java MyFirstJavaProgram

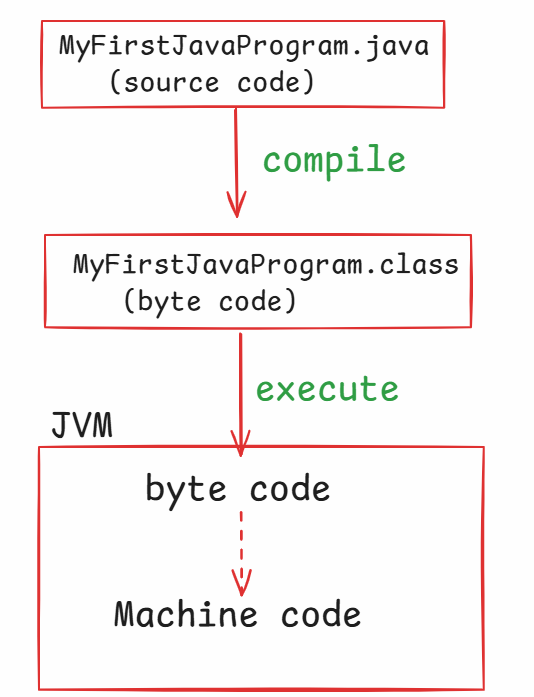
output: Hello, Welcome To Java

Q) If I want to execute the same program multiple times, How many times I need to compile the program?

A) Only one time.

Q) I have made some changes to the program. Do I need to compile the program again?

A) Yes, first we need to compile the program, then we have to execute it.



Identifiers:

----------

* Identifier is a user-defined word in a program.
* In Java programs, classnames, variablenames, method names are all identifiers.
* Follow the below rules, to define an identifier.

1. Identifier should not contain any whitespace.
2. Identifier can contain alphabets(upper/lower), digits, underscore or dollar. Other special characters are not allowed.
3. Identifier should not start with a digit.
4. Don’t use a keyword as an identifier.

examples:

My\_Class //valid

My Class //invalid

\_MyClass //valid

$\_MyClass //valid

4MyClass //invalid

\_\_MyClass //valid

$$MyClass$$ //valid

MyClass$4 //valid

break=1; //invalid

My@Class //invalid

datatypes:

----------

* Java has provided 2 categories of data types.

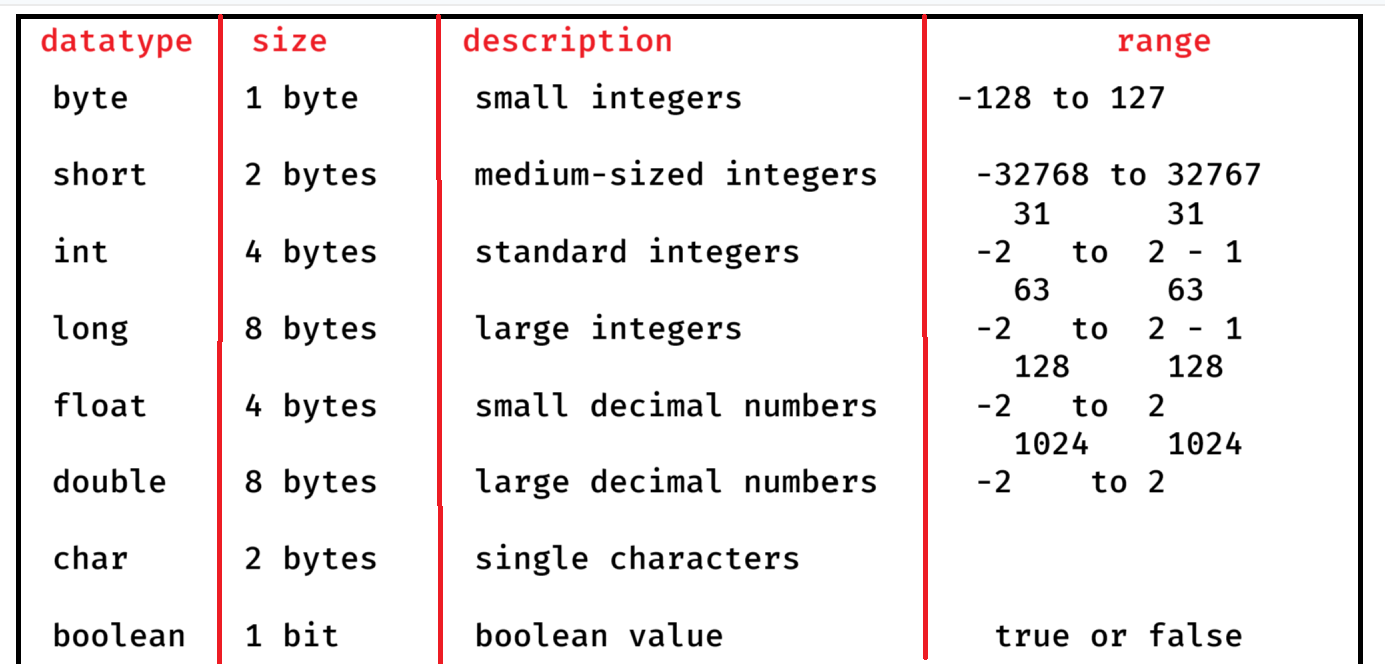
1. primitive data types.
2. reference data types.

* primitive data types are used to store the values.
* reference data types are used to store the objects.

ex:

int x = 10; //primitive data type

Employee e = new Employee(); //reference data type.





ex1:

byte a = 120; //correct

byte b = 120; //correct

byte c = a + b; //error

ex2:

short s1 = 25777; //correct

short s2 = 15100; //correct

short s3 = s1 + s2; //error

ex3:

boolean status = true; //correct

boolean flag = null; //error

ex4:

float f1 = 71.56; //error

float f2 = 71.56f; //correct

float f3 = 71.56F; //correct

Note: when you store a float value, the value must be

suffixed with either f or F.

How to declare a variable?

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

datatype variablename;

variablename = value;

(or)

datatype variablename = value;

ex:

long customerId = 19445302;

double totalDiscountApplied = 199.0;

ex:

boolean isActive;

isActive = true;

Java comments:

-----------

* comments are used to define/provide a description for the classes/variables/methods in a program.
* In Java, there are 3 types of comments.

1. single line comment
2. multiline comment
3. documentation comment

* //single line comment
* /\*

multiline comment

\*/

* /\*\*

\* documentation comment

\*/

* In real-time projects, mostly we add documentation comments. Because, they are used to prepare the project documentation.

Second Java Program:

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

/\*

\* This program is to find the sum

\* of two numbers

\*/

class Addition

{

public static void main(String[] args)

{

int a = 20;

int b = 30;

int c = a + b;

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

}

}

> javac Addition.java

> java Addition

output:

sum = 50

Q) Is the following statement valid?

int a = 20, int b = 30;

1. error

Q) Is the following statement valid?

System.out.println(“Hello”);;;

A) valid

Q) Is the following statement valid?

int a = 10, double x = 3.49;

A) valid

Installing Eclipse IDE:

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

IDE : Integrated Development Environment

* To develop the real-time Java projects, we must use IDE application.
* The popular Java IDE’s are Eclipse, STS and IntelliJ
* With the help of IDE, a programmer can develop the code easily and fastly.

1. visit eclipse.org/downloads website
2. click on Download packages link
3. Download Eclipse IDE for Enterprise Java and Web Developers.
4. eclipse-jee-2024-06-R-win32-x86\_64.zip file is downloaded.
5. Unzip the file.
6. A folder, eclipse-jee-2024-06-R-win32-x86\_64 is created.
7. open this folder and open eclipse folder inside it.
8. You have eclipse application, and you can start the eclipse IDE, by double clicking on the eclipse application.

Note:

create a shortcut on desktop for eclipse, to quickly launch the eclipse.

First Java application on Eclipse:

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

1. create a workspace directory

(Open D: drive, create a new folder 87-JFSD-Workspace)

1. start eclipse.
2. Click on Browse button, choose the workspace folder, then launch.
3. click on File menu -> New -> Project... -> Java project

-> Next -> Project Name: Application1

execution environment: choose Java SE-17

Uncheck Module checkbox.

* Next -> Finish.

1. Expand Application1 -> Right click on src folder -> New -> class -> package : com.example

Name : Multiplication

* choose public static void main check box.
* Next -> Finish

1. Write the below code.

**package** com.example;

**public** **class** Multiplication {

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

**float** x1 = 0.9f;

**float** x2 = 0.4F;

**float** x3 = x1 \* x2;

System.***out***.println(" x3 = " + x3);

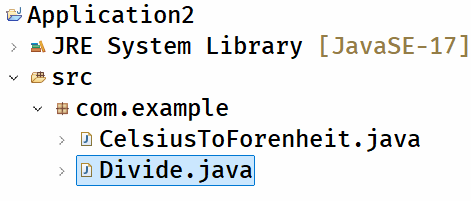
}

}

1. Right click on Multiplication.java -> RunAs -> Java Application

output:

x3 = 0.35999998



Divide.java

---------

/\*\*

\* This program reads two input values

\* from the user and calculates division.

\*/

**package** com.example;

**import** java.util.Scanner;

**public** **class** Divide {

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

//create Scanner class object

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

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

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

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

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

**int** z = x / y;

System.***out***.println("Division = " + z);

scan.close();

}

}

CelsiusToForenheit.java

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

/\*\*

\* This program reads the input in celsius

\* and converts it into forenheit.

\* Hint:

\* formula:

\* f = (c \* 9/5) + 32;

\*/

**package** com.example;

**import** java.util.Scanner;

**public** **class** CelsiusToForenheit {

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

//create Scanner class object

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

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

**double** c = scan.nextDouble();

//convert celsius to forenheit

**double** f = (c \* 9 / 5) + 32;

System.***out***.println("The given celsius : " + c);

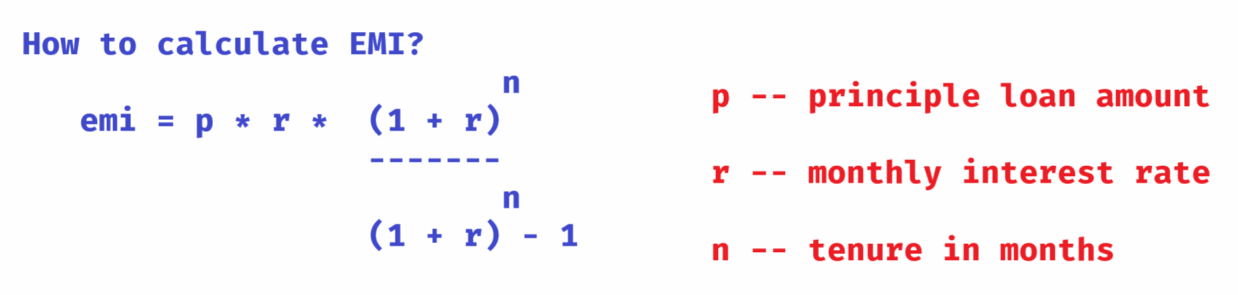
System.***out***.println("The forenheit value : " + f);

//optional

scan.close();

}

}



/\*\*

\* This program reads the principle loan amount,

\* annual interest rate and tenure in years as input,

\* then calculates EMI, displays the output.

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** Solution {

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

//create Scanner class object

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

System.***out***.println("Enter principle loan amount");

**double** p = scan.nextDouble();

System.***out***.println("Enter annual interest rate");

**double** pa = scan.nextDouble();

System.***out***.println("Enter tenure in years");

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

//convert annual interest rate to monthly interest rate

**double** r = pa / 12 / 100;

//convert tenure from years to months

**int** n = years \* 12;

//calculate 1+r power n value

**double** x = Math.*pow*( 1 + r, n );

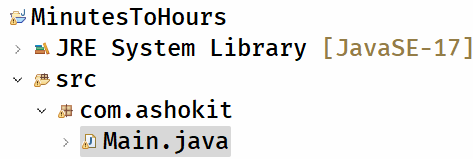
//calculate emi

**double** emi = p \* r \* x / (x - 1) ;

System.***out***.println("EMI = " + emi);

}

}



Main.java

--------

/\*\*

\* This program reads minutes as input and

\* converts into hours and minutes.

\* ex:

\* input: 135

\* output 2 hours and 15 minutes

\*/

**package** com.ashokit;

**import** java.util.Scanner;

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

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

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

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

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

//convert minutes into hours

**int** hours = minutes / 60;

//find the remaining minutes

**int** remainingMinutes = minutes % 60;

System.***out***.println( hours + " hours and " + remainingMinutes + " minutes");

}

}

Type casting:

-----------

* converting a value from one data type to another data type is called type casting.
* Type casting is 2 types.

1. implicit type casting/upcasting / widening
2. explicit type casting/downcasting / narrowing.

* In implicit type casting, a value from lower data type will be converted to higher data type, or same data type.
* implicit type casting will be automatically done by the system.
* In explicit type casting, a value from higher data type will be converted to the lower data type.
* explicit type casting should be done explicitly by providing the target data type with the paranthesis.

ex1:

int x1 = 30;

long x2 = x1; //implicit

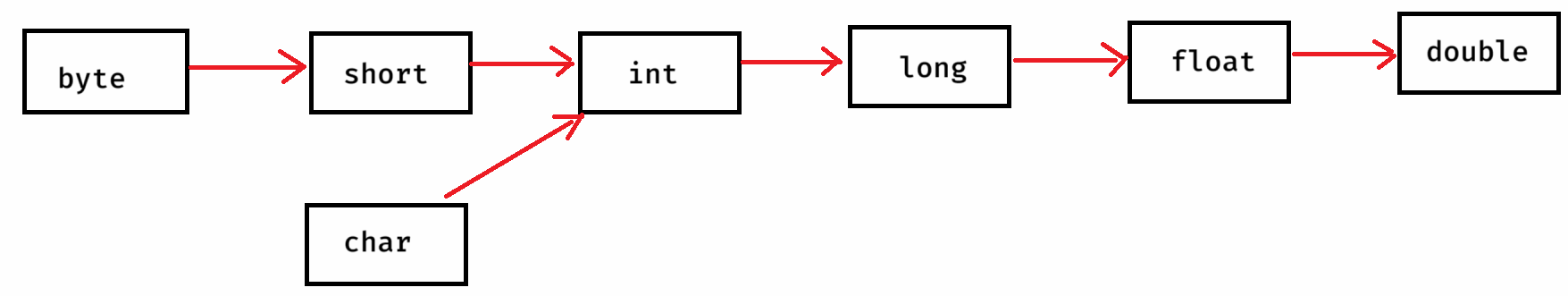
ex2:

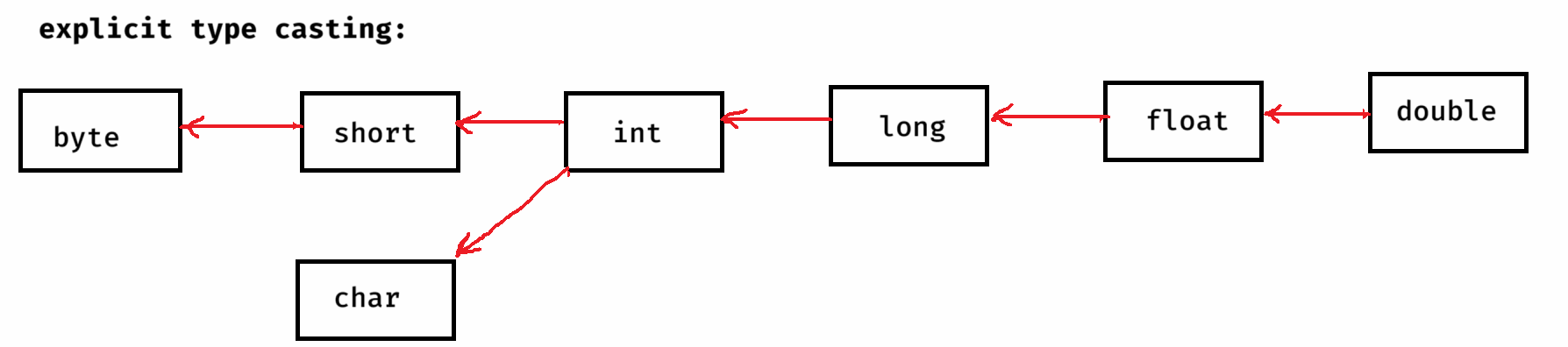
long x3 = 23879;

int x4 = x3; //error

int x4 = (int) x3; //explicit

implicit type casting:







Note:

* boolean data type cannot be converted to any other data type, and also vice-versa.

examples:

char ch = ‘A’;

int i = ch; //implicit

S.o.p(i); //output: 65

int k = 100;

char c = k; //error

char c = (char) k; //explicit

S.o.p(c); //output: d

long x = 125;

float y = x; // implicit

S.o.p(y); //output: 125.0

example code:



Solution.java

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

/\*\*

\* This program demonstrates type casting.

\*/

**public** **class** Solution {

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

**short** s =1024;

**float** f = s; //implicit

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

**double** d = 397.421;

**int** x = (**int**) d; //explicit

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

**byte** k = 100;

**char** ch = (**char**) k; //explicit

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

**char** ch1 = 'A';

**byte** b = (**byte**) ch1; //explicit

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

}

}

output:

f = 1024.0

x = 397

d

65

operators:

-> operator is a symbol, which performs a task.

-> operators are 3 categories.

1. unary operators

2. binary operators

3. ternary operator.

-> unary operator performs a task, on a single operand

-> binary operator performs a task, on two operands.

-> ternary operator performs a task. on three operands.

Unary operators:

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

1. increment operator (++)
2. decrement operator (--)

* increment operator increments a value by 1
* decrement operator decrements a value by 1

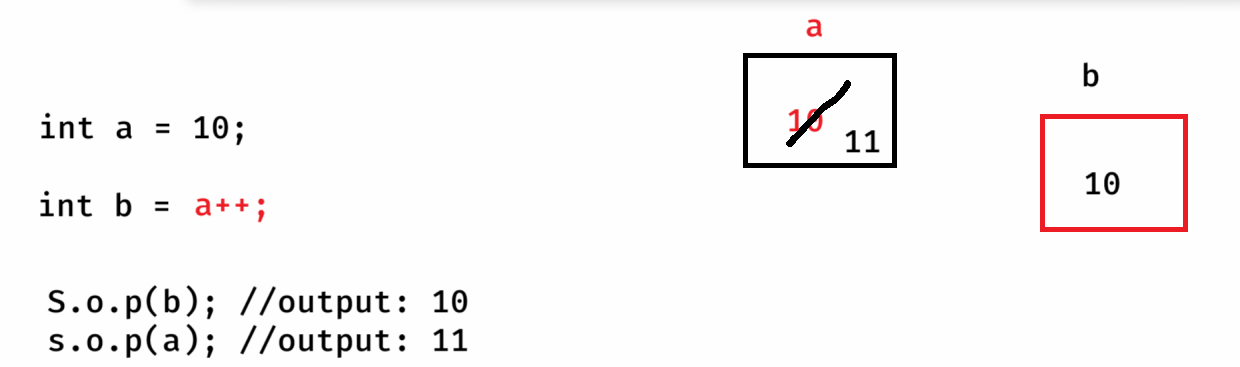
value++ : post increment(increment but after the use)

++value : pre increment (increment but before the use)

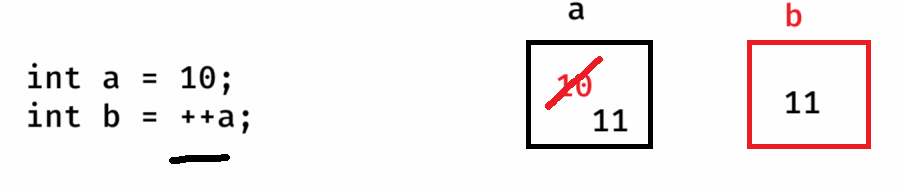
value-- : post decrement(decrement but after the use)

--value : pre decrement(decrement but before the use)

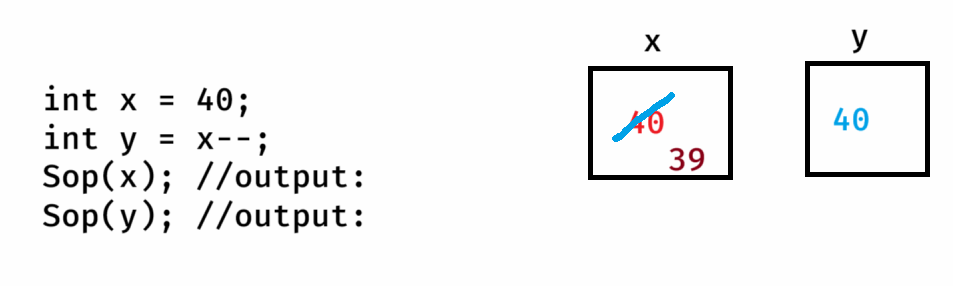
ex1:



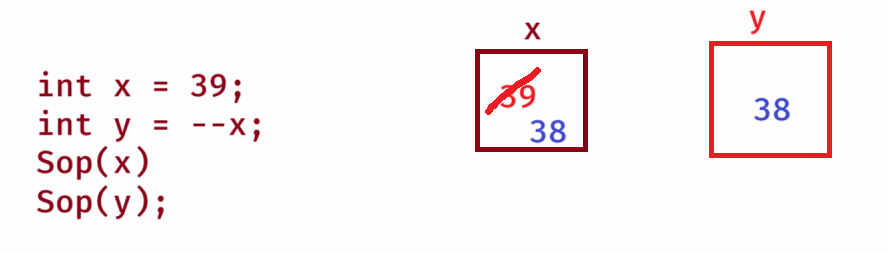
ex2:



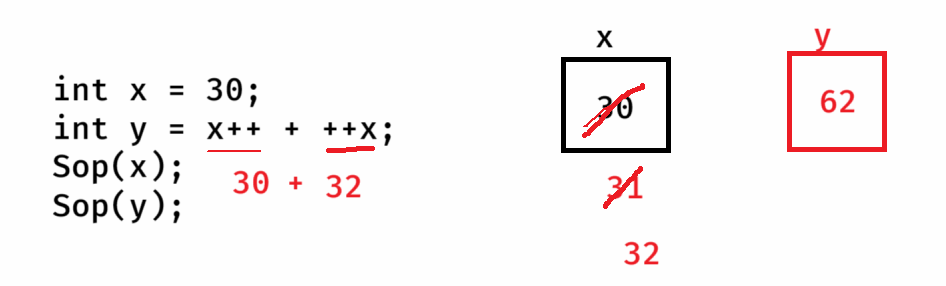
ex3:



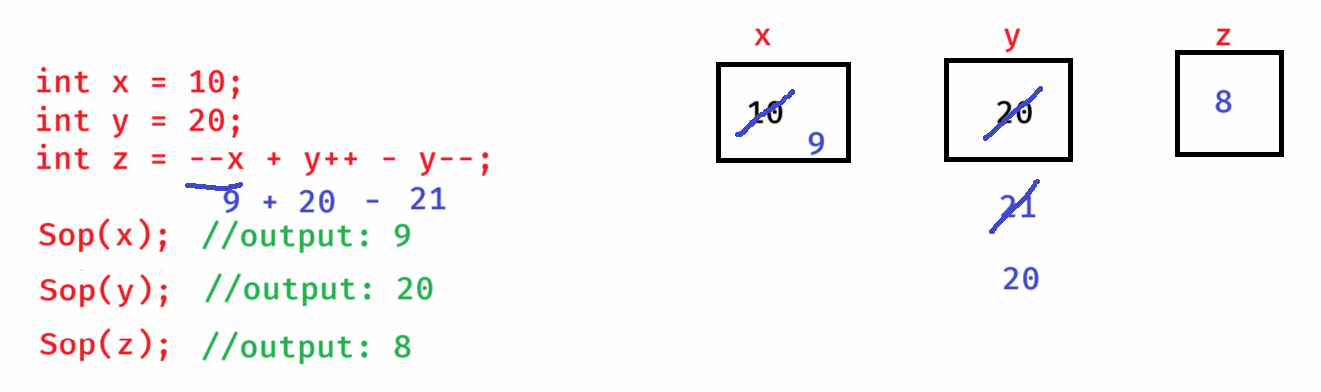
ex4:



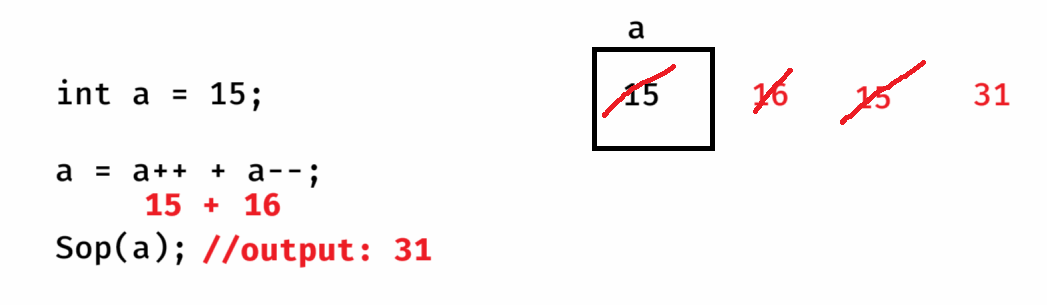
ex5:



ex6:



ex7:



Binary operators:

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

1. Arithmetic operators
2. relational operators
3. logical operators
4. assignment operators
5. bitwise operators

Arithmetic operators:

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

+ -- addition

* -- substraction

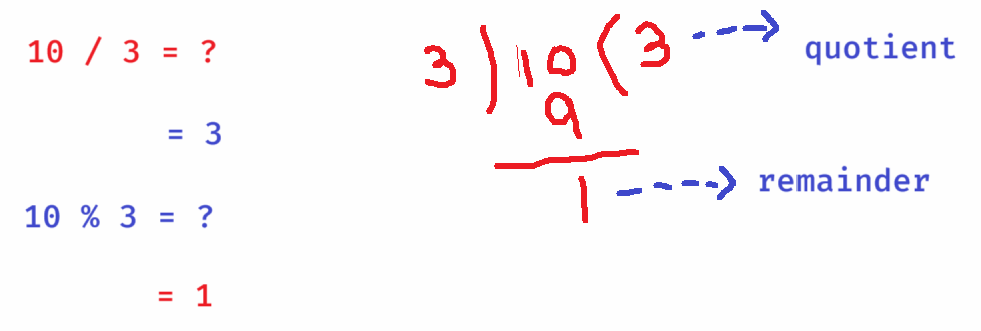
\* -- multiplication

/ -- division

% -- modulus

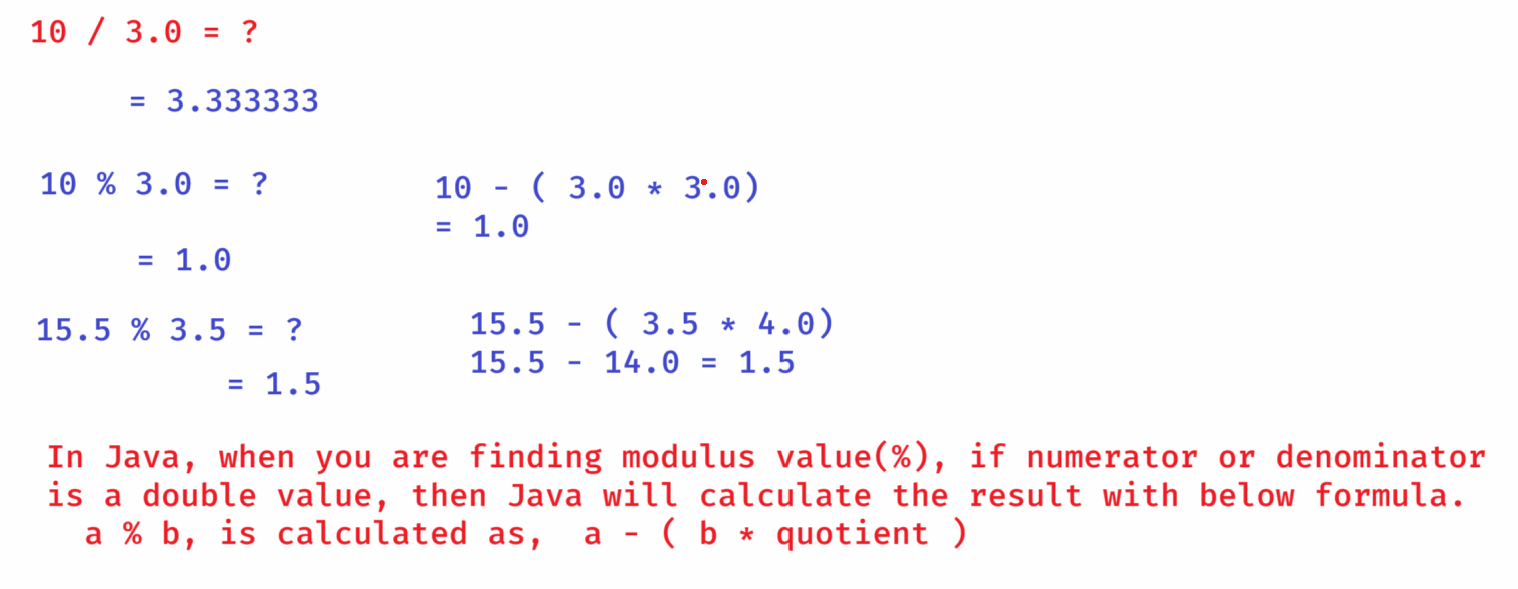
* division operator returns the quotient value and modulus operator returns the remainder value.

ex1:



* In Java, when you divide an integer value with another integer value then the result will also be the integer value only.

ex2



Relational operators:

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

< -- less than

> -- greater than

<= -- less than or equals

>= -- greater than or equals

!= -- not equals to

== -- equals to

* relational operators returns a boolean value(either true or false).

ex1:

10 < 3 // output: false

5 > 10 > 9 //output: error

ex2:

String str = null;

str != null // output: false

str == null // output: true

logical operators:

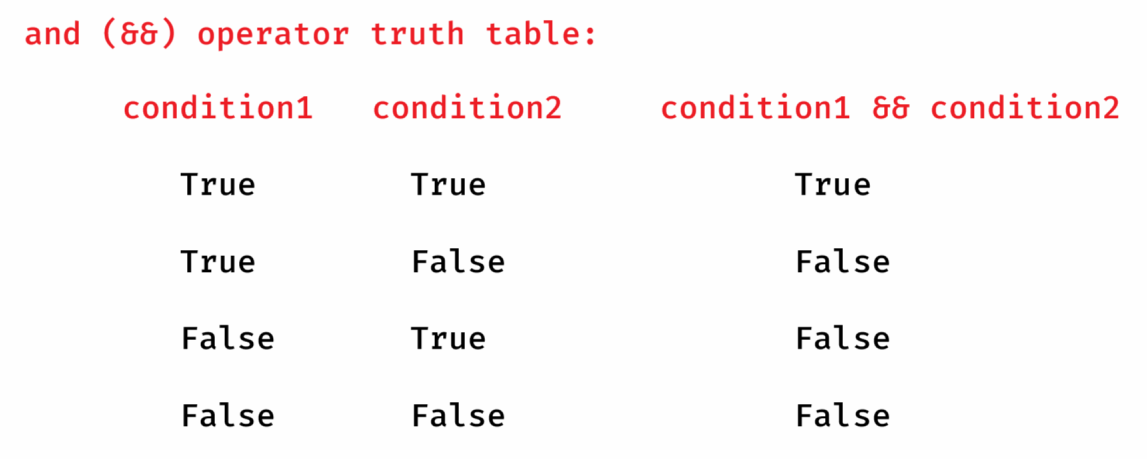
----------------

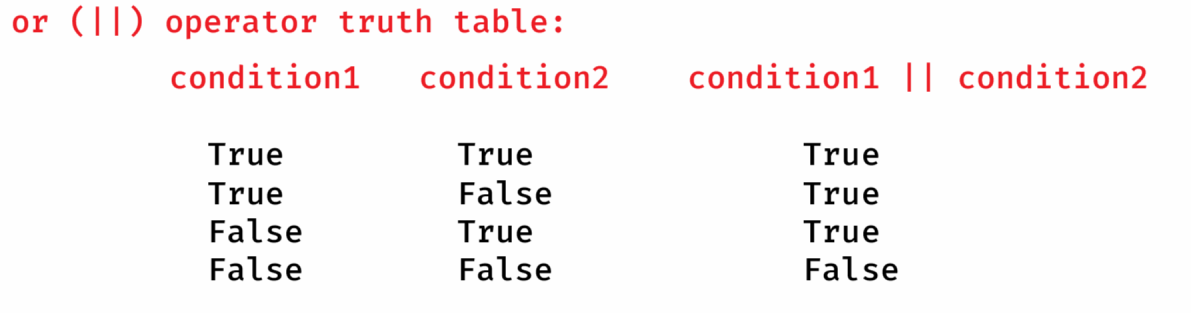
**&& -- and operator**

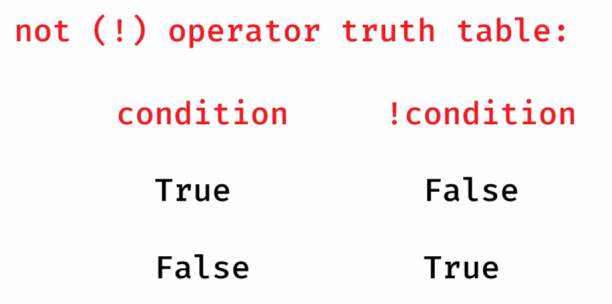
**|| -- or operator**

**! -- not operator**

* Logical operators are used to combine the two conditions together into a single condition.
* and(&&) operator returns true, if both the conditions are true. Otherwise, returns false.
* or(||) operator returns true, if any one condition is true. Otherwise, returns false.
* not(!) operator is used to flip the value. If the condition is true, then it flips to false. If the condition is false then it flips to true.







ex1:

int x = 10;

if( x++ >= 11 && x-- < 11)

{

x = x + 5;

}

System.out.println(x);

output:

11

ex2:

int a = 5;

if( ++a > 5 || a++ > 10 )

{

a = a + 5;

}

System.out.println(a);

output:

11

Note:

1. and(&&) operator executes the second condition, if the first condition is true.
2. or(||) operator executes the second condition, if the first condition is false.

ex3:

int x = 1, y = 2;

if( x++ + y++ < 5 && x-- + y-- > 4)

{

x++;

y++;

}

System.out.println(x);

System.out.println(y);

output:

2

3

ex5:

boolean b = false;

int k = 10;

if ( !b || b )

{

k = 20;

}

System.out.println(k);

output:

20

Assignment operators:

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

= assign

+= add then assign

-= substract then assign

\*= multiply then assign

/= divide then assign

%= modulus then assign

ex1:

int k = 5;

k += 6;

S.o.p(k);

output:

11

ex2:

int a = 40;

a %= 5;

S.o.p(a);

output:

0

ex3:

int c1 = 30;

int c2 = 36;

c1 += c2;

S.o.p(c1);

output:

66

Bitwise operators:

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

* Bitwise operators will work on the bits of the operands.
* Bitiwse operators are used in the development of cryptography applications, VLSI applications, Device driver applications, etc..
* Bitwise operators are not much used in the development of Web applications or the Enterprise applications.

**& -- Bitwise AND operator**

**| -- Bitwise OR operator**

**^ -- Bitwise XOR operator**

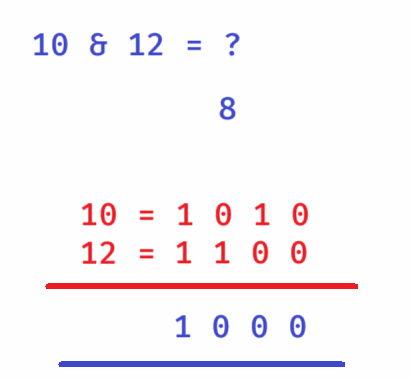
**<< -- Left shift operator**

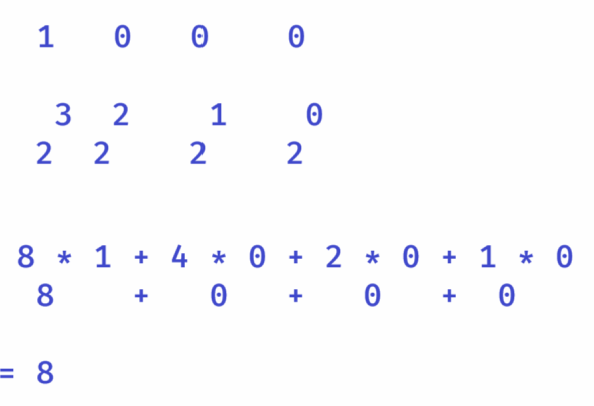
**>> -- Right shift operator**

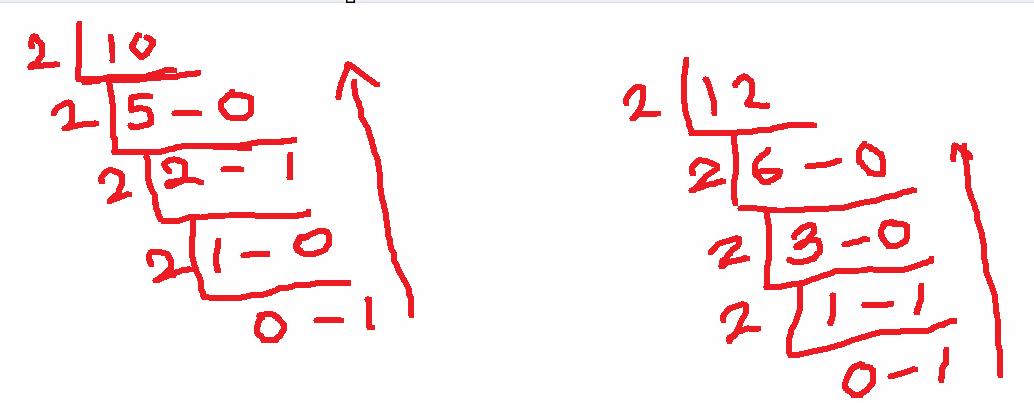
**Bitwise AND(&):**

\* This Bitwise AND operator returns result as a bit 1, if the corresponding bits of the operands are 1. Otherwise, returns 0.

ex:







ex2:

15 & 20 = ?

4

15 = 0 1 1 1 1

20 = 1 0 1 0 0

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

0 0 1 0 0

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

Bitwise OR(|):

* This operator returns a bit 1, if any of the corresponding bits are 1. Otherwise returns 0.

ex:

10 | 12 = ?

10 = 1 0 1 0

12 = 1 1 0 0

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

1 1 1 0

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

1. 1 1 0 = 14

3 2 1 0

2 2 2 2

Bitwise XOR(^):

----------

\* This operator returns a bit 1, if the corresponding bits are opposite. Otherwise returns 0.

ex:

10 ^ 12 = ?

10 = 1 0 1 0

12 = 1 1 0 0

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

0 1 1 0 = 6

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

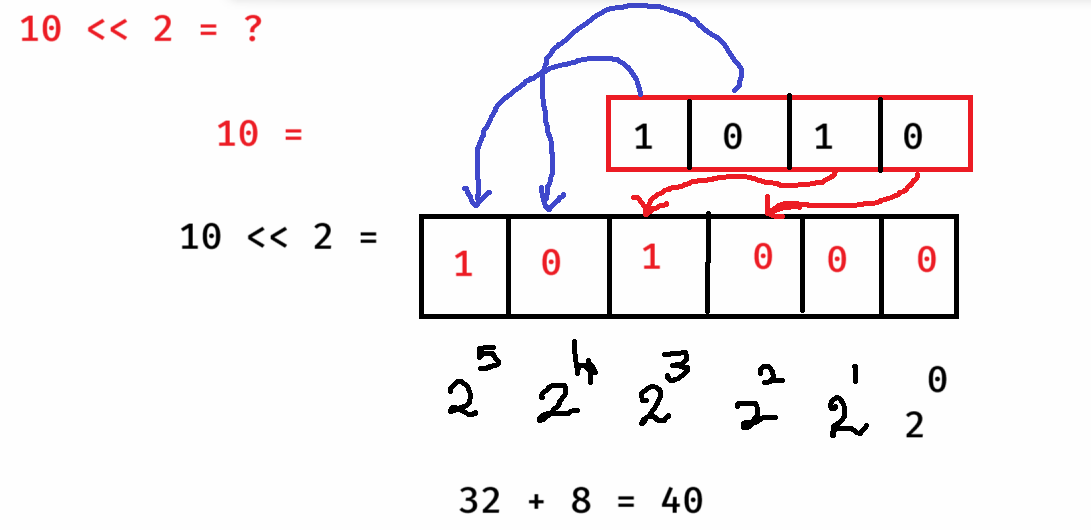
Left Shift operator:

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

\* This operator will shift the bits of the left operand to the left side by the given positions of right operand.

\* The empty bits generated at right are filled with zero.

ex:

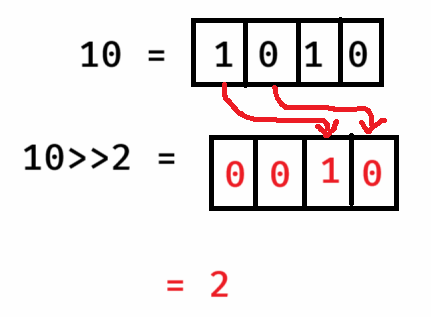


right shift operator:

* This operator will shift the bits of the left operand to the right side by the given positions of the right operand.

ex:

10 >> 2 = ?



Ternary operator(? : )

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

syntax:

variable = condition ? value1 : value2;

* If the condition is true, then value1 will be returned. Otherwise, value2 will be returned.

ex:

int k = 10 > 5 ? 8 : 4;

S.o.p(k);

output: 8

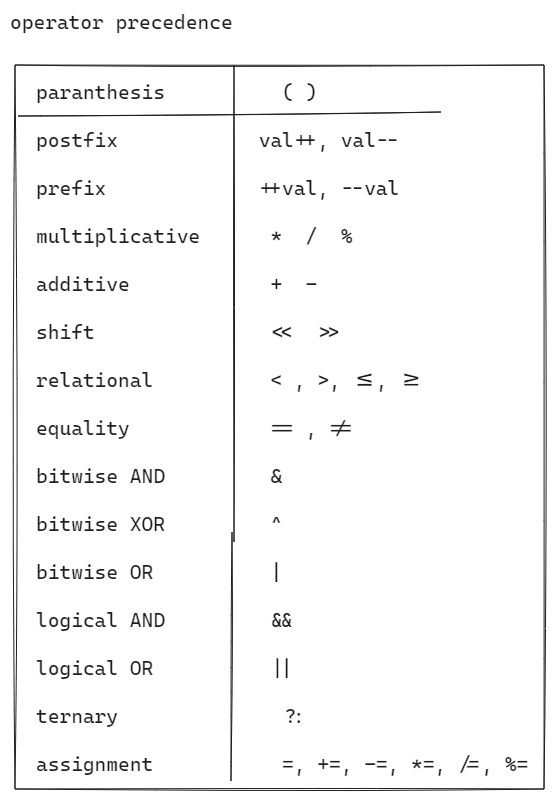
ex:

int a = 19, b = 14;

int k = a > b ? a+b : a-b;

S.o.p(k);

output: 33



ex1:

int x = 3 + 4 \* 9 / 6 – (2 +7);

S.o.p(x);

output:

int x = 3 + 4 \* 9 / 6 – 9;

int x = 3 + 36 / 6 – 9;

int x = 3 + 6 – 9;

int x = 9 – 9;

x = 0;

ex2:

int x = **6 – 7 \* 8 + 9 / 5 – 6 % 3 – 7 / 2 - 3;**

S.o.p(x);

output:

int x = 6 – 56 + 9 / 5 – 6 % 3 – 7 / 2 – 3;

x = 6 – 56 + 1 – 6 % 3 – 7 / 2 – 3;

x = 6 – 56 + 1 – 0 – 7 / 2 – 3;

x = 6 – 56 + 1 – 0 – 3 – 3;

x = -50 + 1 – 0 – 3 – 3;

x = -49 – 0 – 3 – 3;

x = -55

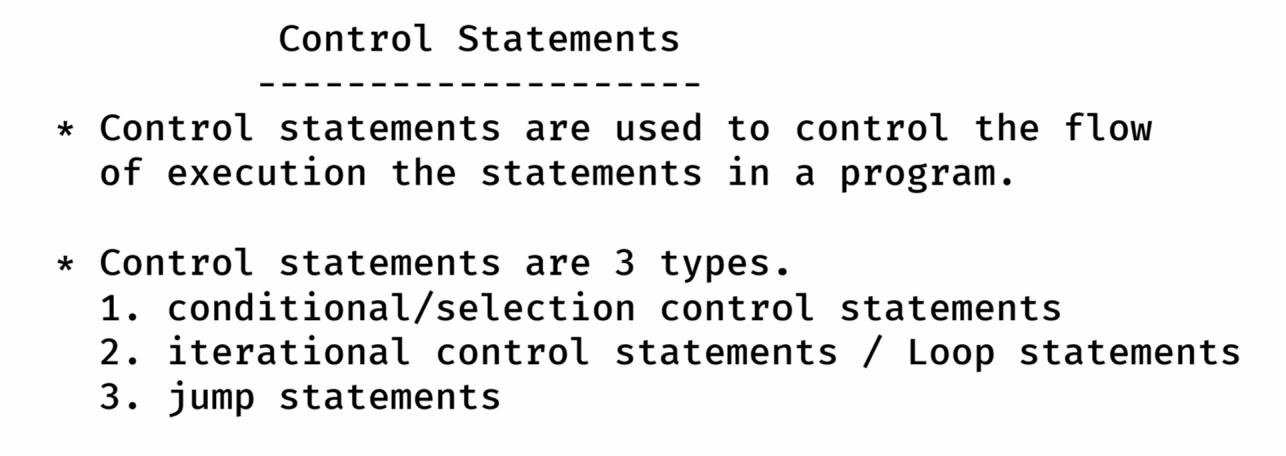
ex3:

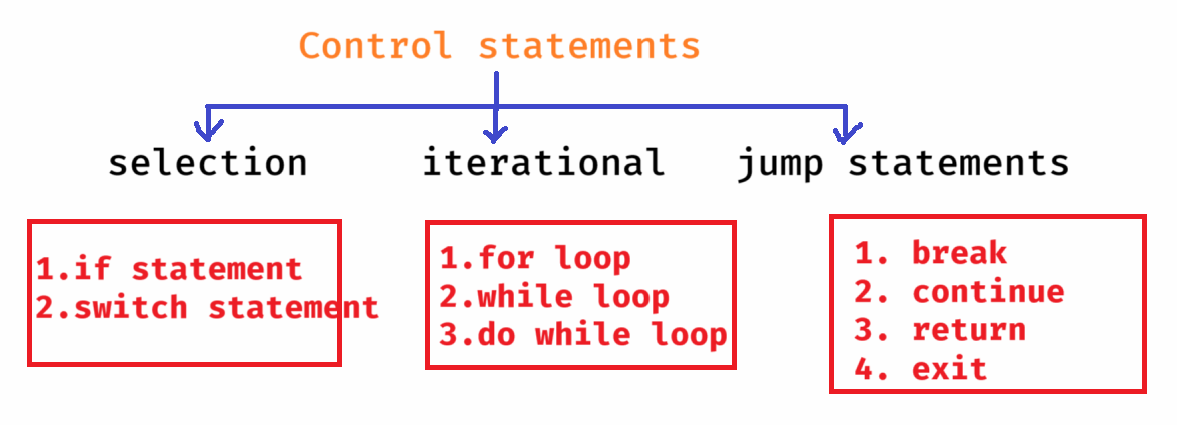
int x = **3 \* 5 – 7 + 7 \* 7 – 7 / 7 + 7 \* 7 % 7;**

S.o.p(x);

output:

56

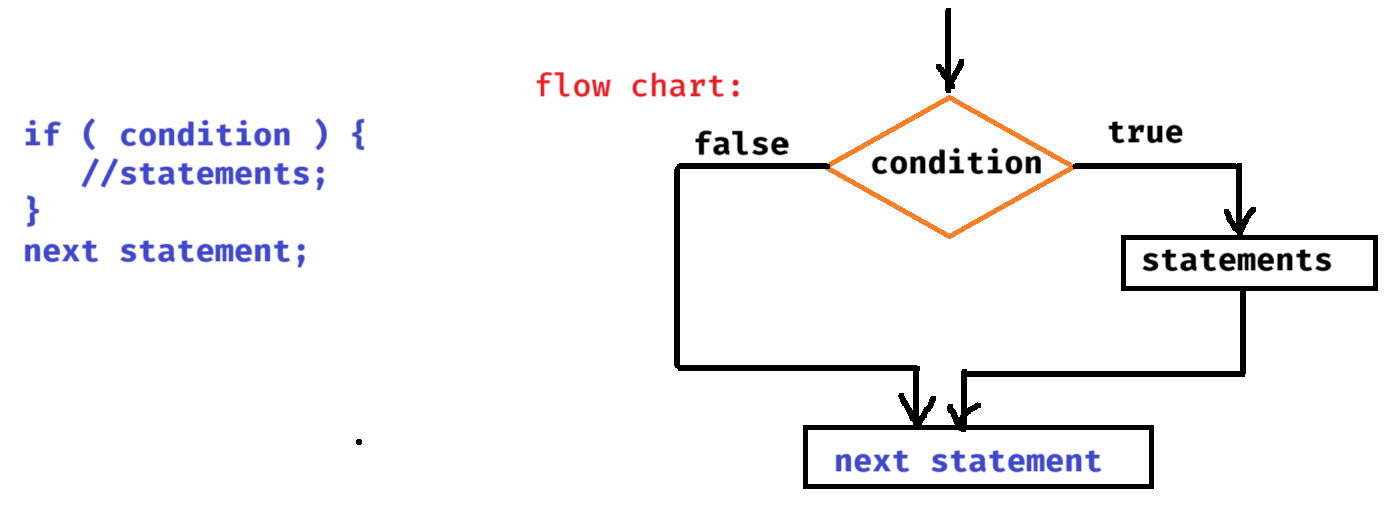




if statement

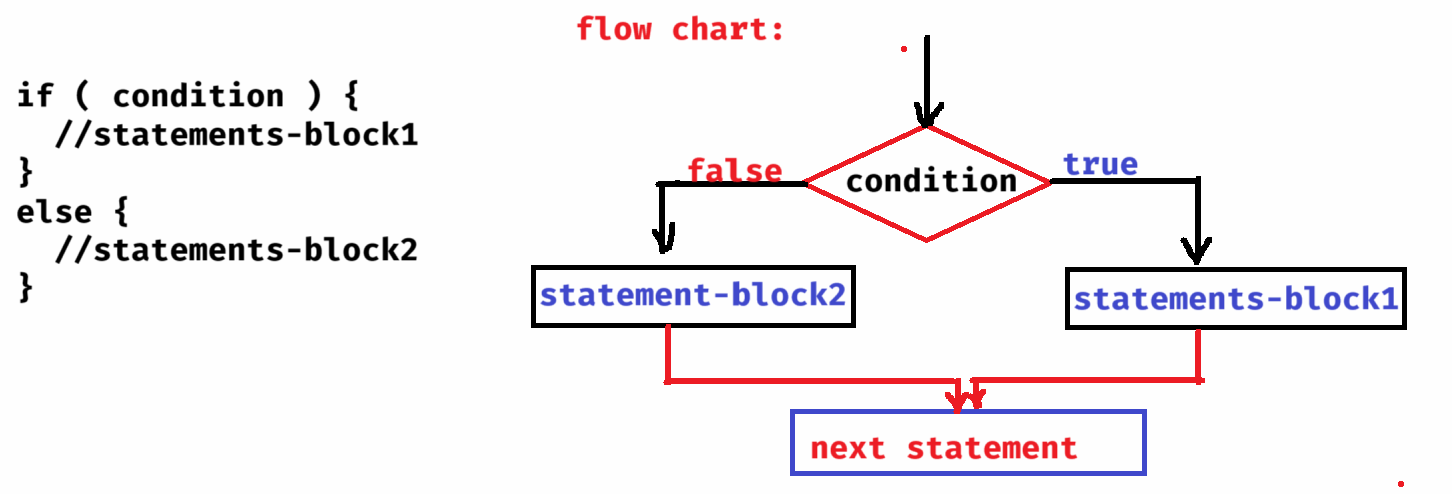
1. simple if statement
2. if else statement
3. if else ladder statement
4. nested if statement

simple if syntax:

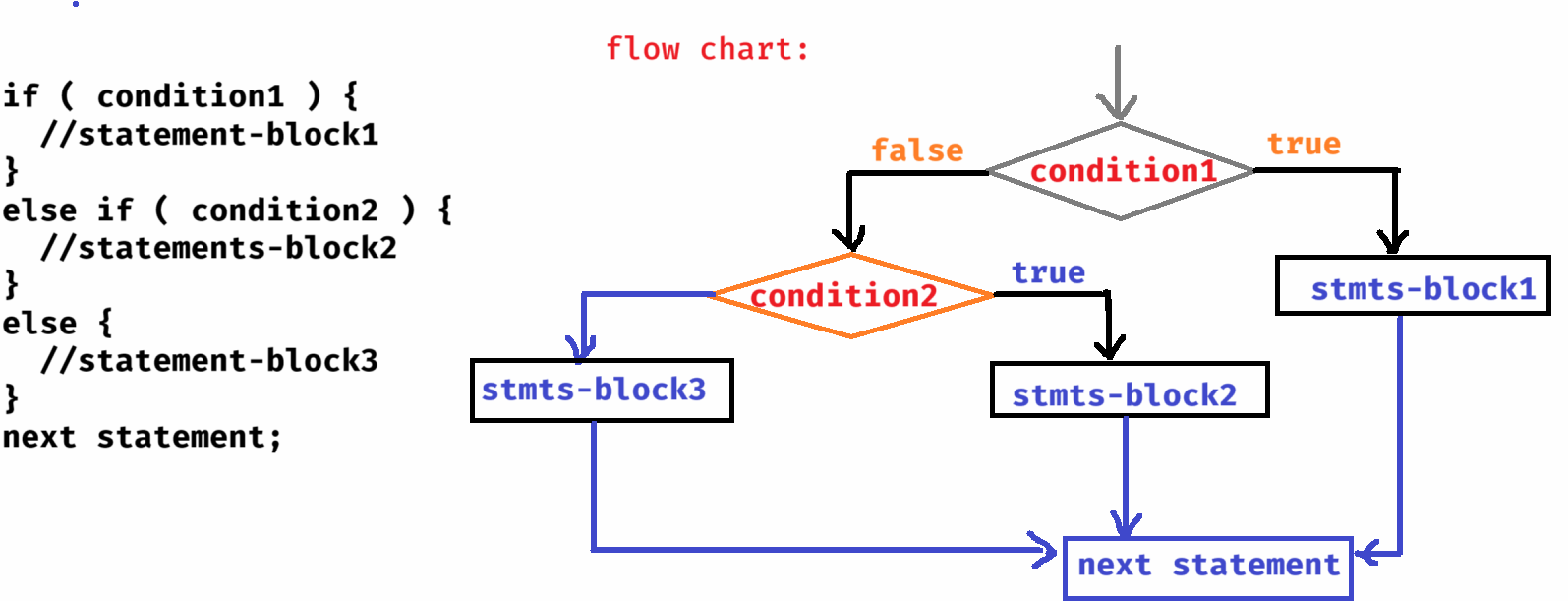


if else syntax:

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

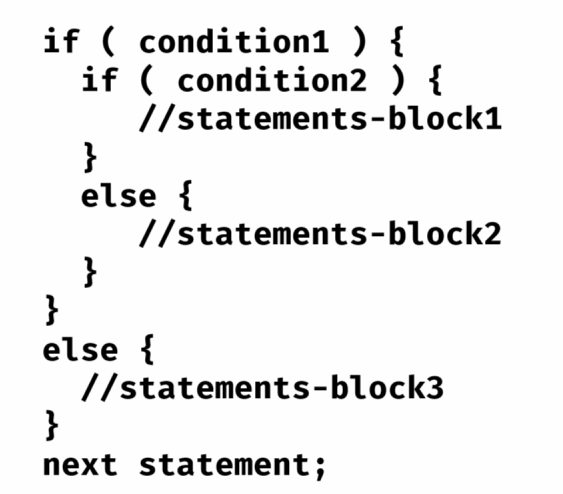


if else ladder syntax:



nested if syntax:

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



/\*

\* This program reads two input values and then

\* if a > b , prints a+b

\* if a < b, prints a\*b

\* otherwise, prints a-b

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** Demo {

**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();

**if**( a > b ) {

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

}

**else** **if** ( a < b ) {

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

}

**else** {

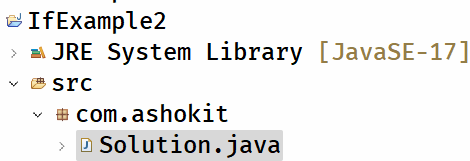
System.***out***.println(a - b);

}

scan.close();

}

}



Solution.java

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

/\*\*

\* This program reads an integer value from the user and

\* performs the following.

\* -> if the given number is divisible by 3 then displays "Zip"

\* -> if the given number is divisible by 5 then displays "Zap"

\* -> if the given number is divisible by 3 and 5 then displays "Rar"

\* -> otherwise, displays "Jar"

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** Solution {

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

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

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

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

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

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

}

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

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

}

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

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

}

**else** {

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

}

scan.close();

}

}



BiggestOfThree.java

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

/\*\*

\* This program finds the biggest of

\* the 3 numbers.

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** BiggestOfThree {

**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();

**if**( a > b && a > c )

System.***out***.println("Biggest = " + a);

**else** **if**( b > c )

System.***out***.println("Biggest = " + b);

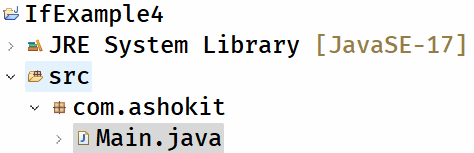
**else**

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

scan.close();

}

}



Main.java

-------

/\*\*

\* Write a program to read the distance in kms as input and

\* calculate the delivery fee as below.

\* -> For first 3 kms, free delivery

\* -> For next 3 kms, Rs 15 per km.

\* -> For the remaining kms, Rs 20 per km.

\*/

**package** com.ashokit;

**import** java.util.Scanner;

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

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

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

System.***out***.println("Enter the distance in kms");

**double** distance = scan.nextDouble();

**double** deliveryFee = 0;

**if**( distance <= 3 ) {

deliveryFee = 0;

}

**else** **if**( distance <= 6 ) {

deliveryFee = (distance - 3) \* 15;

}

**else** {

deliveryFee = 3 \* 15 + (distance - 6) \* 20;

}

System.***out***.println("Delivery Fee = " + deliveryFee);

scan.close();

}

}

Question:

------

Take 3 numbers as input, and do the following.

If last number is 7 then find the product of the two numbers.

If middle number is 7 then find the division of the two numbers.

If first number is 7 then find the modulus of the two numbers

If no number is 7 then display -1.

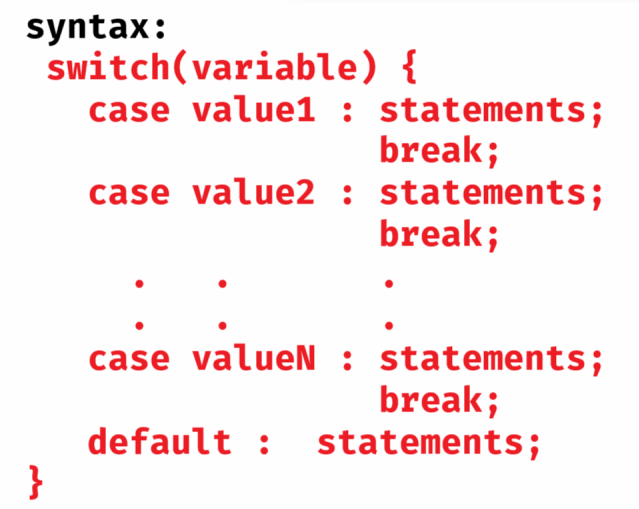
Question:

-------

Take a year as input and check whether it is a leap year or not.

switch statement

* If you want to write multiple conditions on the same variable and the condition is equals(==) operator then instead of using if else ladder, you can use switch statement.
* With switch statement, you can make the code more readable and more understandable.



* In switch statement, the variable must be either integer type(byte/short/int/long), or char data type or String type only.
* The case values are also must be either integer type or char type or String type.
* writing the default case is optional.
* If you don’t write break in a case, then the control will execute the following cases also, until break statement occurs.

Q) can we use float/double/boolean variable in a switch statement?

A) No.

ex1:

int a = 2;

switch( a + 1 ) {

case 1: Sop(“One”);

break;

case 2: Sop(“Two”);

break;

case 3: Sop(“Three”);

case 4: Sop(“Four”);

default : Sop(“Zero”);

}

output:

Three

Four

Zero

ex2:

String signal = “blue”;

switch(signal) {

default : Sop(“RUN”);

case “red” : Sop(“STOP”);

case “yellow”: Sop(“WAIT”);

case “green”: Sop(“START”);

}

output:

RUN

STOP

WAIT

START

EX3:

char ch = ‘w’;

switch(ch) {

case ‘a’:

case ‘e’:

case ‘i’:

case ‘o’:

case ‘u’: Sop(“Vowel”);

break;

default : Sop(“Consonant”);

}

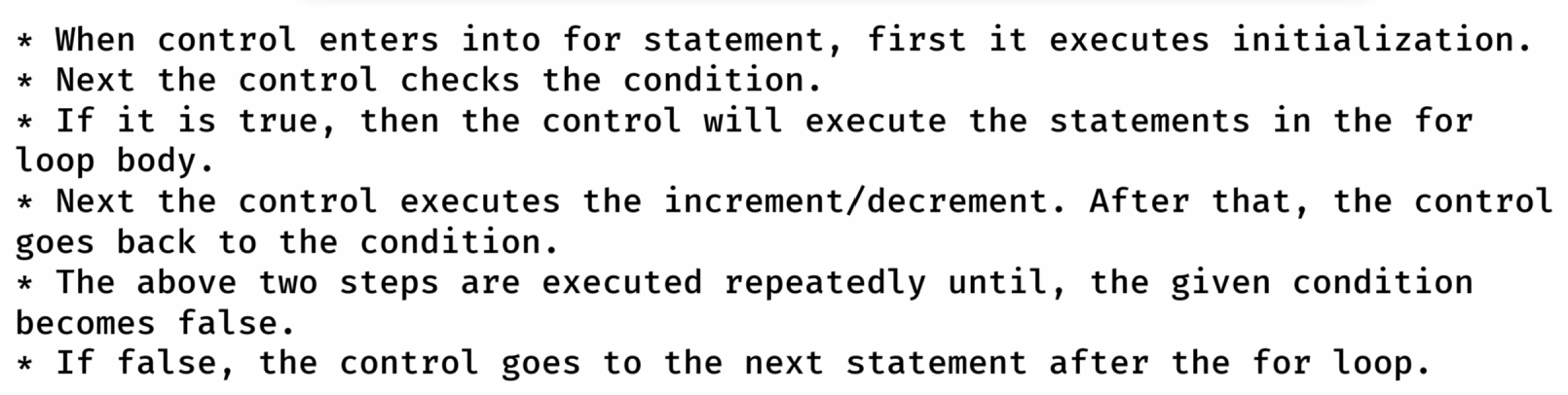
output:

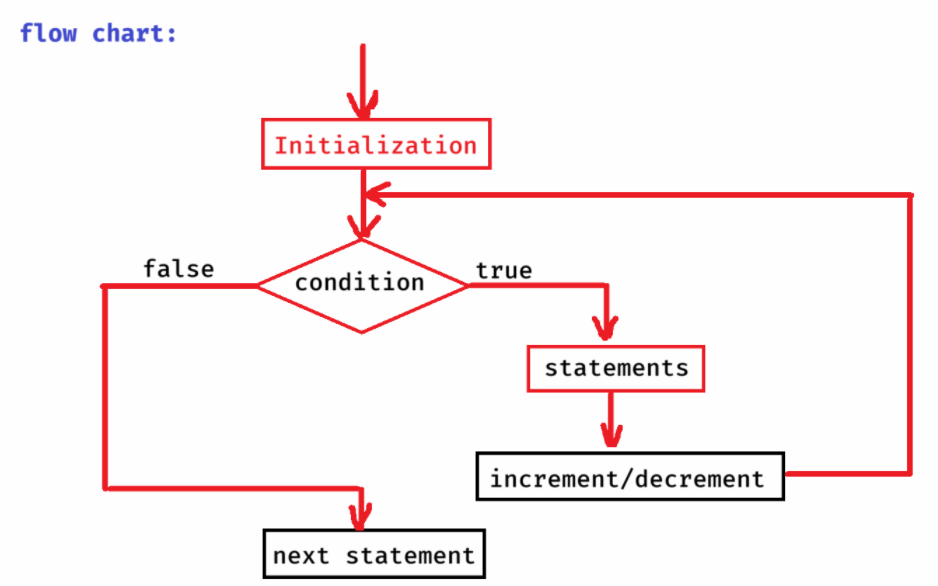
Consonant

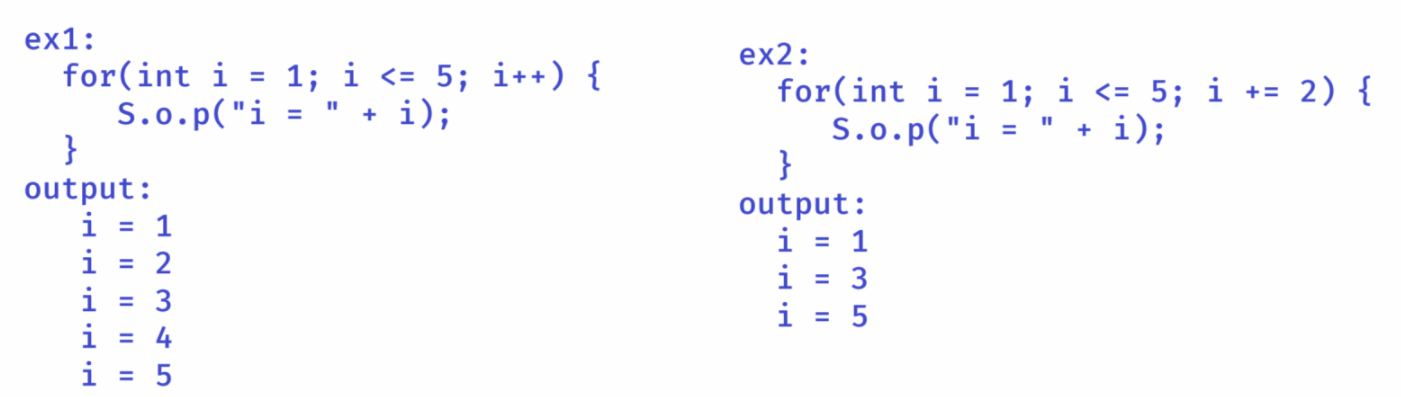
for loop :

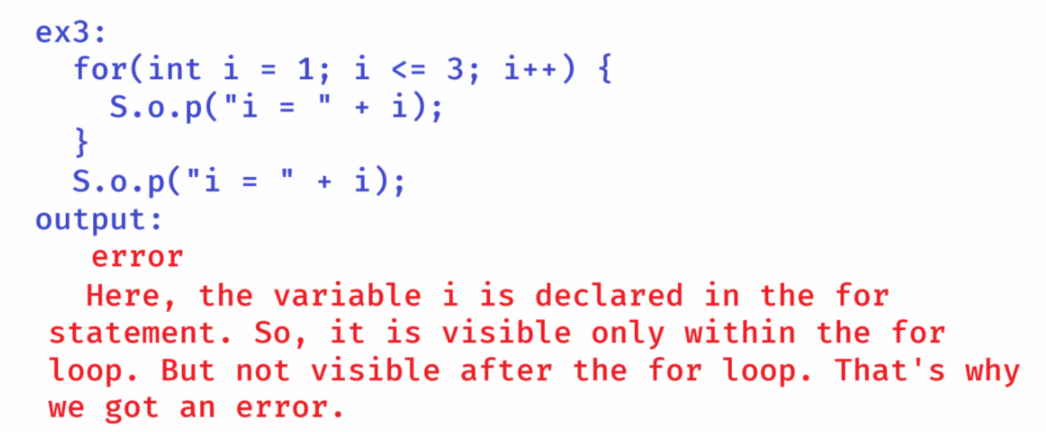
* If you want to execute some statements in a program repeatedly for multiple times based on some condition, then you need a loop.
* If you know how many times you need to repeat the loop, then use for loop.

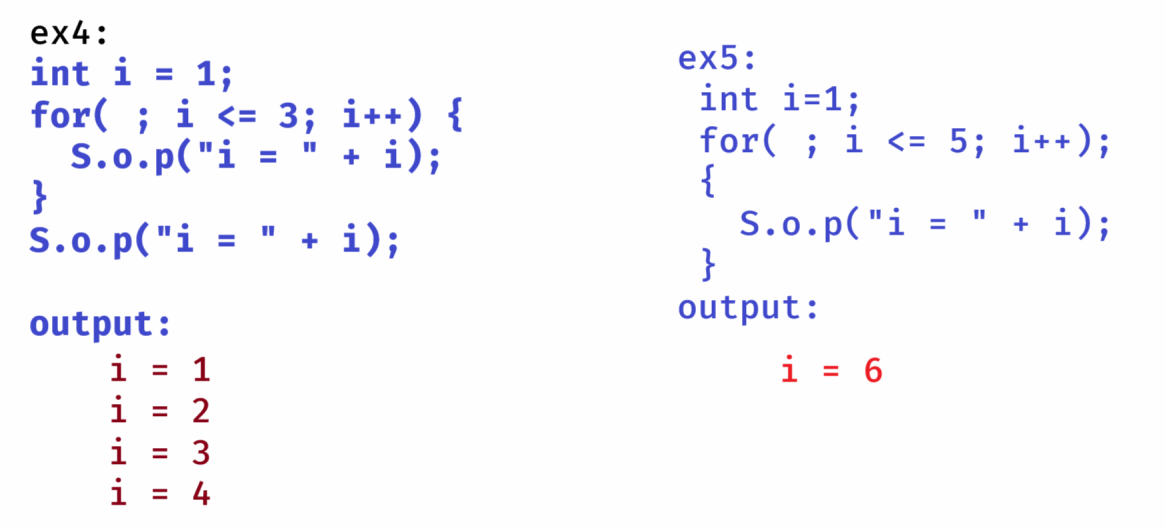












* Suppose, if you write the logic in a main method then when you run the program, JVM calls the main method and the logic in the main will be executed and then the program is terminated.
* If you want to execute the same logic again, then you have to run the program again.
* To make the logic as reusable, it is better to define the logic in another method. So, that you can call that method many times.
* The basic syntax of write a method is,

returntype methodname(argument1, argument2, ..)

{

//logic

}

ex:

void add(int x, int y)

{

//logic

}

Here, void is the return type and its meaning is, there is no value to return from this method.

ex:

int multiply(int a, int b)

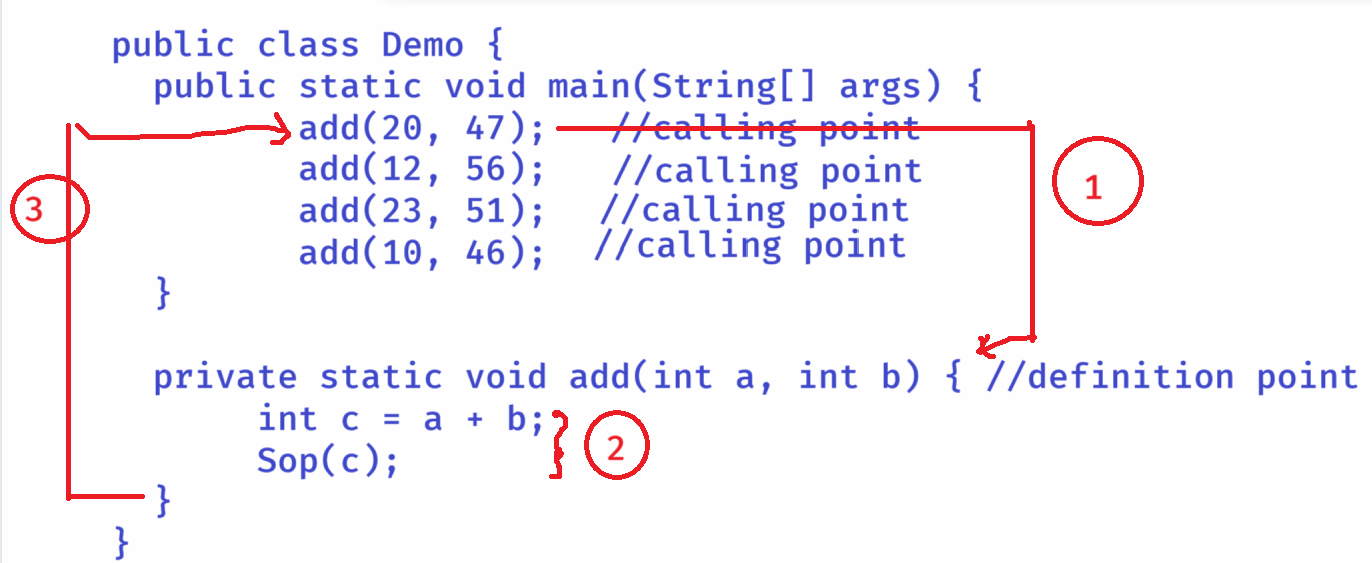
{

//logic

}

Here, int is the return type and its meaning is, the method should return an integer value.

example:



1. when you call a method, the control jumps from calling point to the definition point.
2. The logic defined in that method will be executed.
3. After execution, the control will be sent back to the calling point.

/\*

\* This program reads n value from the user

\* and calculates the sum of n Natural numbers.

\* ex:

\* if n = 5,

\* 1 + 2 + 3 + 4 + 5 = 15 //output

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** Sum {

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

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

System.***out***.println("Please enter n value to calculate sum of n Natural numbers");

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

*findSum*(n); //calling point

}

**private** **static** **void** findSum(**int** n) { //definition point

**int** sum = 0;

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

sum = sum + i;

}

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

}

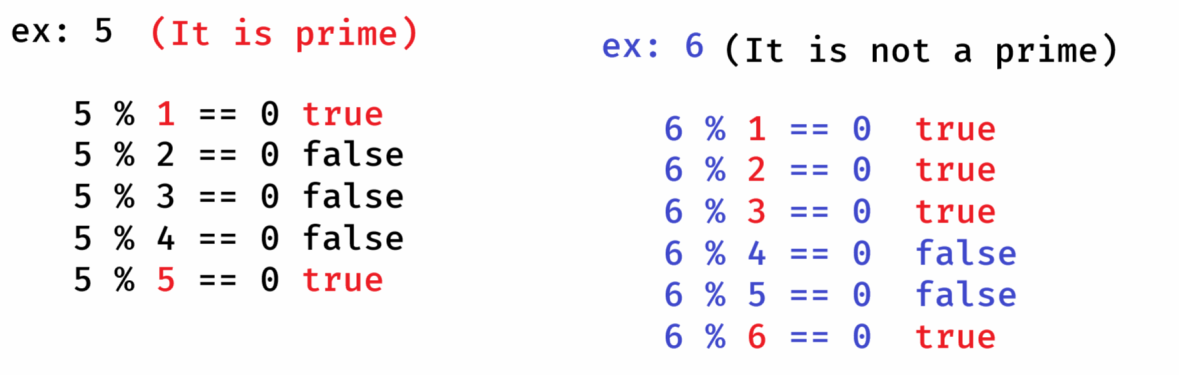
}

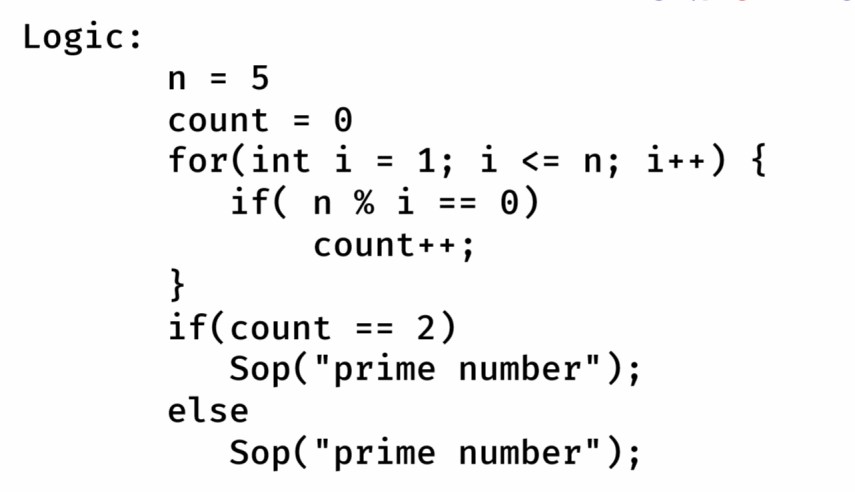
**Prime number :**

**------------**

* **A natural number (>=1), which contains only 2 factors(1 and itself) is called as a prime number.**

**ex:**

****

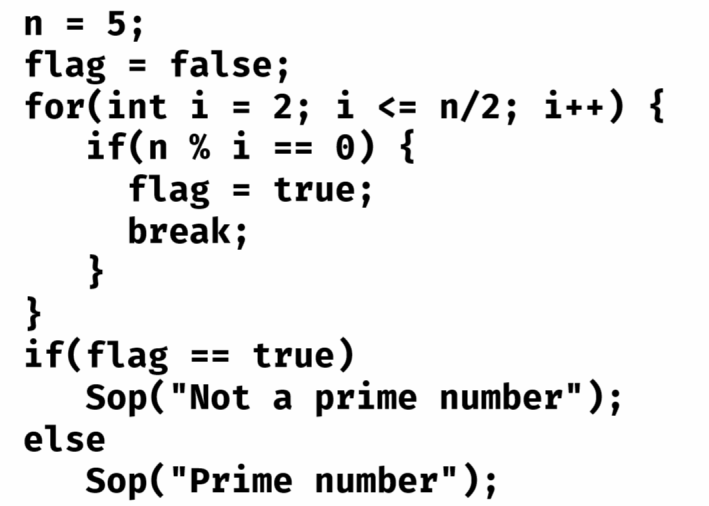
****

**problem with the above logic:**

**\* if n value is large, then the loop will repeat more times.**

**So, it will decrease the performance of the program.**

**Logic2:**

****

**problem with this logic:**

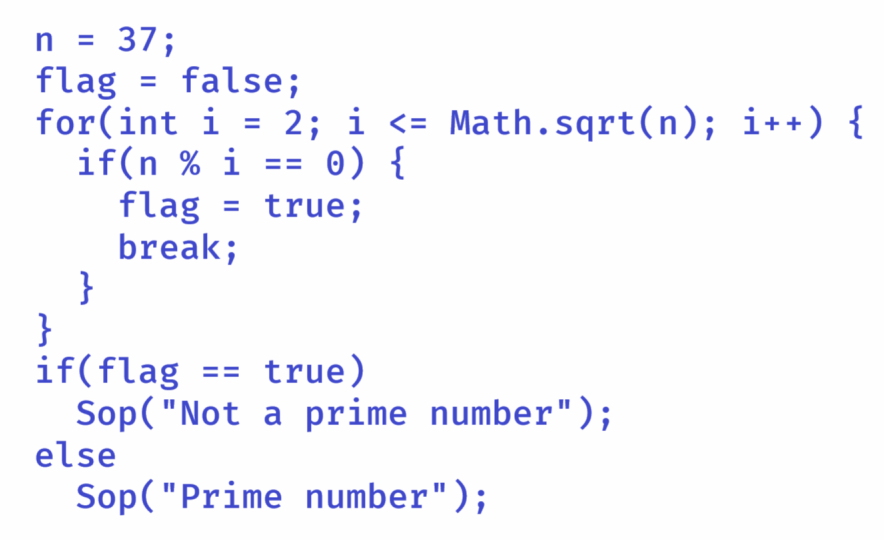
**\* This logic is reducing the iterations by approximately 50%.**

**But still if the n value is large, then the loop will repeat**

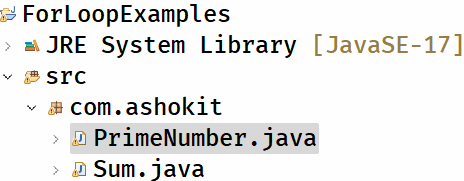
**more times. So, it will decrease the performance of the**

**program.**

**Logic3:**

****

* **This is best logic to check for a prime number. Because it reduces lot of loop iterations, and improves the performance.**

****

**PrimeNumber.java**

**--------------**

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** PrimeNumber {

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

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

System.***out***.println("Enter a number to check it is prime or not");

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

*checkPrime*(n);

}

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

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

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

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

flag = **true**;

**break**;

}

}

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

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

**else**

System.***out***.println("It is a prime number");

}

}



**PerfectNumber.java**

**--------------**

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** PerfectNumber {

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

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

System.***out***.println("Enter a number to check whether it is perfect or not");

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

*checkPerfect*(n);

scanner.close();

}

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

**int** sum = 0;

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

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

sum = sum + i;

}

**if**( sum == n)

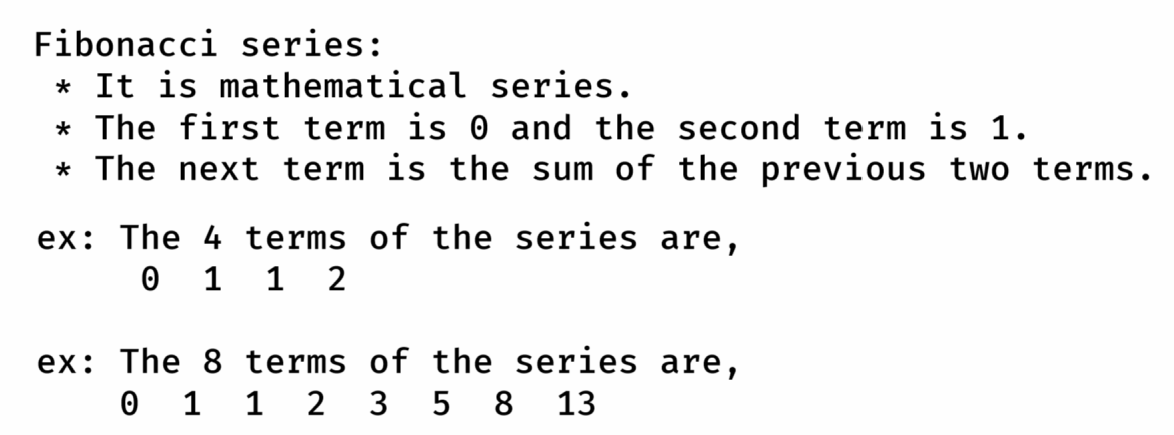
System.***out***.println("It is perfect number");

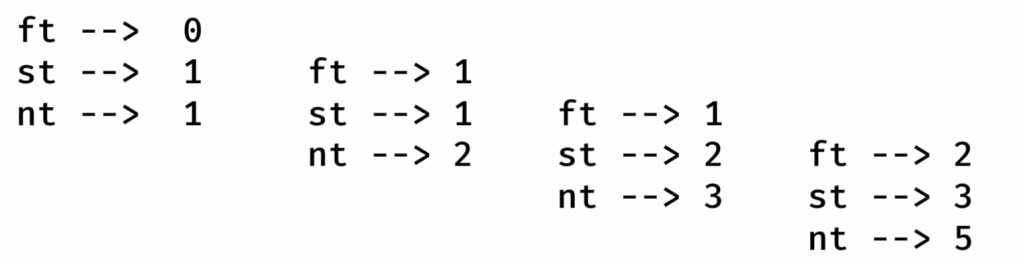
**else**

System.***out***.println("It is not a perfect number");

}

}





FibonacciSeries.java

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

**package** com.ashokit;

**import** java.util.Scanner;

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

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

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

System.***out***.println("Enter n value to print fibonacci series");

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

*printFibonacci*(n);

scanner.close();

}

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

**int** firstTerm = 0;

**int** secondTerm = 1;

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

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

**int** nextTerm = firstTerm + secondTerm;

firstTerm = secondTerm;

secondTerm = nextTerm;

}

}

}

Q) write a program to find the factorial of a given number?

Q) write a program to print the multiplication table of a given number?

Nested for loop:

* If you write a for loop inside another for loop then it is called a nested for loop.
* For each iteration of the outer for loop, the inner for loop will be completely executed.

ex:

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

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

Sop(j);

}

}

output:

1

2

1

2

ex:

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

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

S.o.print(j + “ “);

}

S.o.println();

}

output:

1

1 2

1 2 3

Pattern programs:

* while writing the pattern programs, follow the below steps.
* 1. we always use nested for loops to print the pattern

2. we repeat the outer loop for rows/lines.

* 3. we repeat the inner loop for columns, by somehow finding
* the relationship between rows and columns.
* 4. we always print star(\*) in the inner loop.

**ex: print the below right angle triangle star(\*) pattern.**

**if n = 5, then output should be**

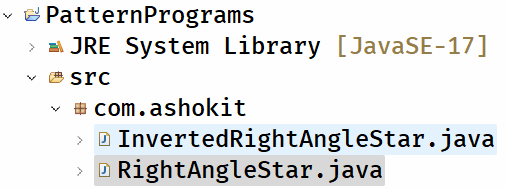
**\***

**\* \***

**\* \* \***

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

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

****

**RightAngleStar.java**

**package com.ashokit;**

**import java.util.Scanner;**

**public class RightAngleStar {**

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

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

**System.*out*.println("Enter n value to print Right angle star pattern");**

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

***showPattern*(n);**

**scanner.close();**

**}**

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

**//rows**

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

**//columns**

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

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

**}**

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

**}**

**}**

**}**

**InvertedRightAngleStar.java**

**/\*\***

**\* This program should display inverted right angle star**

**\* pattern for the given number of rows.**

**\* example:**

**\* n = 5**

**\* output:**

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

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

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

**\* \* \***

**\* \***

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class InvertedRightAngleStar {**

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

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

**System.*out*.println("Enter n value to print Inverted Right angle star pattern");**

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

***showPattern*(n);**

**scanner.close();**

**}**

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

**// rows**

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

**// columns**

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

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

**}**

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

**}**

**}**

**}**

**LeftAngleStar.java**

**/\*\***

**\* This program should print Left Angle Star pattern for**

**\* the given number of rows.**

**\* example:**

**\* n = 5**

**\* output:**

**\* \***

**\* \* \***

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

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

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

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class LeftAngleStar {**

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

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

**System.*out*.println("Enter n value to print Left angle star pattern");**

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

***showPattern*(n);**

**scanner.close();**

**}**

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

**//rows**

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

**//inner loop1 : spaces**

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

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

**}**

**//inner loop2 : stars**

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

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

**}**

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

**}**

**}**

**}**

InvertedLeftAngleStar.java

**/\*\***

**\* This program should print inverted left angle star**

**\* pattern for the given number of rows.**

**\* example:**

**\* n = 5**

**\* output:**

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

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

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

**\* \* \***

**\* \***

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class InvertedLeftAngleStar {**

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

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

**System.*out*.println("Enter n value to print Inverted Left angle star pattern");**

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

***showPattern*(n);**

**scanner.close();**

**}**

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

**// rows**

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

**//inner loop1 : spaces**

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

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

**}**

**// inner loop2 : stars**

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

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

**}**

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

**}**

**}**

**}**

PyramidPattern.java

**/\*\***

**\* This program prints a pyramid pattern for the given**

**\* number of rows.**

**\* example:**

**\* if n = 5,**

**\* output:**

**\* \***

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

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

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

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

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class PyramidPattern {**

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

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

**System.*out*.println("Enter n value to print Pyramid star pattern");**

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

***showPattern*(n);**

**scanner.close();**

**}**

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

**// rows**

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

**//inner loop1 : spaces**

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

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

**}**

**// inner loop2 : stars**

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

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

**}**

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

**}**

**}**

**}**

PascalTriangle.java

**/\*\***

**\* This program should print a pascal triangle for the**

**\* given number of rows.**

**\* example:**

**\* if n = 5**

**\* output:**

**\* 1**

**\* 1 1**

**\* 1 2 1**

**\* 1 3 3 1**

**\* 1 4 6 4 1**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class PascalTriangle {**

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

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

**System.*out*.println("Enter n value to print the pascal triangle");**

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

***showPattern*(n);**

**scanner.close();**

**}**

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

**//rows**

**for(int i = 0; i <= n - 1; i++) {**

**//spaces**

**for(int k = 0; k <= n - i - 1; k++) {**

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

**}**

**//terms of the pascal triangle**

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

**int t = *factorial*(i) / (*factorial*(j) \* *factorial*(i-j));**

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

**}**

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

**}**

**}**

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

**int fact = 1;**

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

**{**

**fact = fact \* i;**

**}**

**return fact;**

**}**

**}**

**/\*\***

**\* write a program to display the below character pattern.**

**\* example:**

**\* if n = 5,**

**\* output:**

**\* A**

**\* A B**

**\* A B C**

**\* A B C D**

**\* A B C D E**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class CharacterPattern {**

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

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

**System.*out*.println("Enter n value to print Right angle character pattern");**

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

***showPattern*(n);**

**scanner.close();**

**}**

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

**//rows**

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

**//columns**

**for(char ch = 'A'; ch <= 65 + i - 1; ch++) {**

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

**}**

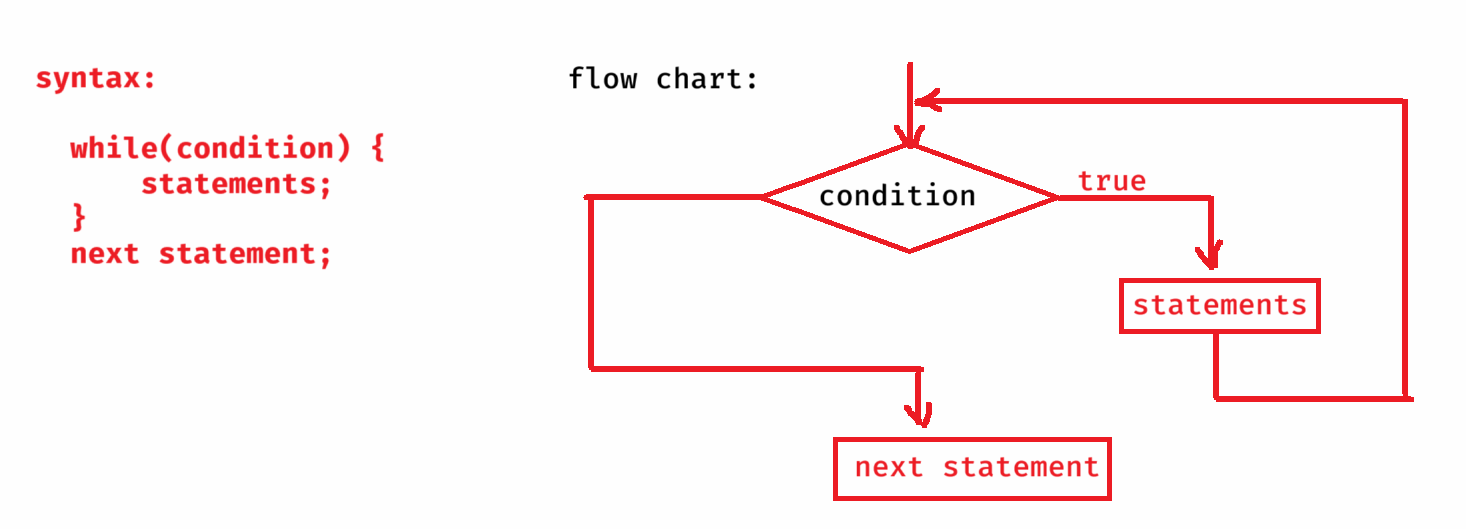
**System.*out*.println();**

**}**

**}**

**}**

while loop



**/\*\***

**\* write a program to find the sum of the digits of**

**\* a given number.**

**\* ex:**

**\* if n = 149**

**\* output: 14**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class SumOfDigits {**

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

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

**System.*out*.println("Enter a number to find the sum of its digits");**

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

***findSumOfDigits*(n);**

**scanner.close();**

**}**

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

**int sum = 0;**

**while( n != 0 ) {**

**int d = n % 10;**

**sum = sum + d;**

**n = n / 10;**

**}**

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

**}**

**}**

Armstrong number:

th

* If the sum of n 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 = 153

So, 153 is an Armstrong number.

ex:

number = 1634

4 4 4 4

1 + 6 + 3 + 4 = 1634

So, 1634 is an Armstrong number

. First findout/count how many digits are there in the given number.

. After that find the nth power of each digit and add it to the sum variable.

. finally check is the sum is matching with the given number or not. If yes, then the number is armstrong. Otherwise, not an armstrong number.

Armstrong.java

**/\***

**\* Write a program to check whether a given number is**

**\* an Armstrong number or not**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class Armstrong {**

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

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

**System.*out*.println("Enter a number to verity is it an Armstrong number or not");**

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

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

**if(flag == true)**

**System.*out*.println("It is an Armstrong number");**

**else**

**System.*out*.println("It is not an Armstrong number");**

**scanner.close();**

**}**

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

**int sum = 0;**

**int temp1 = n;**

**int temp2 = n;**

**int count = 0;**

**//This while loop counts the num of digits**

**while(temp1 != 0) {**

**int d = temp1 % 10;**

**count++;**

**temp1 = temp1 / 10;**

**}**

**//This while loop finds the sum**

**while(temp2 != 0) {**

**int d = temp2 % 10;**

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

**temp2 = temp2 / 10;**

**}**

**if( sum == n )**

**return true;**

**else**

**return false;**

**}**

**}**

Strong number:

* If sum of the factorials of each digit of a number is equal to the same number then it is a strong number.

ex: n = 145

1! + 4! + 5! = 145

So, 145 is a strong number

Palindrome Number:

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

* If reverse of a number is equal to the same number then we call it as a palindrome number.

ex: number = 145

reverse = 541

Not a palindrome

ex:

number = 12321

reverse = 12321

Yes, it’s a palindrome.

Palindrome.java

**/\*\***

**\* This program checks whether a given number is**

**\* palindrome or not**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class Palindrome {**

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

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

**System.*out*.println("Enter a number to check is it palindrome or not");**

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

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

**if(flag == true)**

**System.*out*.println("Yes, it is a palindrome number");**

**else**

**System.*out*.println("No, it is not a palindrome number");**

**scanner.close();**

**}**

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

**int temp = n;**

**int rev = 0;**

**while( n != 0) {**

**int d = n % 10;**

**rev = rev \* 10 + d;**

**n = n / 10;**

**}**

**if(rev == temp)**

**return true;**

**else**

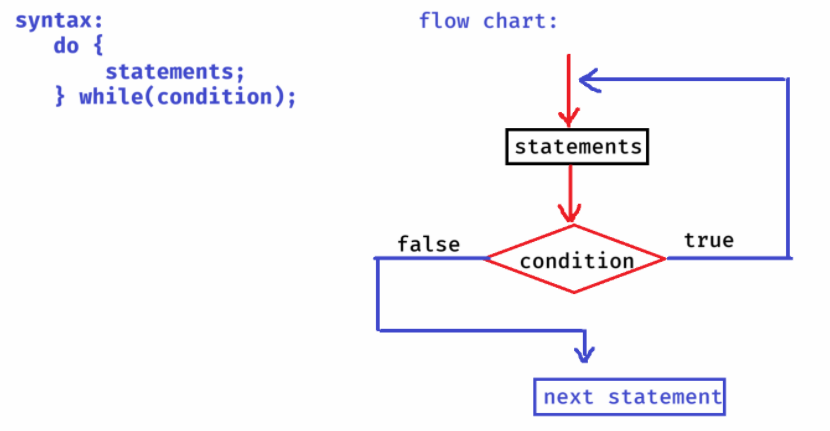
**return false;**

**}**

**}**

**do while loop:**

**-----------**

****

**Guess.java**

**/\*\***

**\* This program will accept a number b/w 0 to 9 from the**

**\* user and also generates a random number b/w 0 to 9.**

**\* If both are matched, then Guess is correct. Otherwise,**

**\* Guess is wrong.**

**\*/**

**package com.ashokit;**

**import java.util.Random;**

**import java.util.Scanner;**

**public class Guess {**

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

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

**int n = -1;**

**do {**

**System.*out*.println("Enter a number between 0 to 9");**

**n = scanner.nextInt();**

**} while( n < 0 || n > 9);**

**Random r = new Random();**

**int k = r.nextInt(10); // generates a random number b/w 0 to 9**

**if(n == k) {**

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

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

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

**}**

**else {**

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

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

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

**}**

**scanner.close();**

**}**

**}**

1. **write a program to print all the prime numbers between 1 to 100.**
2. **write a program to print all the armstrong numbers between 1 to 1000**

**break & continue statements:**

* **break statement is used to move the control to out of the loop or switch statement.**

**ex:**

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

***{***

***if( i \* 2 > 2)***

**break;**

***Sop(i);***

***}***

**output: 1**

**ex:**

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

**{**

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

**{**

**if(i+j > 3)**

**break;**

**Sop(j);**

**}**

**}**

**output:**

**1**

**2**

**1**

**=============**

**PinCheck.java**

**/\*\***

**\* This program is to read atm pin from the user.**

**\* It provides 3 attempts. If 3 attempts are not correct then displays locked.**

**\* If matched in <= 3 attempts, then display pin valid, then breaks the loop.**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class PinCheck {**

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

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

**int correctPin = 1234;**

**int attempt=1;**

**for ( ; attempt <= 3; attempt++) {**

**System.*out*.println("Enter ATM pin : ");**

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

**if (enteredPin == correctPin) {**

**System.*out*.println("Pin is valid. Welcome to ATM");**

**break;**

**} else {**

**System.*out*.println("Pin is incorrect! Remaining attempts : " + (3 - attempt));**

**}**

**}**

**if (attempt > 3)**

**System.*out*.println("Your card is locked for 24 hours");**

**}**

**}**

* **continue statement is used to move the control for the next iteration of the loop, by skipping the execution of the remaining staements of the loop, after the continue statement.**

**ex:**

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

**if( i \* i < 6)**

**continue;**

**Sop(i);**

**}**

**output:**

**3**

**4**

**5**

**Solution.java**

**/\***

**\* This program finds the sum of 5 positive integers**

**\* entered by the user.**

**\* if negitive integer is entered, then continue statement will**

**\* repeat the loop for next iteration.**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class Solution {**

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

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

**int sum = 0;**

**int count = 0;**

**while(true) { //infinite loop**

**System.*out*.println("Enter a positive integer");**

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

**if( n < 0)**

**continue;**

**sum = sum + n;**

**count++;**

**if(count == 5)**

**break;**

**}**

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

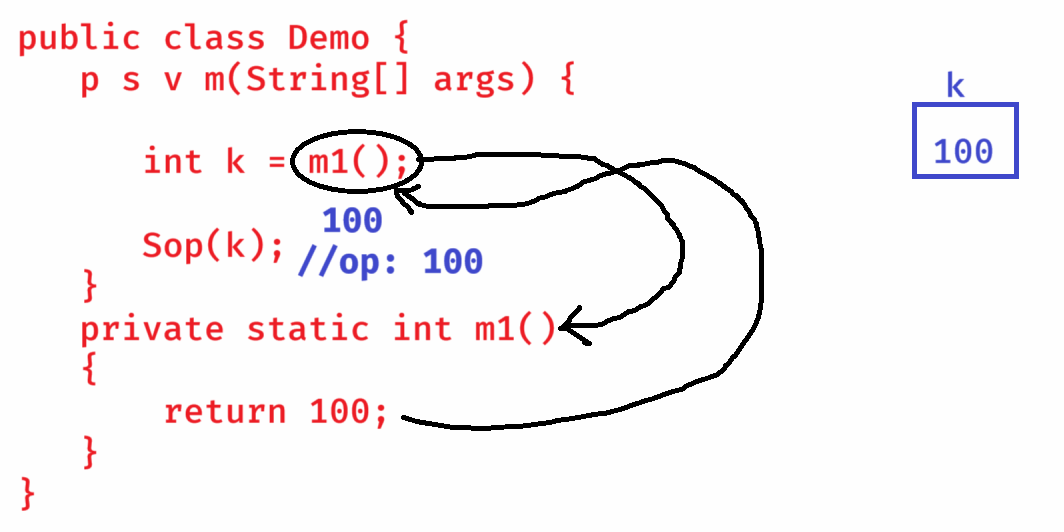
**}**

**}**

**return and exit statements:**

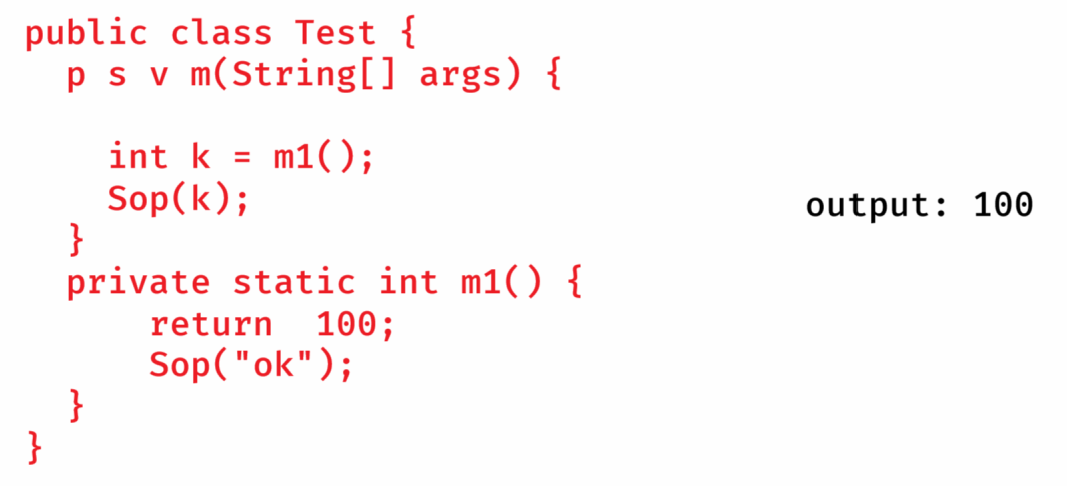
* **return statement is used to return/move the control from the method definition to the method calling point.**
* **If a method returns any value, then the method calling point will be replaced with the returned value.**

**ex:**

****

* **If return statement is executed, the next statements after the return will not be executed.**

**ex:**

****

* **exit statement will terminate the currently running JVM immediately.**
* **After exit statement, no statements of the program will be executed.**

****

**Arrays**

* **Suppose, an application has multiple values and to hold them, multiple variables are created.**
* **The issues are,**

1. **If too many variables are created, then code complexity will be increased.**
2. **The memory for the variables are allocated randomly at different places. So, retrieving/fetching the value of a variable requires entire memory search by the JVM.**

**So, the performance of an application will be decreased.**

* **The solution for the above problems is arrays.**
* **An array is a single variable but it can store multiple values, in continuous memory locations.**
* **So, the number of variables are decreased and the performance will be increased.**

**array declaration, creation and initialization:**

**------------------------------------------**

**datatype[] arrayname; // array declaration**

**(or)**

**datatype arrayname[]; //array declaration**

**arrayname = new datatype[size]; //array creation**

**. we can combine array declaration and creation together.**

**datatype[] arrayname = new datatype[size];**

**ex:**

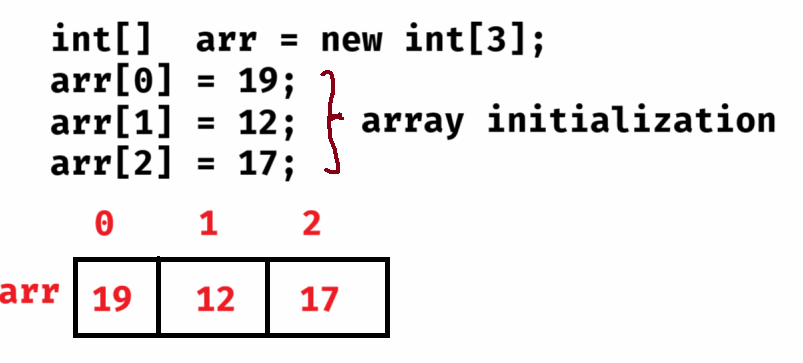
**int[] tkt\_numbers = new int[200];**

**float[] prices\_in\_this\_week = new float[7];**

**char[] vowels = new char[5];**

**ex:**

**int[] arr = new int[3];**

****

* **we can also combine array declarion, creation and initialization in a single statement.**

**ex:**

**int[] arr = new int[] { 19, 12, 17};**

**(or)**

**int[] arr = { 19, 12, 17};**

**length attribute:**

**---------------**

* **length attribute can be used to find the size of an array.**

**ex1:**

**int[] arr = new int[3];**

**arr[0]=10;**

**arr[1]=20;**

**arr[2]=30;**

**System.out.println( arr.length ); //output: 3**

**ex2:**

**int[] arr = new int[5];**

**arr[0] = 10;**

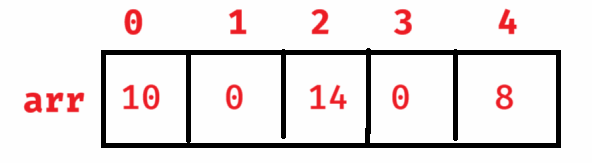
**arr[2] = 14;**

**arr[4] = 8;**

**System.out.println(arr.length); //output: 5**

**System.out.println(arr[1]); //output: 0**

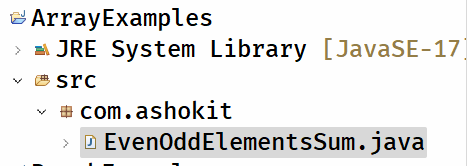
**System.out.println(arr[3]); //output: 0**

****

**limitations of an array:**

**---------------------**

1. **Once an array is created with a size, it is fixed. We can’t increase/decrease the size.**
2. **If an array is created with more than required size then memory will be wasted. If created with less than required size then we will get shortage of memory.**
3. **An array can only store homogeneous elements(same type).**
4. **We don’t have pre-defined methods to perform insert/delete/update/search/sort operations. It is called lack of pre-defined methods.**

****

**EvenOddElementsSum.java**

**-----------------**

**/\*\***

**\* This program finds sum of even and odd elements**

**\* of an array separately.**

**\* It should read the size and the elements of an array**

**\* from the user.**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class EvenOddElementsSum {**

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

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

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

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

**//create an array**

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

**//store elements**

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

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

**arr[i] = scanner.nextInt();**

**}**

***findEvenOddSum*(arr);**

**scanner.close();**

**}**

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

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

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

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

**evenSum = evenSum + arr[i];**

**else**

**oddSum = oddSum + arr[i];**

**}**

**System.*out*.println("sum of even elements : " + evenSum);**

**System.*out*.println("sum of odd elements : " + oddSum);**

**}**

**}**

**MaxConsecutiveElement.java**

**package com.ashokit;**

**public class MaxConsecutiveElement {**

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

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

***findMaxConsecutiveElement*(arr);**

**}**

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

**int maxi = 0, count = 1, element = 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;**

**element = arr[i];**

**}**

**}**

**System.*out*.println("The element : " + element + " has repeated consecutively for : "+maxi + " times");**

**}**

**}**

**LinearSearch.java**

**/\***

**\* This program will search for the given element**

**\* in an array, using linear search.**

**\* linear search : comparing the given element with**

**\* each element of the array sequentially.**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class LinearSearch {**

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

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

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

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

**if(size < 1) {**

**System.*out*.println("Invalid array size!");**

**System.*exit*(0);**

**}**

**//create an array**

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

**//store the elements**

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

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

**arr[i] = scanner.nextInt();**

**}**

**System.*out*.println("Enter search element");**

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

**boolean flag = *linearSearch*(arr, searchElement);**

**if (flag == true)**

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

**else**

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

**scanner.close();**

**}**

**private static boolean linearSearch(int[] arr, int searchElement) {**

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

**if ( arr[i] == searchElement )**

**return true;**

**}**

**return false;**

**}**

**}**

**Solution.java**

**/\***

**\* write a program to search for an element in the given array**

**\* using binary search.**

**\***

**\* binary search:**

**\* 1. array elements must be in ascending order.**

**\* 2. find the low and high index of the array**

**\* 3. find the mid index as (low + high) /2**

**\* 4. if searching element is equal to arr[mid], then element**

**\* is found, andbreak the loop.**

**\* 5. if searching element > arr[mid], then low = mid + 1**

**\* 6. if searching element < arr[mid], then high = mid - 1**

**\* 7. repeat the steps 3 to 6, until low <= high**

**\* 8. if low > high, then element is not found.**

**\*/**

**import java.util.Arrays;**

**import java.util.Scanner;**

**public class Solution {**

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

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

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

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

**if( size < 1 ) {**

**System.*out*.println("size invalid. Try again....");**

**System.*exit*(1);**

**}**

**//create an array**

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

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

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

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

**}**

**System.*out*.println("Enter searching element");**

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

***binarySearch*(arr, searchingElement);**

**scan.close();**

**}**

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

**//sort the array**

**Arrays.*sort*(arr);**

**int low = 0;**

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

**boolean flag = false;**

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

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

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

**System.*out*.println("element is found at index: " + mid);**

**flag = true;**

**break;**

**}**

**else if(arr[mid] > searchingElement)**

**high = mid - 1;**

**else**

**low = mid + 1;**

**}**

**if(flag == false) {**

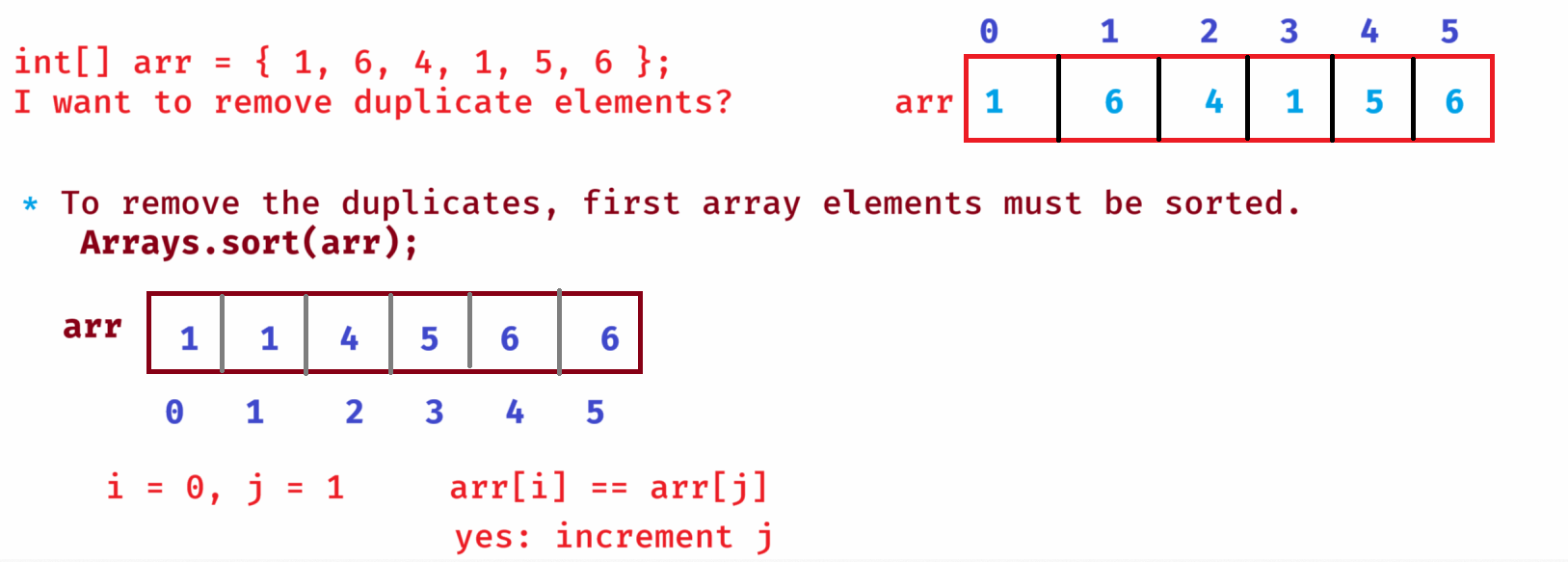
**System.*out*.println("element is not found");**

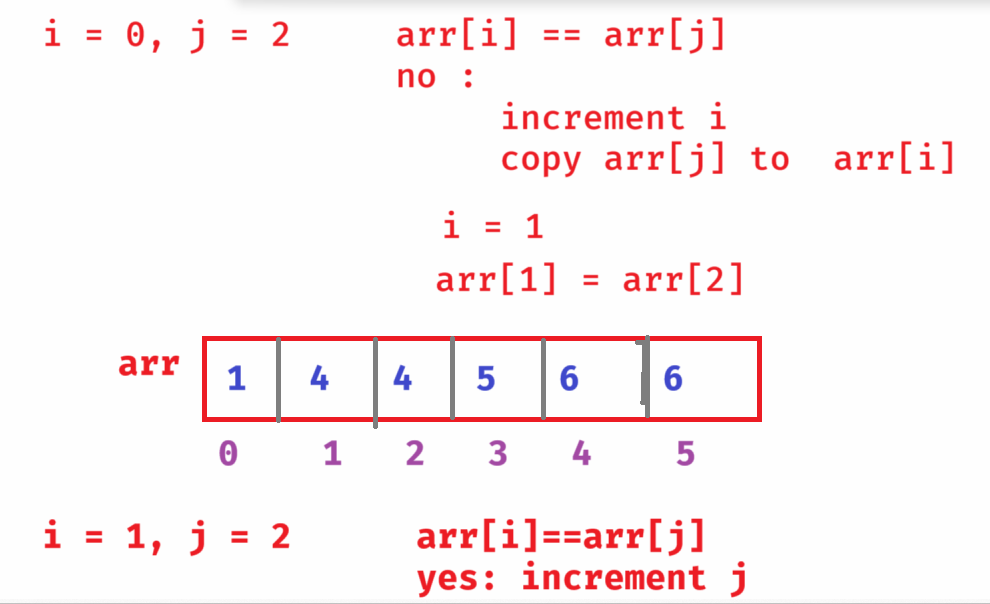
**}**

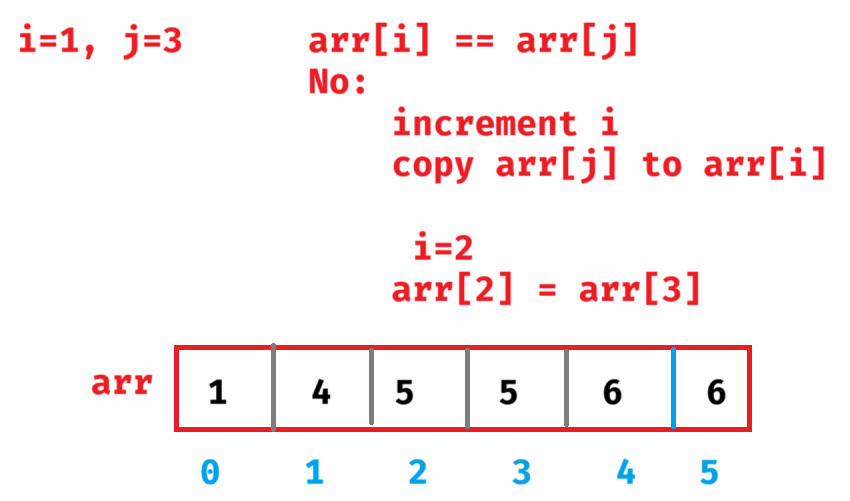
**}**

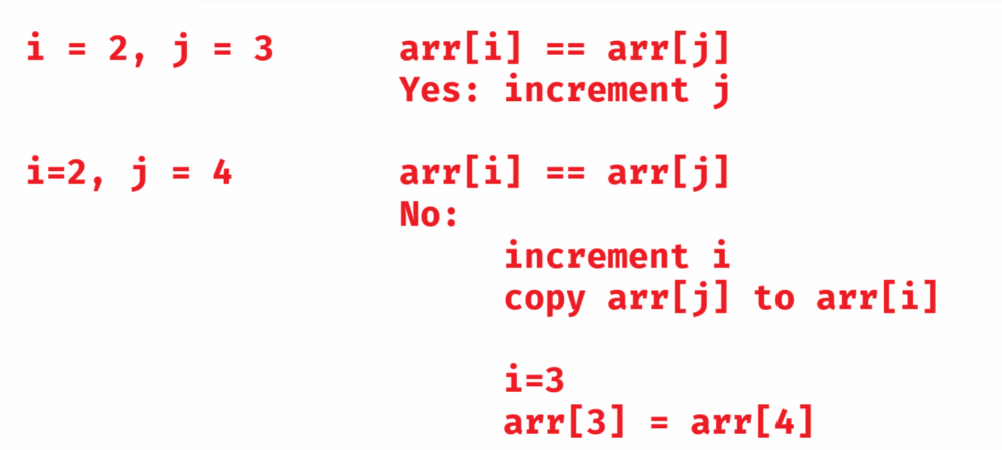
**}**

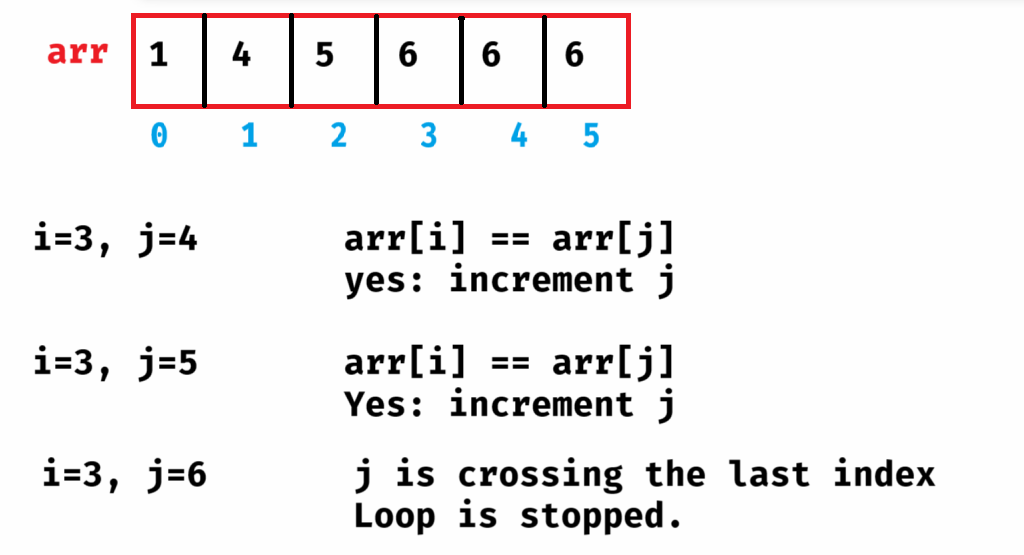
**remove the duplicate elements from the array:**

****

****

****

****

****

****

RemoveDuplicates.java

**/\***

**\* This program will remove the duplicate elements**

**\* and prints only unique elements of the array.**

**\*/**

**package com.ashokit;**

**import java.util.Arrays;**

**public class RemoveDuplicates {**

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

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

***removeDuplicates*(arr);**

**}**

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

**int i = 0;**

**System.*out*.println("Original array elements : ");**

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

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

**}**

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

**System.*out*.println("+\*+".repeat(20));**

**//sort the array**

**Arrays.*sort*(arr);**

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

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

**continue;**

**else {**

**i++;**

**arr[i] = arr[j];**

**}**

**}**

**//display**

**System.*out*.println("After removing duplicate elements : ");**

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

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

**}**

**}**

**}**

for each loop:

* **In Java, we have two types of for loops.**

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

* **for each loop can be used only with arrays and collections.**

**syntax:**

**for(datatype variable : array/collection)**

**{**

**statements;**

**}**

**ex:**

**char[] ch = { ‘a’, ‘e’, ‘i’, ‘o’, ‘u’ };**

**for(char c : ch) {**

**S.o.p(c);**

**}**

**======================**

**ArraysUnion.java**

**/\*\***

**\* This program finds the union of 2 arrays.**

**\* ex:**

**\* int[] a = {2, 5, 1, 4};**

**\* int[] b = {1, 3, 5}**

**\* output:**

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

**\*/**

**package com.ashokit;**

**import java.util.LinkedHashSet;**

**public class ArraysUnion {**

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

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

**int[] b = { 1, 3, 5 };**

**int[] c = *findUnion*(a, b);**

**System.*out*.println("The result of union operation");**

**for(int k : c) {**

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

**}**

**}**

**private static int[] findUnion(int[] a, int[] b) {**

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

**//add first array elements to the LinkedHashSet**

**for(int x : a) {**

**lhs.add(x);**

**}**

**//add second array elements to the LinkedHashSet**

**for(int x : b) {**

**lhs.add(x);**

**}**

**int[] c = new int[lhs.size()];**

**int i = 0;**

**//copy the elements from lhs object to new array**

**for(int x : lhs) {**

**c[i] = x;**

**i++;**

**}**

**return c;**

**}**

**}**

**ArraysIntersection.java**

**/\*\***

**\* This program finds the intersection of 2 arrays.**

**\* ex:**

**\* a[ ] = { 3, 0, 6, 7, 5}**

**\* b[ ] = { 2, 9, 0, 7, 4}**

**\* output:**

**\* 0 7**

**\*/**

**package com.ashokit;**

**import java.util.LinkedHashSet;**

**public class ArraysIntersection {**

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

**int[] arr1 = { 2, 7, 0, 1, 4};**

**int[] arr2 = { 3, 0, 5, 4, 7, 2};**

***findIntersection*(arr1, arr2);**

**}**

**private static void findIntersection(int[] arr1, int[] arr2) {**

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

**//add first array elements to the lhs**

**for(int x : arr1) {**

**lhs.add(x);**

**}**

**//read the elements from second array and**

**//check is it exist in lhs**

**for(int k : arr2) {**

**if(lhs.contains(k)) {**

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

**}**

**}**

**}**

**}**

**Bubble sort:**

* **In bubble sort, we have to compare the two adjacent elements.**
* **if the first element is greater than second element then we should swap them.**
* **In each pass, one largest element will be bubbled to the end of the array.**
* **If array has n elements, then elements can be sorted in n-1 passes.**
* **Suppose, if n = 5, then the elements indexes to compare are,**

**pass1 : 0,1 1,2 2,3 3,4**

**pass2 : 0,1 1,2 2,3**

**pass3 : 0,1 1,2**

**pass4 : 0,1**

**example:**

**[3, 8, 1, 5, 0]**

**pass1:**

**[3, 8, 1, 5, 0]**

**[3, 1, 8, 5, 0]**

**[3, 1, 5, 8, 0]**

**[3, 1, 5, 0, 8] 8 is bubbled.**

**pass2:**

**[1, 3, 5, 0, 8]**

**[1, 3, 5, 0, 8]**

**[1, 3, 0, 5, 8] 5 is bubbled**

**pass3:**

**[1, 3, 0, 5, 8]**

**[1, 0, 3, 5, 8] 3 is bubbled.**

**pass4:**

**[0, 1, 3, 5, 8] 1 is bubbled.**

**recursion:**

**--------**

* **recursion means, calling a method again from the body of the same method.**
* **recursion must know, when it should stop. Otherwise, it becomes infinite, at some point of time, JVM throws StackOverflowError.**

**FactorialReursion.java**

**package com.ashokit;**

**import java.util.Scanner;**

**public class FactorialRecursion {**

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

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

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

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

**int result = *factorial*(n);**

**System.*out*.println("Factorial = " + result);**

**scanner.close();**

**}**

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

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

**return 1;**

**else**

**return n \* *factorial*(n - 1);**

**}**

**}**

**FibonacciRecursion.java**

**/\*\***

**\* This program will print the fibonacci series of**

**\* n terms, with recursion.**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class FibonacciRecursion {**

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

**//connect Scanner to the keyboard input**

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

**System.*out*.println("Enter n value to print fibonacci terms");**

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

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

**int k = *fibonacci*(i);**

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

**}**

**scanner.close();**

**}**

**private static int fibonacci(int i) {**

**if (i == 0)**

**return 0;**

**if (i == 1)**

**return 1;**

**return *fibonacci*(i-1) + *fibonacci*(i-2);**

**}**

**}**

**MergeSort.java**

**package com.ashokit;**

**public class MergeSort {**

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

**int[] arr = { 3, 0, 9, 5, 2, 11 };**

**System.*out*.println("Original array:");**

***printArray*(arr);**

***mergeSort*(arr, 0, arr.length - 1);**

**System.*out*.println("Sorted array:");**

***printArray*(arr);**

**}**

**// Main mergeSort method**

**private static void mergeSort(int[] array, int left, int right) {**

**if (left < right) {**

**// Find the middle point**

**int mid = (left + right) / 2;**

**// Sort the first and second halves**

***mergeSort*(array, left, mid);**

***mergeSort*(array, mid + 1, right);**

**// Merge the sorted halves**

***merge*(array, left, mid, right);**

**}**

**}**

**// Merge two subarrays**

**private static void merge(int[] array, int left, int mid, int right) {**

**// Sizes of subarrays**

**int n1 = mid - left + 1;**

**int n2 = right - mid;**

**// Create temp arrays**

**int[] L = new int[n1];**

**int[] R = new int[n2];**

**// Copy data**

**for (int i = 0; i < n1; i++)**

**L[i] = array[left + i];**

**for (int j = 0; j < n2; j++)**

**R[j] = array[mid + 1 + j];**

**// Merge temp arrays**

**int i = 0, j = 0;**

**int k = left;**

**while (i < n1 && j < n2) {**

**if (L[i] <= R[j]) {**

**array[k++] = L[i++];**

**} else {**

**array[k++] = R[j++];**

**}**

**}**

**// Copy remaining elements**

**while (i < n1) {**

**array[k++] = L[i++];**

**}**

**while (j < n2) {**

**array[k++] = R[j++];**

**}**

**}**

**// Utility to print array**

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

**for (int val : arr)**

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

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

**}**

**}**

**Two Dimensional array:**

**datatype[][] arrayname = new datatype[rows][cols];**

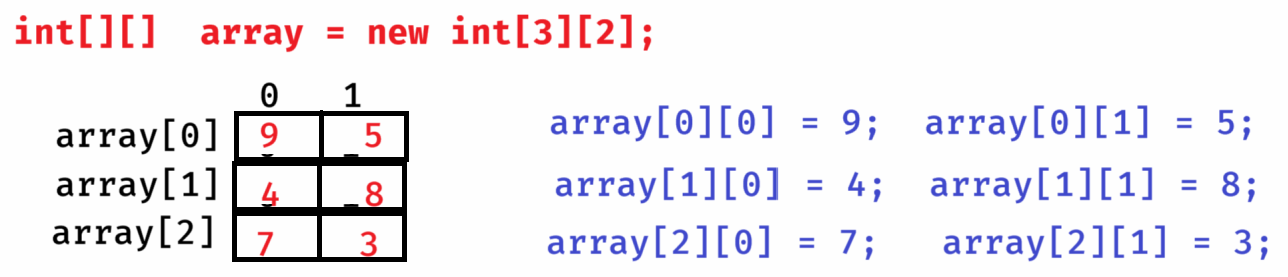
**(or)**

**datatype arrayname[][] = new datatype[rows][cols];**

**(or)**

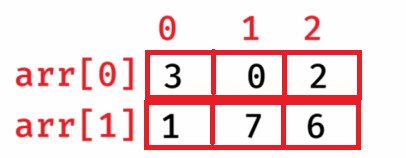
**datatype[] arrayname[] = new datatype[rows][cols];**

**ex:**

****

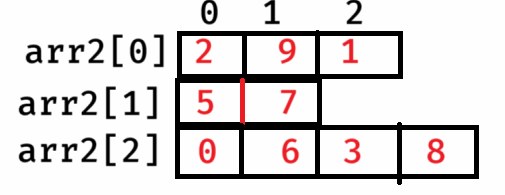
**static initialization:**

**int[][] arr = { {3, 0, 2}, {1, 7, 6} };**

****

**int[][] arr2 = { {2, 9, 1}, {5, 7}, {0, 6, 3, 8} };**

**\* Here, each row of the array has different size. So, it is called a Jogged Array.**

****

**DiagonalSum.java**

**---------------**

**/\*\***

**\* This program will find the sum of left and**

**\* right diagonals of the 2D array.**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class DiagonalSum {**

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

**int leftSum = 0;**

**int rightSum = 0;**

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

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

**if (i == j)**

**leftSum = leftSum + arr[i][j];**

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

**rightSum = rightSum + arr[i][j];**

**}**

**}**

**System.*out*.println("\u001B[31m" + "Left diagonal sum = " + leftSum);**

**System.*out*.println("\u001B[35m" + "Right diagonal sum = " + rightSum);**

**}**

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

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

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

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

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

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

**//create an array**

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

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

**//read the elements**

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

**for (int j = 0; j < cols; j++) {**

**System.*out*.println("Enter element for : " + i + ", " + j);**

**arr[i][j] = scanner.nextInt();**

**}**

**}**

***findDiagonalSum*(arr);**

**}**

**else {**

**System.*out*.println("It's rectangular matrix. So, diagonal doesn't exist!");**

**}**

**scanner.close();**

**}**

**}**

**//MatrixMultiplication.java**

**package com.ashokit;**

**public class MatrixMultiplication {**

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

**int[][] a = { {2, 5}, {1, 0}, {5, 3} }; // size 3 \* 2**

**int[][] b = { {1, 9}, {2, 4} }; // size 2 \* 2**

***multiply*(a, b);**

**}**

**private static void multiply(int[][] a, int[][] b) {**

**int r1 = a.length;**

**int c1 = a[0].length;**

**int r2 = b.length;**

**int c2 = b[0].length;**

**if (c1 == r2) {**

**int[][] c = new int[r1][c2];**

**for (int i = 0; i < r1; i++)**

**{**

**for (int j = 0; j < c2; j++)**

**{**

**int x = 0;**

**for (int k = 0; k < c1; k++)**

**{**

**x = x + a[i][k] \* b[k][j];**

**}**

**c[i][j] = x;**

**}**

**}**

**for(int i = 0; i < r1; i++) {**

**for (int j = 0; j < c2; j++) {**

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

**}**

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

**}**

**}**

**else**

**{**

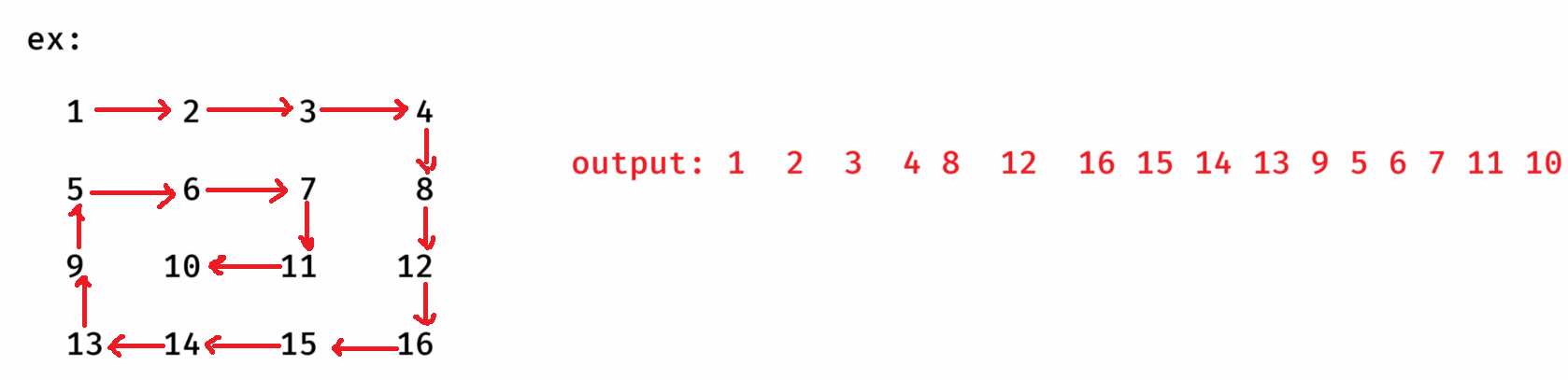
**System.*out*.println("multiplication not possible");**

**}**

**}**

**}**

**printing the elements in spiral manner.**

****

**Loop1: To print elements from left to right**

**Loop2: To print elements from top to bottom**

**Loop3: To print elements from right to left**

**Loop4: To print elements from bottom to top.**

**SpiralMatrix.java**

**package com.ashokit;**

**import java.util.ArrayList;**

**public class SpiralMatrix {**

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

**int[][] mat = { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15, 16} };**

***printSpiral*(mat);**

**}**

**private static void printSpiral(int[][] mat) {**

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

**int rows = mat.length; //no. of rows**

**int cols = mat[0].length; //no. of cols**

**int left = 0, right = cols - 1;**

**int top = 0, bottom = rows - 1;**

**while (top <= bottom && left <= right) {**

**//moving from left to right**

**for(int i=left; i<=right; i++) {**

**al.add(mat[top][i]);**

**}**

**top++;**

**//moving from top to bottom**

**for(int i=top; i <=bottom; i++) {**

**al.add(mat[i][right]);**

**}**

**right--;**

**//moving from right to left**

**if (top <= bottom) {**

**for (int i=right; i >= left; i--) {**

**al.add(mat[bottom][i]);**

**}**

**bottom--;**

**}**

**//moving from bottom to top**

**if(left <= right) {**

**for (int i=bottom; i>=top; i--) {**

**al.add(mat[i][left]);**

**}**

**left++;**

**}**

**}**

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

**}**

**}**

**String operations**

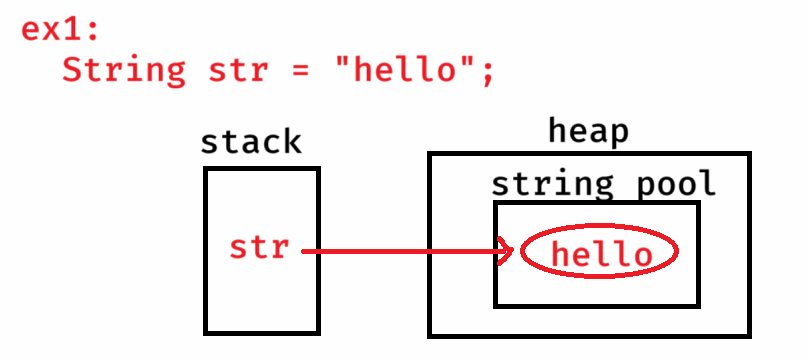
* **String is not a primtive data type. It is a class in java.lang package.**
* **There are 2 ways of creating a string object.**

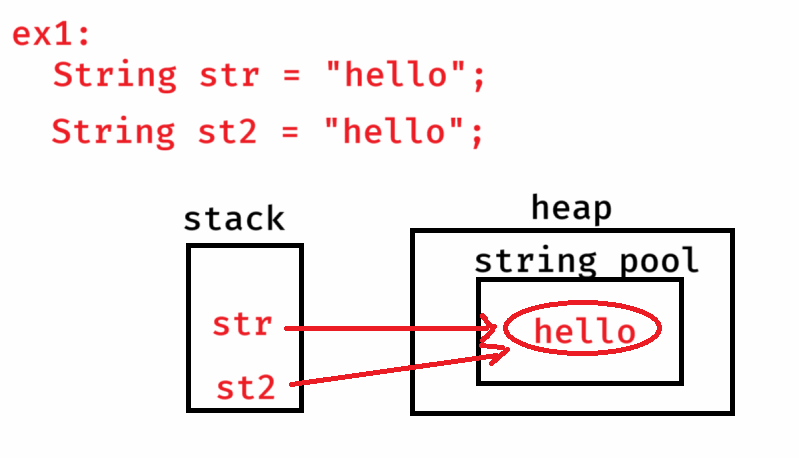
1. **String str = “hello”; //string literal**
2. **String str = new String(“hello”); //using new keyword**

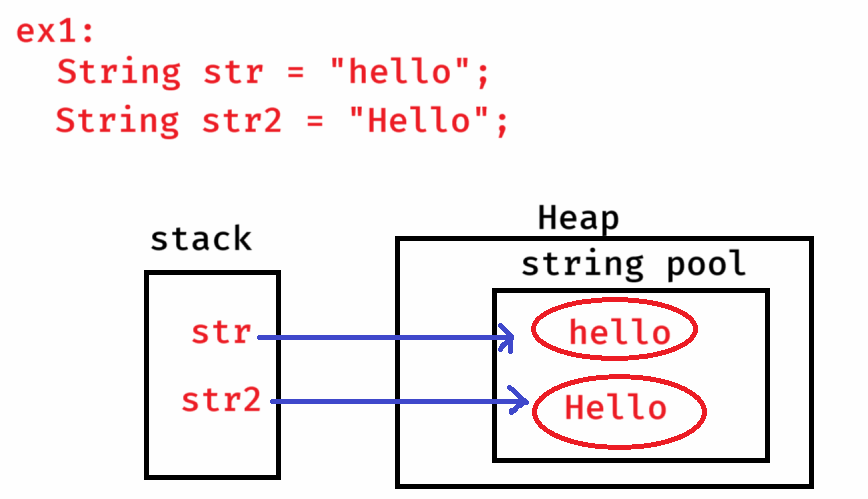
* **In Heap, a special memory area is available called string pool.**
* **When a String object is created using literal, then JVM will check**

**for the object in string pool. If exist, then it will point the variable to the same object.**

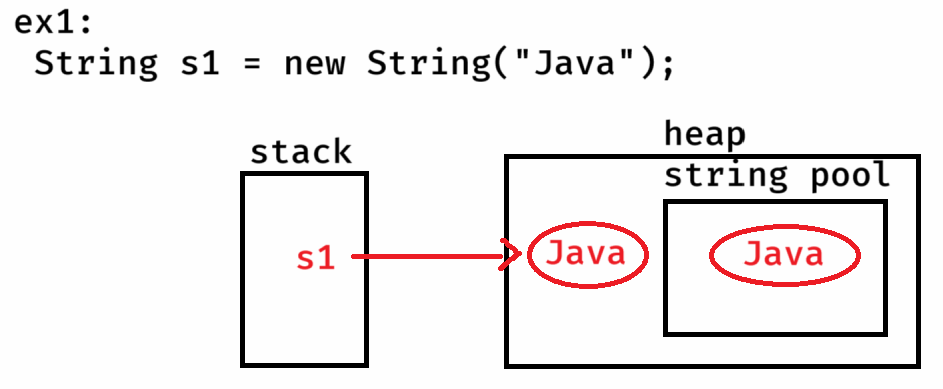
* **if doesn’t exist then JVM creates a new object in the string pool, and then will point the variable to that object.**

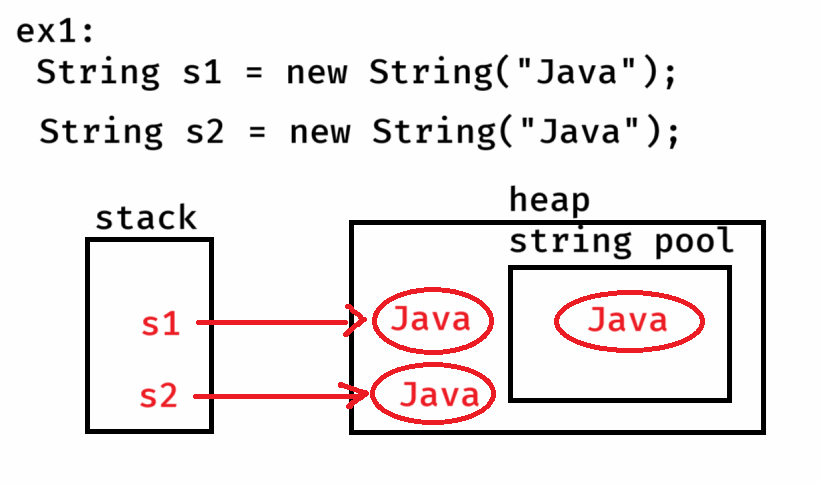
****

****

****

* **When new keyword is used, if the string object exist in the string pool, then a new string object will be created in the heap and points the variable to the object in heap.**
* **If the object doesn’t exist in string pool, then a new object is created in string pool and also in the heap.**

****

****

**Q) How many objects are created for the below code?**

**String str1 = new String(“Ashokit”);**

**String str2 = str1;**

**Ans: 2 objects**

**Q) How many objects are created for the below code?**

**String m1 = new String(“Hibernate”);**

**String m2 = new String(“JPA”);**

**Ans: 4 objects**

**Q) How many objects are created for the below code?**

**String str1 = “dell”;**

**String str2 = null;**

**Ans: 1 object.**

**Q) How many objects are created for the below code?**

**String str1 = “apple”;**

**String str2 = “null”;**

**Ans: 2 objects.**

**Q) How many objects are created for the below code?**

**String str = new String();**

**Ans: 1 object.**

**Q) How many objects are created for the below code?**

**String str = new String(null);**

**Ans: throws NullPointerException**

**find the length of a string:**

**length : It is an attribute**

**length() : It is a method**

**ex:**

**String str = “ashokit”;**

**Sop(str.length); //output: error**

**Sop(str.length()); //output: 7**

**ex:**

**String[] strArr = { “Clara”, “Joseph”, “Mary” };**

**Sop(strArr.length); //output: 3**

**Sop(strArr.length()); //output: error**

**Sop(strArr[1].length); //output: error**

**Sop(strArr[1].length()); //output: 6**

**ex:**

**String str = null;**

**Sop(str.length); //output: compile-time error**

**Sop(str.length()); //output: NullPointerException**

**strings comparision:**

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

* **== operator compares whether the two reference variables are pointing to the same object in the memory or not.**
* **If yes, then returns true. Otherwise, returns false.**

**ex1:**

**String s1 = “hello”;**

**String s2 = new String(“hello”);**

**Sop(s1 == s2); //output: false**

**ex2:**

**String str1 = new String(“Java”);**

**String str2 = str1;**

**Sop(str1 == str2); //output:** true

**ex3:**

String t1 = “sky”;

String t2 = “Sky”;

Sop(t1 == t2); //output: false

* equals() method compares the content in the string objects. If they are same then returns true. Otherwise, returns false.

ex1:

String s1 = “dell”;

String s2 = new String(“dell”);

Sop(s1 == s2); //output: false

Sop(s1.equals(s2)); //output: true

ex2:

String str1 = new String(“lenovo”);

String str2 = new String(“Lenovo”);

Sop(str1 == str2); //output: false

Sop(str1.equals(str2)); //output: false

* equalsIgnoreCase() method compares the content of the two strings, by ignoring the case. If they are same then returns true. Otherwise, returns false.

ex:

String username = “admin”;

Sop(username.equalsIgnoreCase(“Admin”)); //output: true

* compareTo() method compares the content of the two strings. If they are same then returns 0.
* If the first string is less than second string, then returns

-ve integer.

\* If the first string is greater than second string then returns +ve integer.

ex1:

String s1 = “Admin”;

String s2 = “admin”;

Sop(s1.compareTo(s2)); //output: -32

Sop(s2.compareTo(s1)); //output: +32

ex2:

String s1 = new String(“Java”);

String s2 = new String(“Java”);

Sop(s1.compareTo(s2)); //output: 0

substring method:

* substring is a some part/portion of a string.
* substring(begin, end) : returns a string from begin to end-1 index.
* substring(begin) : returns a string from begin to end of the string.

ex1:

String str = “The sky is blue”;

String str2 = str.substring(4, 10);

Sop(str2); //output: sky is

String str3 = str.substring(4);

Sop(str3); //output: sky is blue

String str4 = str.substring(4, 17); //exception: IndexOutOfBoundsException

charAt & indexOf methods:

* charAt() returns the character at the specified index.
* indexOf() returns the index of the specified character.

ex:

String str = “The cat sat”;

Sop(str.charAt(5)); //output: a

Sop(str.charAt(15)); //exception: IndexOutOfBoundsException

ex:

String str = “The cat sat”;

Sop(str.indexOf(‘t’)); //output: 6

Sop(str.lastIndexOf(‘t’)); //output: 10

ex:

String str = “The cat sat on mat”;

Sop(str.indexOf(‘t’, 7)); //output: 10

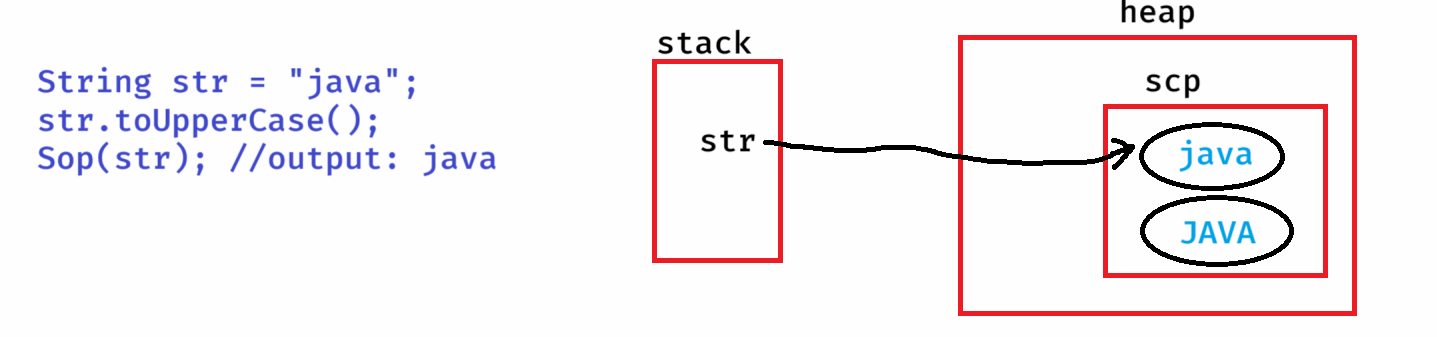
Sop(str.indexOf(‘k’)); //output: -1

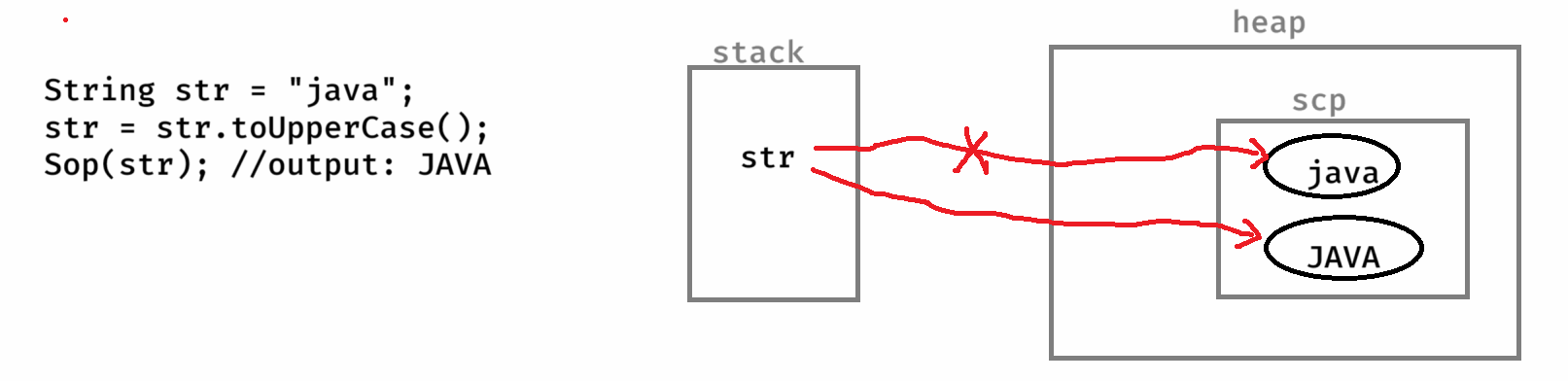
Sop(str.indexOf(“at”)); //output: 5

toLowerCase and toUpperCase methods:

* In Java, string objects are immutable objects.
* immutable object means, an object which does not allow to modify its data.
* mutable object means, an object which allows to modify its data.
* when you call toLowerCase/toUpperCase methods, JVM creates a new object for the string value, because existing object doesn’t allow to modify the data.

ex:





ex:

String str = “ASHOKit”;

str.toLowerCase();

Sop(str); //output: ASHOKit

ex:

String str = “dell”;

Sop(str.toUpperCase()); //output: DELL

Sop(str); //output: dell

replace() method:

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

* It replaces the each substring of a string with a replacement string.

replace(substring, replacement)

ex1: String str1 = “The cat sat on mat”;

str1.replace(“cat”, “dog”);

S.o.p(str1);

output: The cat sat on mat

ex2: String str1 = “The cat sat on mat”;

str1 = str1.replace(“cat”, “dog”);

S.o.p(str1);

output: The dog sat on mat

replaceAll():

replaces each substring of a string that matches the specified regular expression with a given replacement string.

replaceAll(String regex, String replacement)

ex1:

String str = “My contact number : 123-456-7890”;

String result = str.replaceAll(“\\d”, “#”);

S.o.println(str);

S.o.println(result);

output:My contact number : 123-456-7890

My contact number : ###-###-####

Note: [\\d](file:///\\d) pattern matches to any digit(0-9)

ex2:

String str = “Spring is a framework”;

String result = str.replaceAll(“\\s”, “\_”);

S.o.println(result);

output: Spring\_is\_a\_framework

Note: [\\s](file:///\\s) pattern matches to a space.

ex3:

String str = “Java is a Programming language”;

String result = str.replaceAll(“\\s+”, “ “);

S.o.println(result);

output: Java is a Programming language

Note: [\\s](file:///\\s)+ pattern matches one or more spaces

ex4:

String str = “Java#@is$great!&”;

String result = str.replaceAll(“[^a-zA-Z0-9]”, “”);

S.o.println(result);

output: Javaisgreat

Note: [^a-zA-Z0-9] pattern matches to except letters and digits.

ex5:

String str = “Ashokit Solutions”;

String result = str.replaceAll(“[aeiouAEIOU]”, “”);

S.o.println(result);

output: shkt Sltns

ex6:

String creditCard=”1234-5678-9012-3456”;

String result = creditCard.replaceAll(“[^-](?=.{4})”, “X”);

S.o.println(result);

output: XXXX-XXXX-XXXX-3456

Note: (?=.{4}) pattern checks if there are 4 characters following the current character or not.

ex7:

String str = “<h1>Hello, <b>Shekher</b>!</h1>”;

String result = str.replaceAll(“<[a-zA-Z0-9/]>”, “”);

S.o.println(result);

output: Hello, Shekher!

split() method:

* It will divide a string into an array of substrings based on a given delimiter.

ex1:

String str1 = “JSP is a technology”;

String[] str2 = str1.split(“ “);

for(String s : str2)

S.o.println(s);

output:

JSP

is

a

technology

ex2:

String str = “apple,banana,orange,grapes”;

String[] fruits = str.split(“,”);

for(String fruit : fruits) {

S.o.println(fruit);

}

output:

apple

banana

orange

grapes

ex3:

String str = “The cat sat on mat”;

String[] str2 = str.split(“t”);

for(String s : str2)

{

S.o.p(s);

}

output:

The ca

sa

on ma

join() method:

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

* It will combine multiple strings together into a single string.
* It is a static method, so we can call it with classname.

ex1:

String[] names = { “Spring”, “Boot”, “Microservices” };

String joinedString = String.join(“ “, names);

S.o.println(joinedString);

output: Spring Boot Microservices

ex2:

String[] technologies = { “JDBC”, “Servlet”, “JSP” };

String joinedString = String.join(“ and “, technologies);

S.o.println(joinedString);

output: JDBC and Servlet and JSP

trim():

-------

* It removes white space characters, before start of the first non-white space character and after the end of last non-white space character.

ex1:

String username = “ guest”;

S.o.p(username.equals(“guest”)); //output: false

S.o.p(username.trim().equals(“guest”)); output: true

ex2:

String username = “ashok it”;

S.o.p(username.equals(“ashokit”)); //output: false

S.o.p(username.trim().equals(“ashokit”)); // output: false

isEmpty() and isBlank() :

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

* isEmpty() method finds the length of a string and if it is zero then returns true, otherwise returns false.
* isBlank() method finds the length of a string, after trim and if it is zero then returns true, otherwise returns false.

ex1:

String str1 = “”;

S.o.p(str1.isEmpty()); //true

S.o.p(str1.isBlank()); //true

ex2:

String str1 = “ “;

S.o.p(str1.isEmpty()); //false

S.o.p(str1.isBlank()); //true

ex3:

String str1 = “Hello”;

S.o.p(str1.isEmpty()); //false

S.o.p(str1.isBlank()); //false

startsWith() and endsWith():

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

ex: String str1 = “ashokit@gmail.com”;

S.o.println(str1.startsWith(“gmail”)); //false

S.o.println(str1.endsWith(“gmail”)); //false

ex:

String str1 = “ASHOKIT@gmail.com”;

S.o.println(str1.startsWith(“ashok”)); //false

S.o.println(str1.toLowerCase().startsWith(“ashok”)); //true

ex:

String str1 = “ashokit@gmail.com”;

str1.toUpperCase();

S.o.println(str1.endsWith(“GMAIL.COM”)); //false

ex:

String str1 = “ashokit@gmail.com”;

str1 = str1.toUpperCase();

S.o.println(str1.endsWith(“GMAIL.COM”)); // true

intern():

This method will return the object from the string pool.

When this method is called, the variable will point to the object

in the string pool.

ex:

String s1 = "hello";

String s2 = **new** String("hello");

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

s2 = s2.intern();

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

valueOf():

* This method converts a value from any primitive data type to string type.
* It is a static method, so we have to call this method with the classname.

ex:

**int** x = 100;

i**nt** y = 200;

String str = String.*valueOf*(x) + String.*valueOf*(y);

System.***out***.println(str); //output: 100200

String str2 = String.*valueOf*(**true**) + String.*valueOf*(**false**) +

String.*valueOf*(0.0) + String.*valueOf*(1234);

System.***out***.println(str2); //truefalse0.01234

toCharArray():

. converts a string into a character array.

ex:

String str = “Java”;

char[] ch = str.toCharArray();

concat():

It will concat a given string to the existing string.

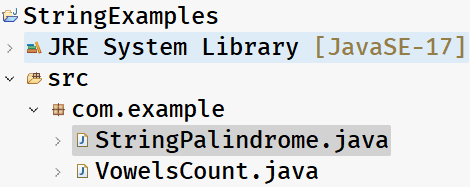
ex: String str = “Core ”;

str.concat(“Java”);

Sop(str); //output:Core

str = str.concat(“Java”);

Sop(str); //output: Core Java



StringPalindrome.java

/\*

\* This program checks whether a given string is

\* palindrome or not.

\* ex:

\* str = "madam"

\* reverse = "madam"

\* It is palindrome

\* ex:

\* str = "malayalam"

\* reverse = "malayalam"

\* It is palindrome

\* ex:

\* str = "dell"

\* reverse = "lled"

\* It is not a palindrome

\*/

package com.example;

import java.util.Scanner;

public class StringPalindrome {

private static boolean isPalindrome(String str) {

//convert the string to lowercasae

str = str.toLowerCase();

//convert the string to char[]

char[] ch = str.toCharArray();

boolean flag = true;

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

{

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

{

flag = false;

break;

}

}

return flag;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String str = scanner.nextLine();

boolean flag = isPalindrome(str);

if(flag == true)

System.out.println("It is a palindrome");

else

System.out.println("It is not a palindrome");

scanner.close();

}

}

VowelsCount.java

**/\***

**\* This program is to count the vowels and**

**\* consonants in a given string.**

**\* example:**

**\* str = "Lilliput@River2"**

**\* output:**

**\* vowels count : 5**

**\* consonants count : 8**

**\*/**

**package com.example;**

**import java.util.Scanner;**

**public class VowelsCount {**

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

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

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

**String str = scanner.nextLine();**

***countVowels*(str);**

**scanner.close();**

**}**

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

**// TODO Auto-generated method stub**

**// convert the string to lowercase**

**str = str.toLowerCase();**

**// convert the string to char[]**

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

**int vowelsCount = 0;**

**int consonantsCount = 0;**

**// loop**

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

**if (Character.*isLetter*(ch[i])) {**

**switch(ch[i]) {**

**case 'a':**

**case 'e':**

**case 'i':**

**case 'o':**

**case 'u':**

**vowelsCount++;**

**break;**

**default:**

**consonantsCount++;**

**}**

**}**

**}**

**System.*out*.println("vowels count = " + vowelsCount);**

**System.*out*.println("consonants count = " + consonantsCount);**

**}**

**}**

Anagrams.java

**/\*\***

**\* This program is to check whether the**

**\* given two strings are anagrams or not.**

**\* example:**

**\* str1 = "state"**

**\* str2 = "taste"**

**\* output: Anagrams**

**\* str1 = "Java"**

**\* str2 = "Python"**

**\* output: Not Anagrams**

**\*/**

**package com.example;**

**import java.util.Arrays;**

**import java.util.Scanner;**

**public class Anagrams {**

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

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

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

**String str1 = scanner.nextLine();**

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

**String str2 = scanner.nextLine();**

**boolean flag = *checkAnagrams*(str1, str2);**

**if (flag == true)**

**System.*out*.println("Yes, the strings are anagrams");**

**else**

**System.*out*.println("No, the strings are not anagrams");**

**scanner.close();**

**}**

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

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

**{**

**return false;**

**}**

**//convert the strings to lowercase**

**str1 = str1.toLowerCase();**

**str2 = str2.toLowerCase();**

**//convert the strings to char[]**

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

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

**//sort them**

**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;**

**}**

**}**

**return flag;**

**}**

**}**

Frequency.java

/\*\*

\* This program finds the frequency of each character

\* in a given string.

\* example:

\* str = "MiSSisiPi"

\* output:

\* i - 4

\* m - 1

\* p - 1

\* s - 3

\*/

package com.example;

import java.util.Arrays;

import java.util.Scanner;

public class Frequency {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String str = scanner.nextLine();

frequency(str);

scanner.close();

}

private static void frequency(String str) {

// TODO Auto-generated method stub

str = str.toLowerCase();

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

{

break;

}

}

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

}

}

}

FirstNonRepeatingCharacter.java

**/\*\***

**\* This program will print the first non-repeating**

**\* character of a string.**

**\* example:**

**\* str = "swiss"**

**\* output: w**

**\*/**

**package com.example;**

**import java.util.Scanner;**

**public class FirstNonRepeatingCharacter {**

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

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

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

**String str = scanner.nextLine();**

***nonRepeatingChar*(str);**

**scanner.close();**

**}**

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

**// TODO Auto-generated method stub**

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

**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;**

**}**

**}**

**if(count == 1) {**

**System.*out*.println("The first non-repeating char : " + ch[i]);**

**break;**

**}**

**}**

**}**

**}**

StringBuffer class

* String, StringBuffer and StringBuilder are the independent classes from java.lang package and these classes are use to perform string operations.
* StringBuffer class object is a mutable and a thread-safe object.
* StringBuffer class object can be created only with new keyword.

ex:

StringBuffer sb = “hello”; // error

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

* when a StringBuffer object is created, the JVM checks for the exisiting object in the SCP for the string value. If already exists, then creates a new object in Heap and makes the variable to point to the object in heap.
* If not exist, then creates a new object in SCP, then a new object in Heap, then makes the variable pointing to the heap object.
* While creating a StringBuffer object, JVM allocates a 16 characters buffer for the existing value in the StringBuffer object.
* For example:
* StringBuffer sb =new StringBuffer(“hello”);
* length of the StringBuffer object: 5
* capacity of the StringBuffer object: 5 + 16 = 21
* If the length of the StringBuffer object crosses the capacity, then the capacity will be resized with the formula,

oldcapacity \* 2 + 2

* Suppose, if 17 more characters are appended to the above StringBuffer object, then

length : 22

capacity: 21 \* 2 + 2 = 44

How a StringBuffer object is thread-safe?

* StringBuffer is a mutable object, but it doesn’t allow multiple threads exactly at the same time to make the changes.
* If first thread is allowed to make the changes, then second thread has to wait.
* So, a StringBuffer object is a thread safe object.

comparing StringBuffer objects:

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

ex:

StringBuffer sb1 = new StringBuffer(“Java”);

StringBuffer sb2 = new StringBuffer(“Java”);

S.o.println( sb1.equals(sb2) ); //output: false

Note: equals() method in StringBuffer also works like == operator.

* If you want to compare the StringBuffer objects, you have to

convert the StringBuffer objects into String objects again.

ex:

String s1 = sb1.toString();

String s2 = sb2.toString();

S.o.p( s1.equals(s2)); // output: true

some key methods of StringBuffer class:

1. append(): appends the given value to the existing object.

ex:

String s1 = new String(“Core “);

s1.concat(“Java”);

S.o.println(s1); // output:Core

StringBuffer sb = new StringBuffer(“Core “);

sb.append(“Java”);

S.o.println(sb); // output:Core Java

1. insert(index, String): inserts the given string at the given index.

ex:

StringBuffer sb1 = new StringBuffer(“The sat on mat”);

sb1.insert(4, “dog “);

S.o.println(sb1); // output:The dog sat on mat

ex:

StringBuffer sb2 = new StringBuffer(“The cost is : rupees”);

sb2.insert(14, 100);

S.o.println(sb2);

output:The cost is : 100 rupees

1. delete(start, end):Removes the characters from start to end-1.

ex:

StringBuffer sb1 = new StringBuffer(“The sky is blue”);

sb1.delete(0, 4);

S.o.println(sb1); // output:sky is blue

ex:

StringBuffer sb2 = new StringBuffer(“hello”);

sb2.delete(2, 9);

S.o.println(sb2); // output:he

ex:

StringBuffer sb3 = new StringBuffer(“Lilliput”);

sb3.delete(5, 3); // StringIndexOutOfBoundsException

sb3.delete(5, 8);

S.o.println(sb3); // output:Lilli

1. replace(start, end, str) : replaces the content between start to end – 1 with the given string.

ex:

StringBuffer sb1 = new StringBuffer(“The sky is blue”);

sb1.replace(11, 15, “black”);

S.o.println(sb1); //output:The sky is black

1. reverse() : reverse the characters of a given string value.

ex:

StringBuffer sb1 = new StringBuffer(“level”);

if(sb1.reverse().equals(“level”))

S.o.p(“palindrome”);

else

S.o.p(“not a palindrome”);

output: not a palindrome

ex:

StringBuffer sb1 = new StringBuffer(“level”);

if(sb1.reverse().toString().equals(“level”))

S.o.p(“palindrome”);

else

S.o.p(“not a palindrome”);

output: palindrome

ex:

StringBuffer sb2 = new StringBuffer(“Madam”);

if(sb2.reverse().toString().toLowerCase().equals(“madam”))

Sop(“palindrome”);

else

Sop(“not a palindrome”);

output: palindrome

ex:

StringBuffer sb3 = new StringBuffer(“Liril”);

sb3.reverse();

String s1 = sb3.toString();

s1.toLowerCase();

if(s1.equals(“liril”))

S.o.p(“palindrome”);

else

S.o.p(“not a palindrome”);

output:

not a palindrome

Compression.java

**/\*\***

**\* This program prints the string compression.**

**\* example:**

**\* str = "aabccdeaa"**

**\* output: a2b1c2d1e1a2**

**\*/**

**package com.example;**

**import java.util.Scanner;**

**public class Compression {**

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

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

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

**String str = scanner.nextLine();**

***compress*(str);**

**scanner.close();**

**}**

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

**StringBuffer buffer = new StringBuffer();**

**int count = 1;**

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

**if ( i + 1 < str.length() && str.charAt(i) == str.charAt(i+1)) {**

**count++;**

**}**

**else {**

**buffer.append(str.charAt(i)).append(count);**

**count = 1;**

**}**

**}**

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

**}**

**}**

StringBuilder class:

* StringBuilder is a class provided in java.lang package.
* StringBuilder class also performs operations on string values.
* StringBuilder is a mutable and not a thread-safe object.
* In Multithreaded applications, if StringBuilder object is shared with multiple threads and if the threads are modifying the value exactly at the same time then unexpected results will be produced.
* So, StringBuilder is recommended to use in Single thread applications.
* StringBuilder object can also be created with new keyword only.

ex:

StringBuilder builder = “apple”; // error

StringBuilder builder = new StringBuilder(“apple”); //correct

* Since, StringBuilder is also a mutable object, additional space for 16 characters will be allocated, to allow the changes to the value.
* StringBuilder class also contains the same methods like StringBuffer class, but StringBuffer class methods are synchronized methods and StringBuilder class methods are not synchronized.

Q) what is the difference between String and StringBuffer?

Ans: String is immutable object, StringBuffer is mutable object.

Q) what is the difference between String and StringBuilder?

Ans: String is immutable and thread-safe object. StringBuilder is mutable and not a thread-safe object.

Q) what is the difference between StringBuffer and StringBuilder?

Ans: StringBuffer is mutable and thread-safe object. StringBuilder is mutable but not a thread-safe object.

Q) why Java is called Robust language?

Ans: 1. A language can be called as a Robust langugage, based on two parameters.

1. The language should provide Garbage Collection mechanism.
2. The language should provide Error handling support.

2. In Java, we have automatic garbage collection mechanism and also exception handling. So, Java can be called as a Robust language.

Q) why Java is called a secured language?

Ans: 1. When a Java program is compiled, we get byte code.

2. When byte code is loaded into JVM, there is a byte code

verifier in JVM, who checks whether this byte code is

original or tampered by some one in the middle.

1. If tampared, then exceptions/error will be thrown.
2. So, a hacker can’t hack the Java Byte code. So, we call

as a secured language.

Q) why Java is called multithreaded language?

Ans: Java allows multiple tasks to execute parallelly within a single application, by creating multiple threads. So, Java is called mulithreaded language.

Q) Why Java is called distributed language?

Ans: Java provides built-in support for developing applications, where different parts of the application run on a different computer in a network and they are all connected together, and provides the services to the end-users/customers. So, Java is called distributed language.

Object Oriented Programming System(oops)

why OOP?

* To develop a software application, a programmer/developer has to choose one of the two programming models(paradigms).
* 1. procedure oriented programming model
* 2. object oriented programming model.
* Procedure oriented programming languages are a good fit for developing System-level applications.
* examples:
* operating systems
* database engines
* microcontrollers
* device drivers, etc..
* But, Procedure oriented programming languages are not a good fit for developing web and enterprise applications.
* Reasons?
* 1. No built-in support for protocols like HTTP, FTp, SMTP etc..
* 2. No built-in support for Database access
* 3. No built-in support for multithreading
* 4. No built-in support for error handling
* 5. No real-world mappings(like customer, products, employees, etc..)
* 6. No Garbage collection support, etc..
* So, Object oriented programming languagues were introduced to develop the web and enterprise applications.

OOP principles:

* OOP principles are provided to make a web/enterprise application as maintainable, reusable, scalable, secured etc..
* OOP principles are four.
* 1. Encapsulation
* 2. Abstraction
* 3. Inheritance
* 4. Polymorphism

Encapsulation:

* Encapsulation means, combine the data and related functionalities together at one place as a single unit.
* Don’t allow other components from assinging invalid data.
* So, to implement this principle, you should create a class with private variables and public setter/getter methods.

Abstraction:

* Abstraction means, it process of providing the essential data and hiding the un-essential data from the user of the system.
* A general example of abstraction is,
* i) you are a user of a TV Remote
* ii) The essential data provided to the user, the buttons

to perform actions like switch on/off, volume incr/decr,

etc..

iii) The un-essential data hidden is like, how signals are sent from remote to TV and how they are processed by the TV.

iv) This is called Abstraction.

* To implement this Abstract principle, you should use abstract classes and interfaces of Java programming.

Inheritance:

* Inheritance means, creating new types/classes by inheriting from already existing types/classes.
* The newly created classes are called child classes and already exisiting classes are called parent classses.
* To implement this inheritance principle, you should use the keywords like “extends” and “implements” of Java.

Polymorphism:

* Polymorphism means, providing many forms to perform one action or one action should exihibit multiple behaviours.
* For example, one action is opening a bank account and many forms to perform this action are, through Adhaar Card, or through PAN Card, or through VoterId, etc.. This is polymorphism.
* Overloading and overriding are the two mechanisms provide by Java to implement polymorphism.
* Note:
* Polymorphsim is a combination of two Greek words poly means “Many” and morphism means “forms”.

class and object:

* In terms OOP, everything in the real-world is considered as an object.
* The objects are divided into different groups based on their attributes/properties and/or behavior.
* Now I should create a template/blueprint/plan with the properties and behaviour, for each group of objects.
* For example, In e-commerce application, we have different groups of objects like customers, products, orders, payments, employees, etc…. So, for each group of objects, we should create a template to reprent their properties and behaviour.
* These templates/blueprints/plans are called as classes.
* class is a keyword in Java, which is derived from another word called “classification”.
* definition of a class:
* A class is a template which contains variables to represent the properties and methods to represent the behaviours of a certain group of objects.
* A class is a blueprint with a collection of variables and methods, to represent a group of objects.
* A class is a container with data members and methods, to represent a certain group of objects.
* syntax of a class:
* -----------------
* <access specifier> <access modifier> class <classname>
* {
* variables;
* methods;
* }
* example:
* class Person {
* String name;
* int age;
* void eating() {
* // code
* }
* void talking() {
* // code
* }
* }
* definition of an object:
* It is an instance of a class.
* syntax:

ClassName objectName = new ClassName();

ex:

Person p1 = new Person();

1 2 3 4

1. It is the type of the variable. When you create a class, then it automatically becomes as a datatype(user-defined).
2. It is the reference variable, which stores the address of the object.
3. It is a memory allocation operator. It uses the class/bluepring to allocate the memory space in heap.
4. constructor call

All together:

. Java uses Person as a blueprint for allocating the memory space, initializes the object, and stores the address of that object into a reference variable called p1.