Java FullStack Development Program

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

Time: 6PM – 7:30PM

Duration: 6 Months

Fee : 30K

(Live classes + Class Notes + Class videos with

1 year validity + Mock interviews + Placement

assitance)

FullStack = Front-end + Back-end + Database + Cloud

Example:

* You have opened amazon.in website, then you will get a web page.
* This web page will display the information about the products to the user.
* This web page will also take input from the user to search for the products, then it will display the required information to the user.
* So, a web page can do 2 things.

1. It can display the information to the user.
2. It can take the input from the user.

* To develop these web pages, you have to use Front-end technologies.
* example: HTML(Hyper Text Markup Language)

CSS (Cascading Style Sheet)

Javascript

Bootstrap

React JS

* End-users will interact with the webpages.
* Amazon sells millions of products to the customers.
* So, do they have developed millions of web pages?
* No.
* The details of each product, images of each proudct will be stored in the Database.
* Database is used to store the large amount of information and will provide the information whenever it is required.
* When you/user clicks on a product in the webpage, a request goes to back-end application, and this back-end application will read the product details from the database.
* After that, it will apply the tax calculations, discounts etc..

and then it will provide the details to the front-end.

* So, now the front-end will display a web page with all the product details.
* This back-end program will be developed in the technologies like Java, Python, .Net, etc..
* So, in the Amazon.in website, front-end development, back-end development and database are available.
* So, it is a FullStack application.

JAVA

========

* Java Editions:

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

* Java SE is a module of Java, which can be used to develop the stand-alone applications/desktop applications.
* Java SE is the base module for every Java Programmer.
* Java SE module is also known as Core Java.
* Java EE is another module of Java, which can be used to develop web applications and enterprise applications.
* Web applications means, the small scale website like restaurent websites, or hotel booking websites, ITtrainings websites. etc…
* Enterprise applications means, the large scale websites like irctc, banking websites, stock trading applications, etc...
* Java ME is a module of Java, which is used for developing mobile applications, device applications, infotainment systems, etc..
* For a FullStatck Java developer, Java SE and Java EE modules are required.
* In Java EE module, you will learn,

. Advanced Java Technologies(JDBC, Servlet, JSP)

. Spring Framework

. Hibernate

. Spring Boot

. Microservices

* Java is a most popular programming language, because of its features like, security, platform-independency, Multithreading etc..
* The first name of Java was “OAK”.
* In 1991, OAK language development started, and it is renamed to JAVA in 1995 and released to the developers in 1996.
* The father of JAVA language is “James Gosling”.
* Java language was released with a slogan “Write Once, Run Anywhere(WORA)”.

JDK, JRE and JVM:

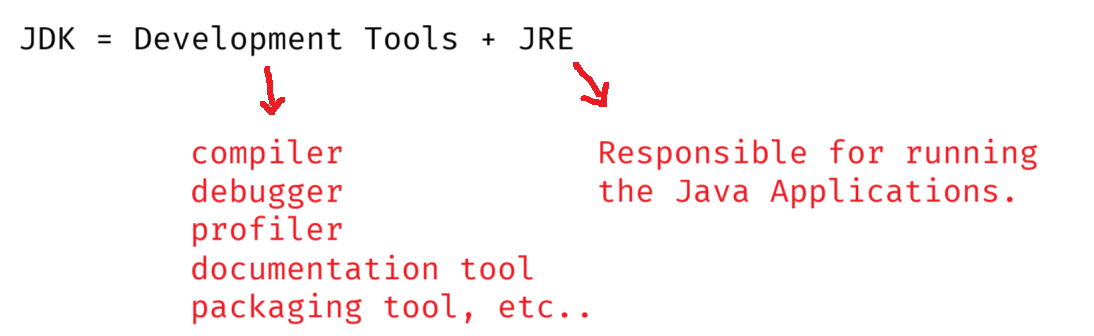
=================

JDK : Java Development Kit

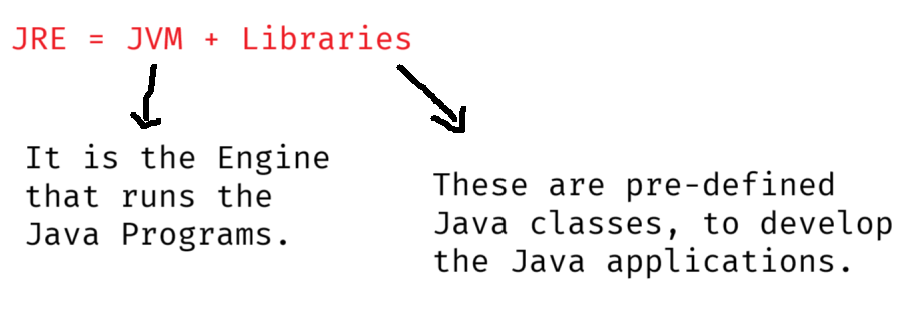
JRE : Java Runtime Environment

JVM : Java Virtual Machine

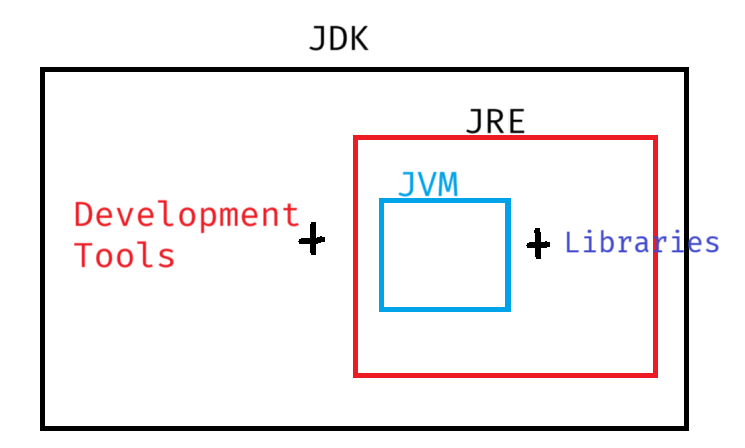
* JDK is the name of the Java Software and we need install this software to write, compile and run the Java programs.
* JDK is the complete package for Java Development.

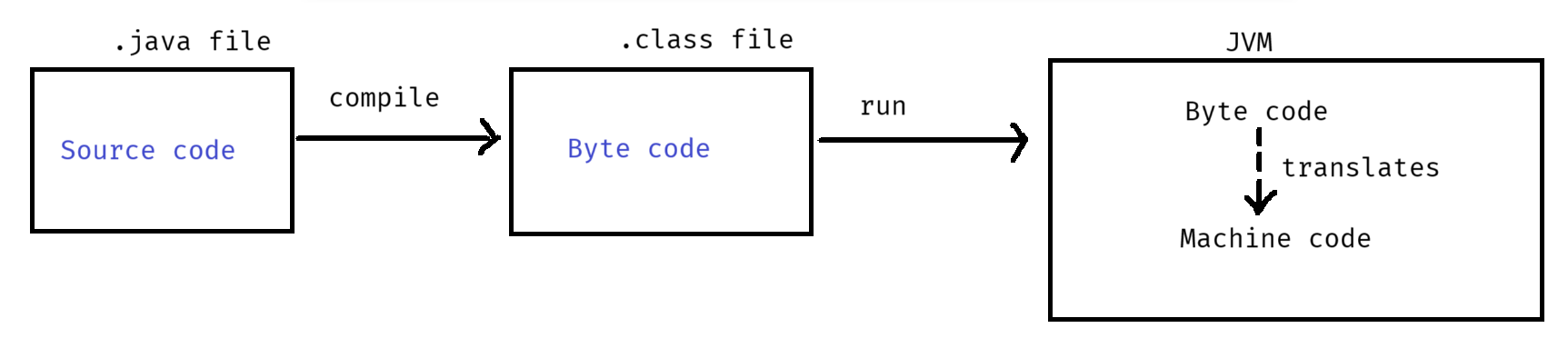


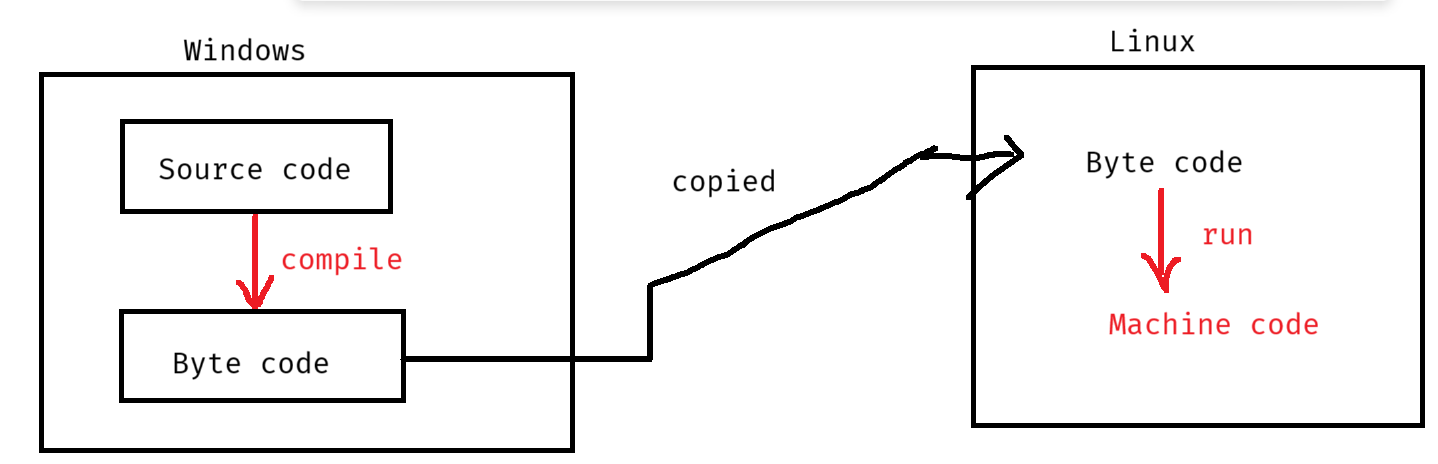
* JRE allows Java programs to run.
* JRE does NOT contain compiler. So, it can’t be used for compilation. It is used only for execution.



* JVM is the heart of Java. It will translate Byte code instructions into Machine code instructions.







Installing Java SE 21 :

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

1. visit <https://www.oracle.com/java/technologies/javase/jdk21-archive-downloads.html>
2. Downloaded Windows X64 msi Installer
3. A file jdk-21.0.5\_windows-x64\_bin.msi will be downloaded.
4. Double click on the downloaded file, follow Next buttons, then close.
5. Check for the JDK-21 at C:\Program Files\Java folder.

Path Setting:

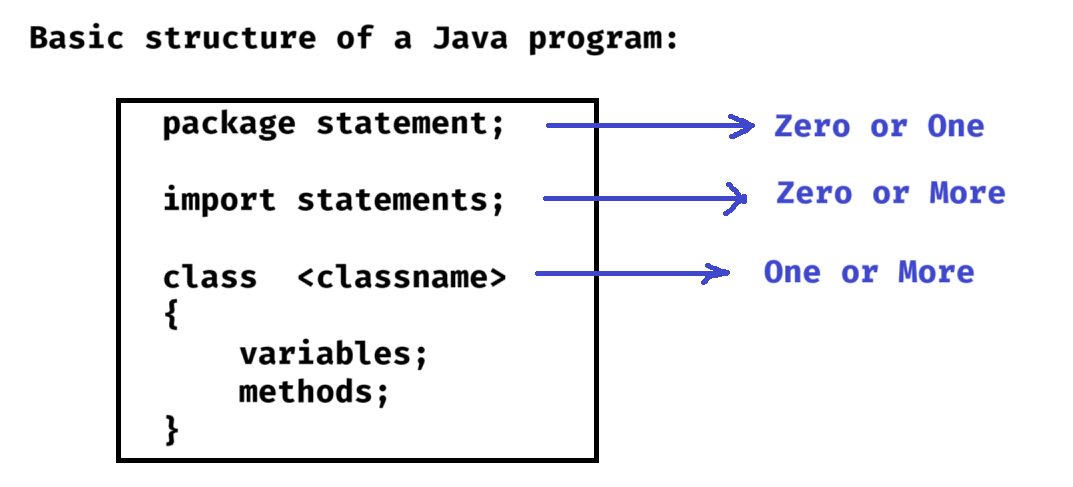
------------

1. Windows Search 🡪 environment 🡪 choose Edit system environment variables 🡪 click on Environment variables button 🡪 choose Path from system variables 🡪 Edit..
2. Click on New button, copy the value : C:\Program Files\Java\jdk-21\bin.
3. Move this Java path to the first place, with Move Up button.
4. click OK.

Note: To verify the path setting is done correctly or not,

* open command prompt, and type the following command.

java --version



First Java Program:

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

1. open notepad
2. type the below code.

class MyFirstProgram

{

public static void main(String[] args)

{

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

}

}

1. File 🡪 Save 🡪 select D:\ drive 🡪 change save as type : all files 🡪 enter filname: MyFirstProgram.java 🡪 save.
2. open command prompt
3. compile the program

D:\>javac MyFirstProgram.java

1. run the program.

D:\>java MyFirstProgram

output: Hello, Welcome to Java

keywords, identifiers and data types:

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

keyword:

* Keywords are also called reserved words, and these words have a pre-defined meaning defined by the programming language.
* You cannot use these keywords for your variables or methods or classes.
* In Java, we have more than 55 keywords.

examples:

class, public, private, static, final, abstract, return, break, continue, if, switch, for, while, volatile, transient,..

identifier:

* An identifier is a name assigned to a variable or a method or a class.
* An identifier is a user-defined word.
* Rules for identifiers are,

1. can contain letters(A-Z, a-z), digits(0-9), underscore(\_) and dollar sign($).
2. cannot start with a digit.
3. cannot be a Java keyword.
4. cannot contain a whitespace.
5. case-sensitive.
6. should be descriptive.

examples:

int \_speed1 = 80; // \_speed1 is an identifier

int \_speed2 = 180; // \_speed2 is an identifier

String carName = “Tesla”; // carName is an identifier

int tot@l = 100; //error

int 7wonders = 7; //error

void accelerate() // accelerate is an identifier

{

//logic

}

datatype:

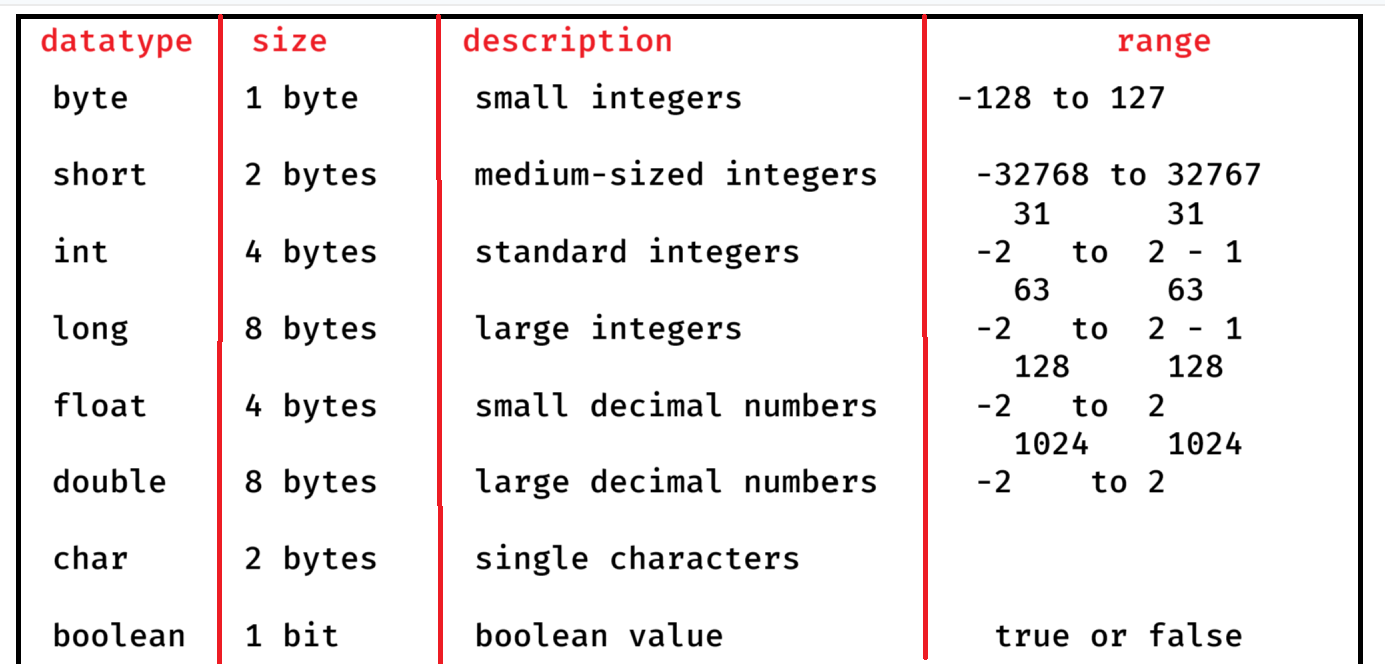
--------

* A data type defines the kind of data that a variable can hold/store.
* Java is a strongly typed programming language. It means, every variable must have a data type.
* Java has two categories of data types.

1. primitive data types
2. non-primitive/reference data types

* primitive data types stores simple values.
* reference data types stores objects.

Primitive Data Types:





ex1:

byte b1 = -108; // valid

byte b2 = 126; // valid

byte b3 = 110; // valid

byte b4 = b2 + b3; // error

ex2:

float x = 3.64; //error

float x = 3.64F; //valid

float x = 3.64f; //valid

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

Install Eclipse IDE(Integrated Development Environment):

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

1. visit eclipse.org/downloads
2. click on Download packages link.
3. Download Eclipse IDE For Enterprise Java and Web Developers
4. A zip file, eclipse-jee-2025-03-R-win32-x86\_64.zip is downloaded
5. Extract the zip file.
6. A folder, eclipse-jee-2025-03-R-win32-x86\_64 is created.
7. Open this folder, next open eclipse folder, then you can find eclipse application, to start the Eclipse IDE.

First program in Eclipse:

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

1. create a workspace directory in the computer.
2. Launch/start Eclipse.
3. click on browse and choose workspace.
4. click on File 🡪 New 🡪 Project... 🡪 Java Project 🡪 Next 🡪 Project Name: DemoApplication

Uncheck Module checkbox

Next 🡪 finish.

1. Expand DemoApplication 🡪 src 🡪 right click 🡪 New 🡪 package 🡪 enter package name: com.ashokit 🡪 finish.
2. com.ashokit 🡪 right click 🡪 New 🡪 class 🡪 Name: Test 🡪finish.
3. Type the below code:

**package** com.ashokit;

**public** **class** Test {

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

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

System.***out***.println("This is my first program in Eclipse");

}

}

1. Test.java 🡪 right click 🡪 RunAs 🡪 Java Application.

output:

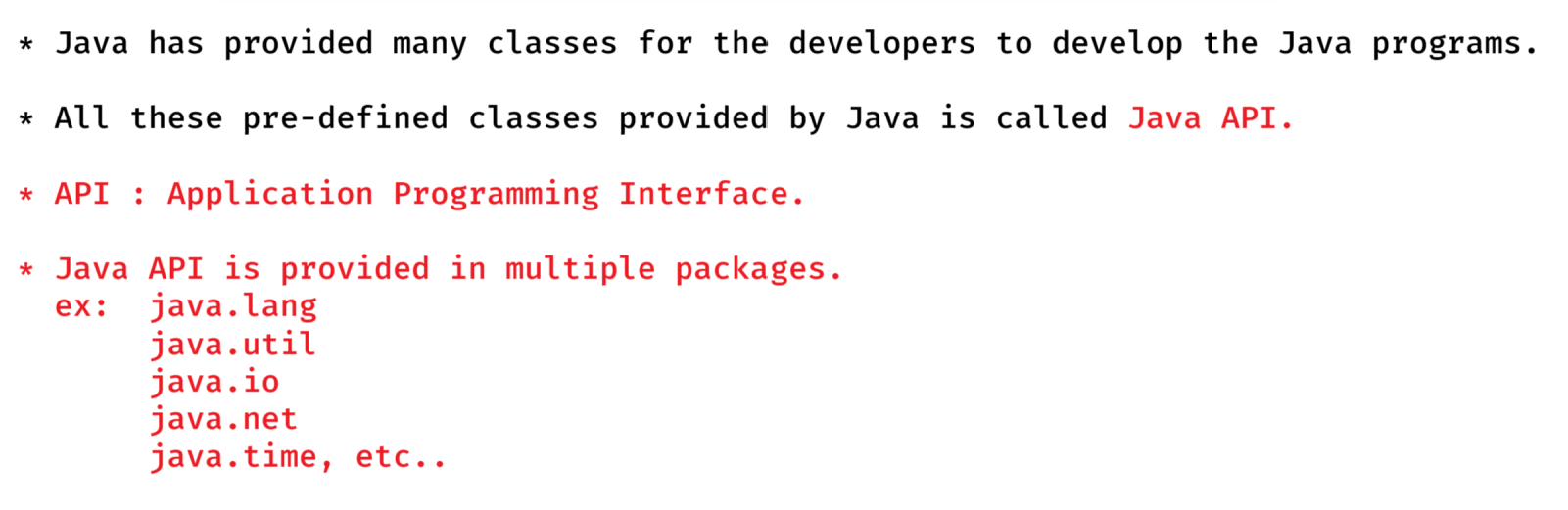
Hi

This is my first program in Eclipse

Reading the user’s input:

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

* To read the input provided by the user through the keyboard, Java has provided a pre-defined class called Scanner class.



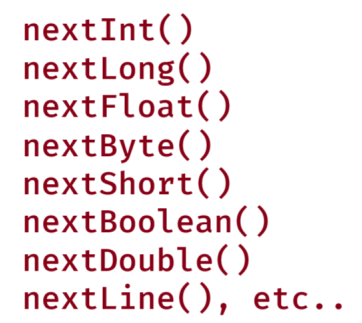
* To read the input values given by the user, in the program you have to create Scanner class object like below.

Scanner scan = new Scanner(System.in);

* Before we create the Scanner class object, first we have to import the Scanner class into our Java program.

import java.util.Scanner;

* The below methods are provided in the Scanner class, to read the different types of input.



Java comments:

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

* comments are nothing but they are the descriptions, written above the variables, or methods, or classes or some logic, etc..
* The comments will help the others to understand the code easily.
* In Java, we have 3 types of comments.

//single line comment

/\*

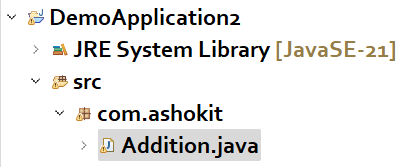
multiple lines comment

\*/

/\*\*

multiple lines comment

\*/



Addition.java

-----------

/\*

\* This program reads two integer values from

\* the user, then finds the addition.

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** Addition {

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

// create Scanner class object

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

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

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

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

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

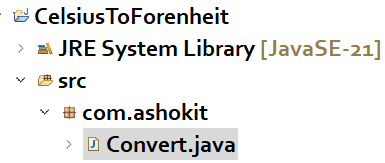
//calculate addition

**int** c = a + b;

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

}

}



Convert.java

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

/\*

\* This program converts the given celsius value

\* to the forenheit value.

\* formula:

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

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** Convert {

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

//create a Scanner class object

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

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

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

//calculation

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

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

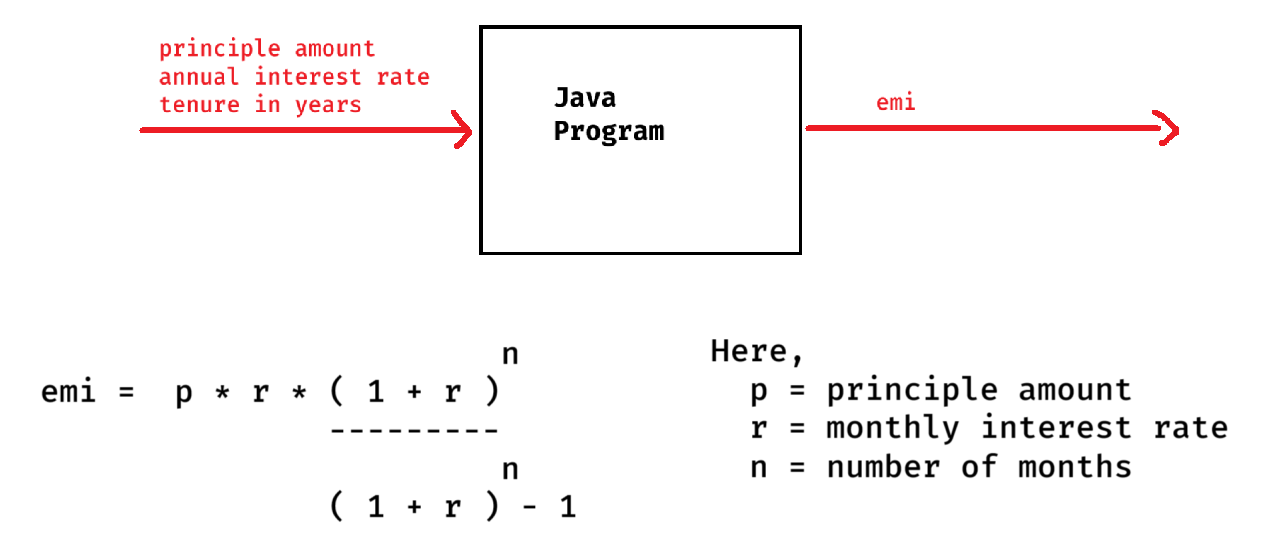
scan.close();

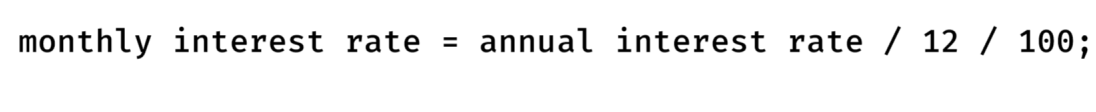
}

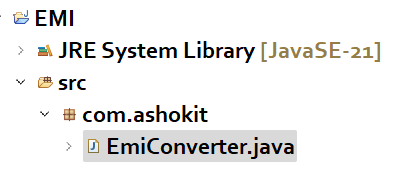
}

EMI(Equated Monthly Installment):

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







EmiConverter.java

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

/\*

\* This program takes principle amount, annual interest rate,

\* tenure in years as input, then converts annual interest rate

\* to monthly interest rate, converts tenure from years to months

\* then calculate emi.

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** EmiConverter {

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

//create Scanner class object

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

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

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

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

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

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

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

//convert annual interest rate to monthly interest rate

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

//convert years to months

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

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

//calculate emi

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

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

scan.close();

}

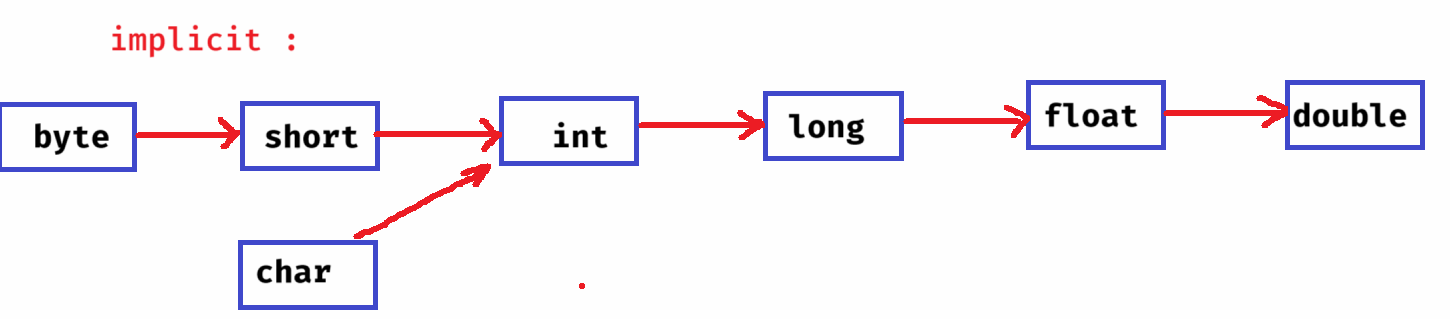
}

Type casting:

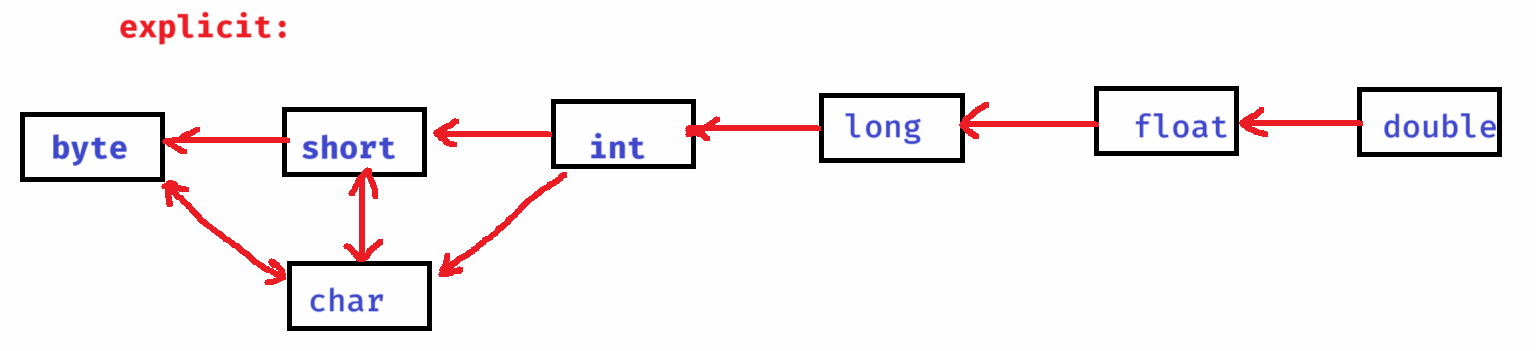
* converting a value from one primitive data type to another primitive data type is called type casting.
* Type casting is of 2 types.

1. implicit type casting / up-casting
2. explicit type casting / down-casting

* implicit type casting means, converting a value from low memory size to high memory size or same memory size.



* explicit type casting means, converting a value from high memory size to low memory size or same memory size.



ex1:

int x = 3;

float y = x; //implicit

S.o.p(y); //output: 3.0

ex2:

char ch = ‘A’;

int k = ch; //implicit

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

ex3:

float a = 4.53f;

int b = (int) a; //explicit

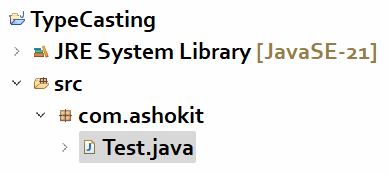
S.o.p(b); //output: 4

ex4:

short s = 97;

char ch = (char) s; //explicit

S.o.p(ch); //output: a



Test.java

--------

//A sample type casting program.

**package** com.ashokit;

**public** **class** Test {

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

**int** i = 23;

**float** j = i; //implicit

System.***out***.println("int to float : " + j);

**double** x = 46.87;

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

System.***out***.println("double to int : " + y);

**char** ch = 'w';

**int** a = ch; //implicit

System.***out***.println("char to int : " + a);

**short** s = 68;

**char** c = (**char**) s; //explicit

System.***out***.println("short to char : " + c);

}

}

Q) **For which data type, type casting is not possible?**

**A) boolean**

Operators

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

* operators are 3 types.

1. unary operators
2. binary operators
3. ternary operator.

* unary operators works with single operand.
* binary operators works with two operands.
* ternary operator works with three operands.

unary operators:

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

* 1. increment operator( ++ )
* 2. decrement operator( -- )
* increment operator increments the value by 1 and decrement operator decrements the value by 1.

ex:

a++ 🡪 post increment.

increment after the use

++a 🡪 pre increment.

increment before the use

a-- 🡪 post decrement.

decrement after the use

--a 🡪 pre decrement

decrement before the use.

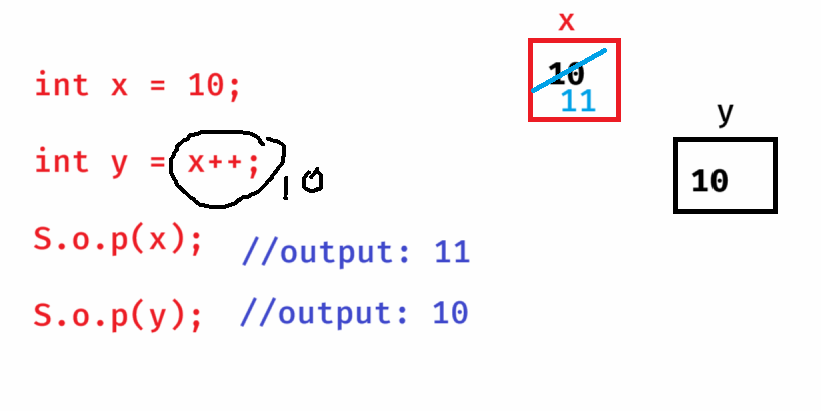
ex1:

int x = 10;

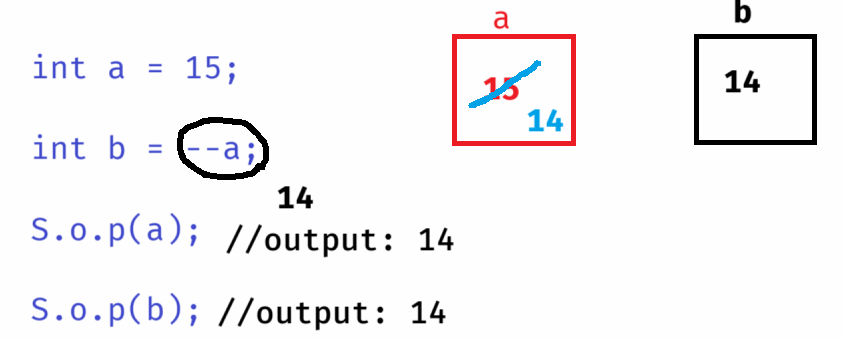
int y = x++;

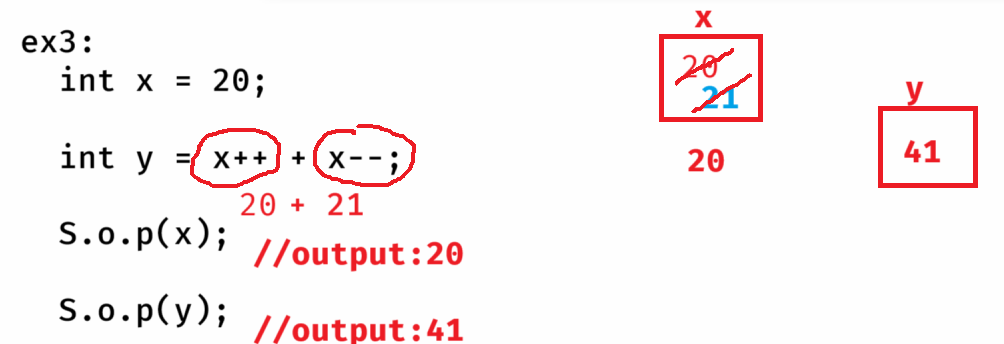
S.o.p(x);

S.o.p(y);



ex2:





Binary operators:

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Bitwise operators

Arithmetic operators:

+ addition

* substraction

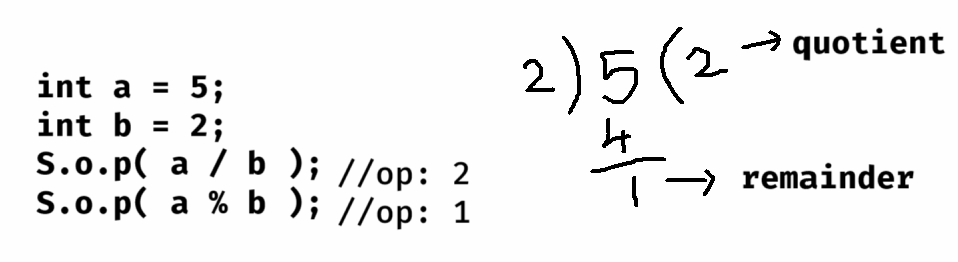
\* multiplication

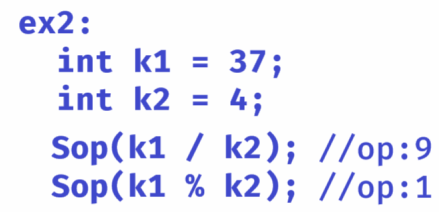
/ division

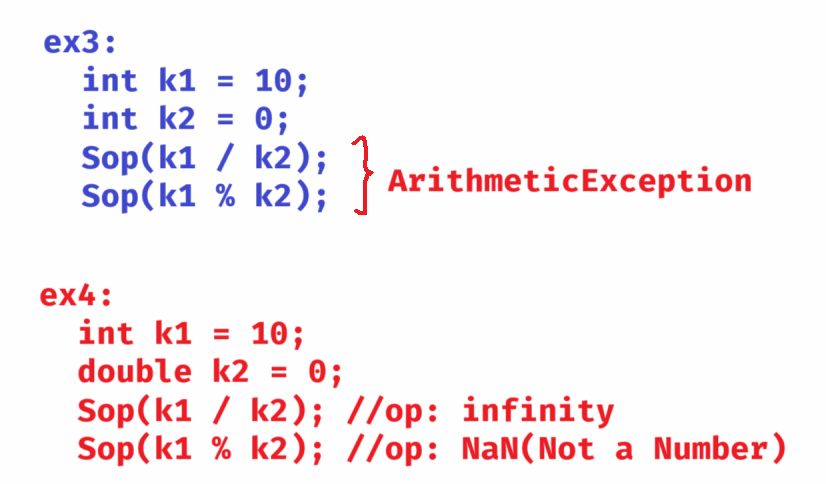
% modulus

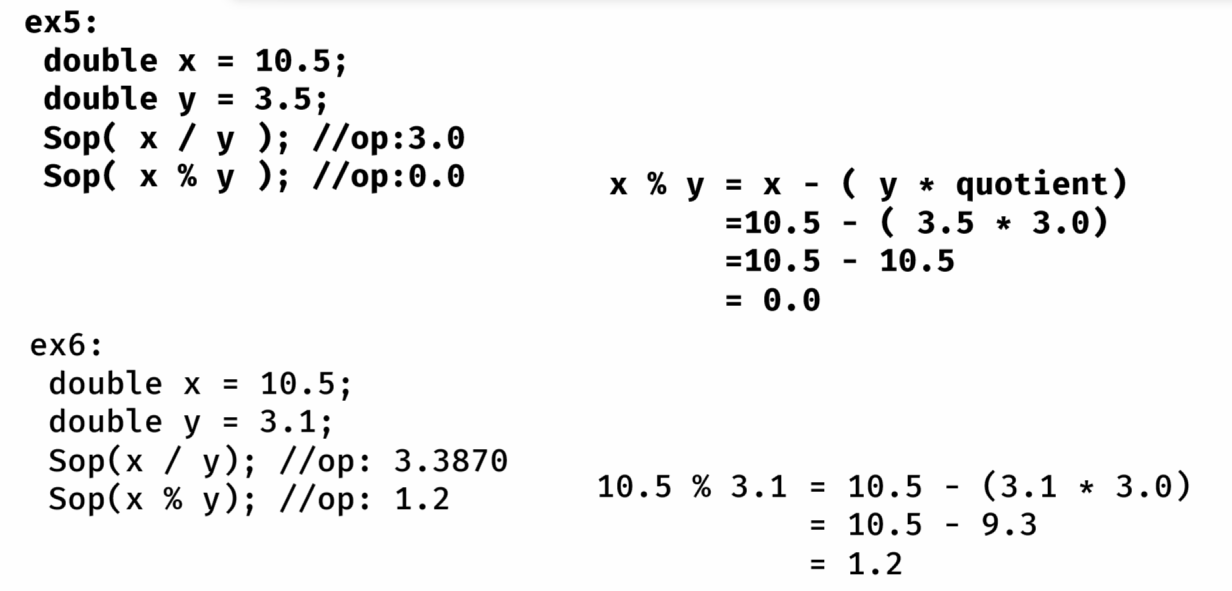
Note: division(/) retrurns quotient and modulus(%) returns remainder.

ex1:









Relational operators:

< less than

> greater than

<= less than or equals

>= greater than or equals

== equals

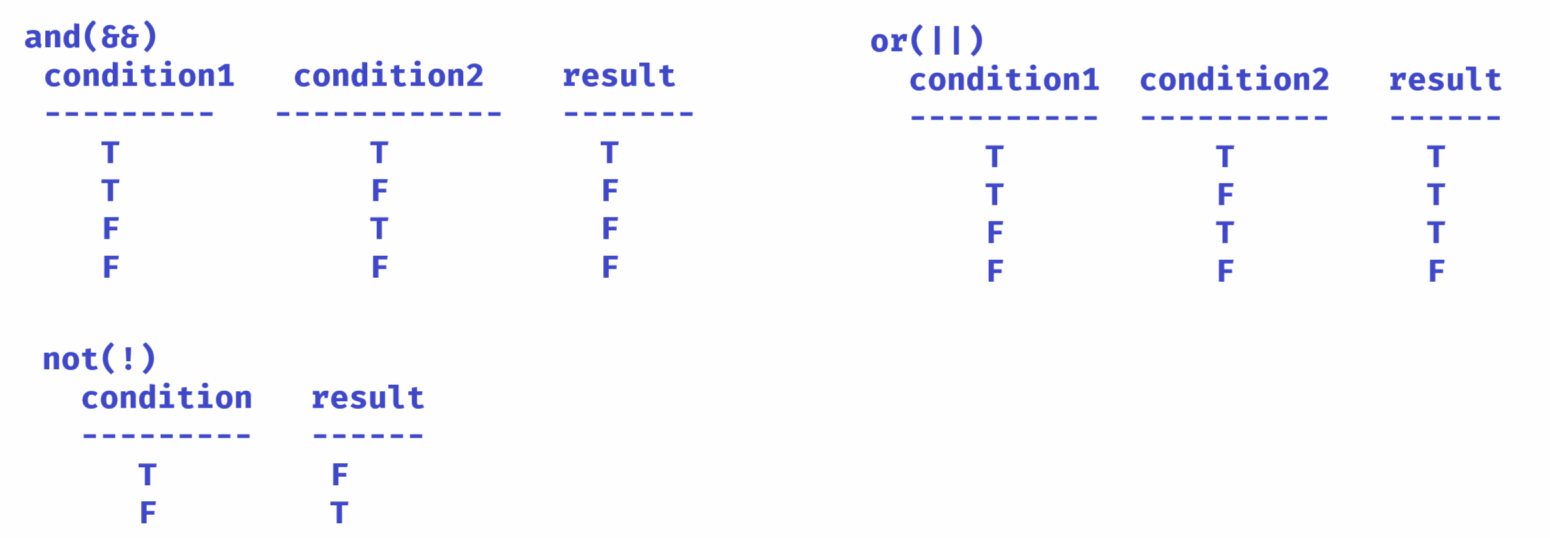
!= not equals

Logical operators:

&& Logical and

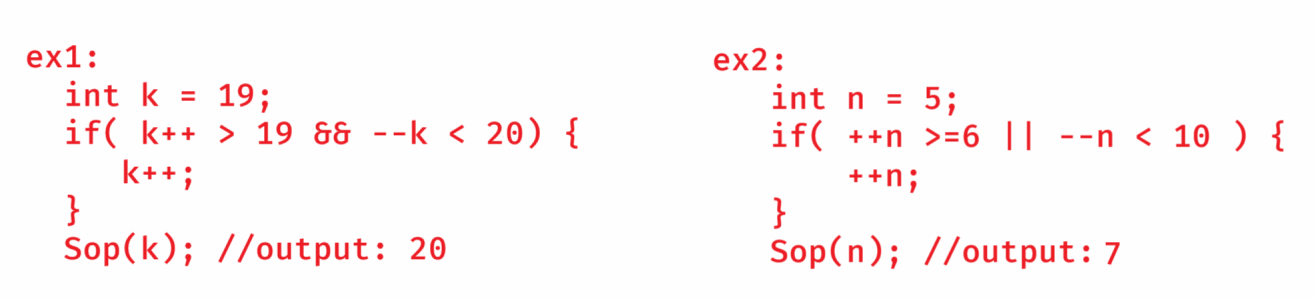
|| Logical or

! Logical not



Note: Logical and(&&) operator, checks the second condition, if the first condition is true. Otherwise, by without executing the second condition, it will result false.

Logical or(||) operator, checks the second condition, if the first condition is false. Otherwise, by without executing the second condition, it will result false.



Assignment operators:

= assign

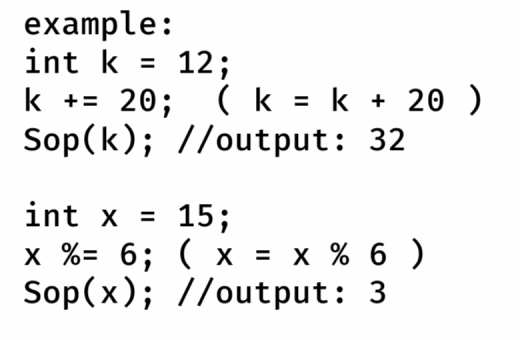
+= add then assign

-= substract then assign

\*= multiply then assign

/= divide then assign

%= modulus then assign



Bitwise operators:

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

* Bitwise operators works on bits of the operands.
* Bitwise operators are mostly used in developing cryptography applications, networking applications, device driver applications, etc..

& Bitwise AND

| Bitwise OR

^ Bitwise XOR

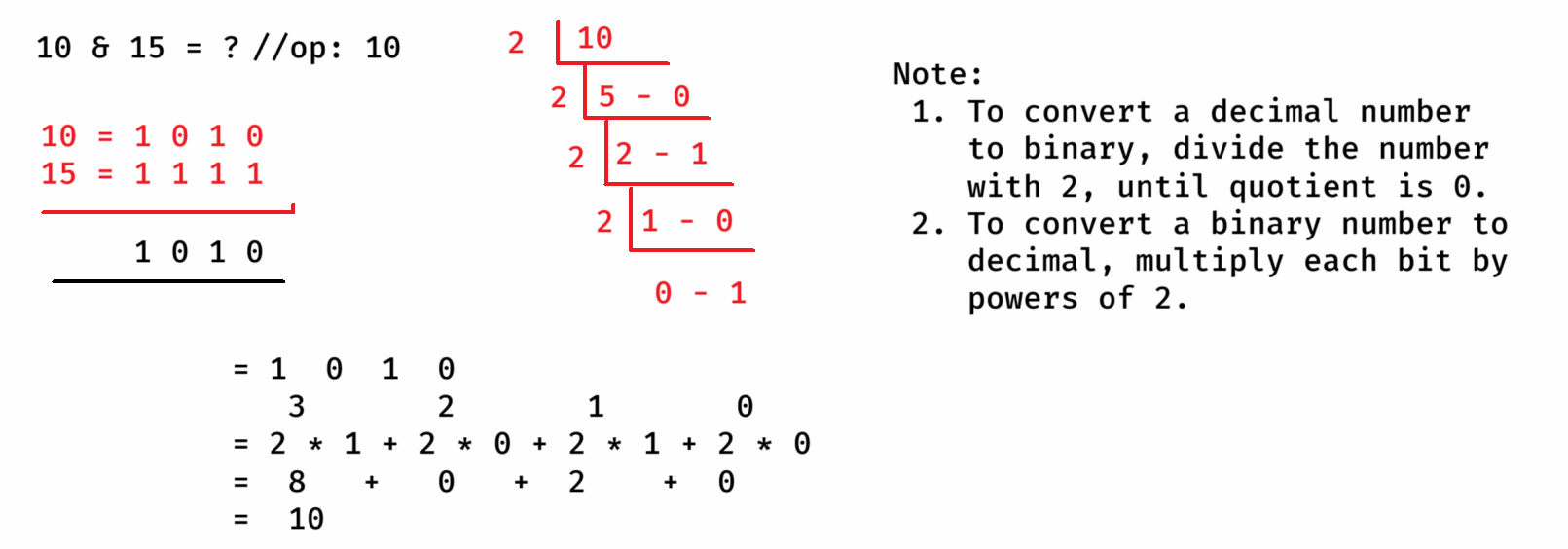
<< Left shift

>> Right shift

Bitwise AND(&&):

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

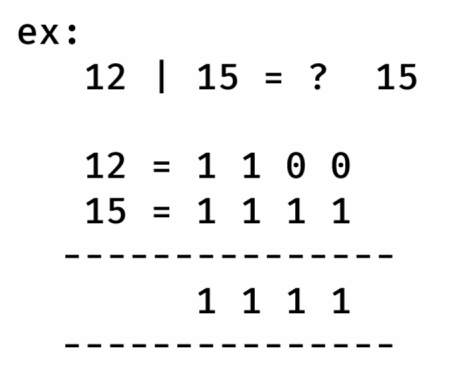
* It will set a bit 1, if the corresponding bits are 1. Otherwise, it will set a bit 0.



Bitwise OR( | ):

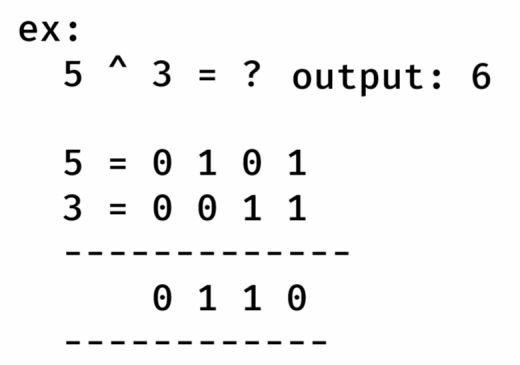
--------------

\* It will set a bit 1, if either of the corresponding bits is 1. Otherwise, it will set a bit 0.



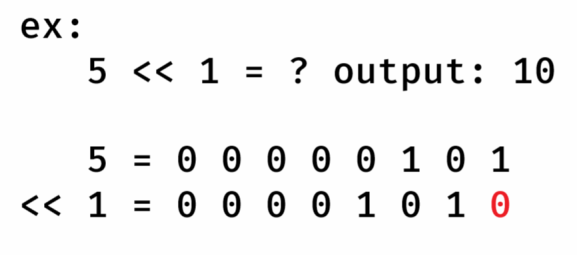
Bitwise XOR( ^ )

\* It will set a bit 1, if the corresponding bits are opposite. Otherwise, it will set a bit 0.



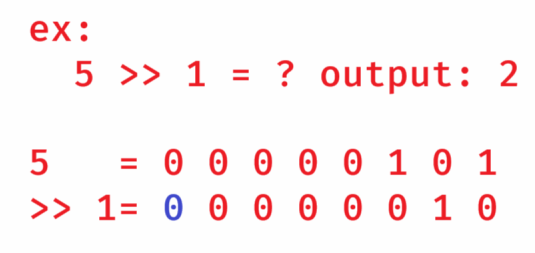
Left Shift( << ):

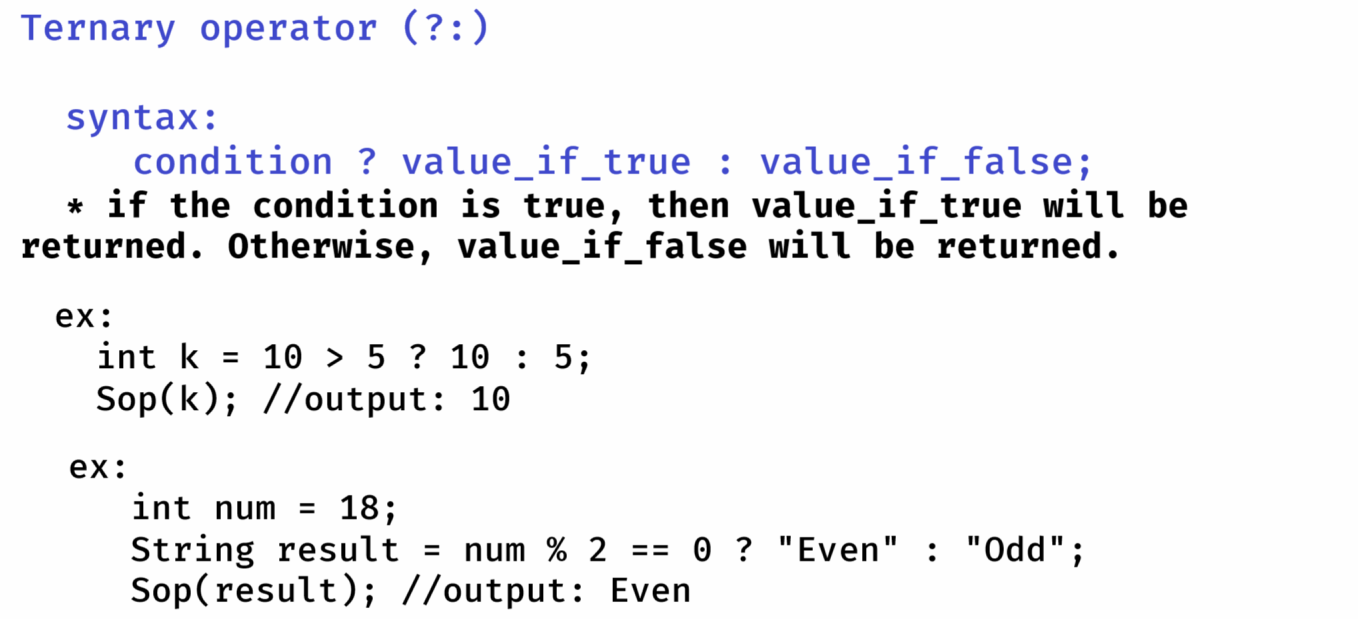
\* It moves the bits of a number to the left by the specified number of positions. Fills the empty bits with 0.

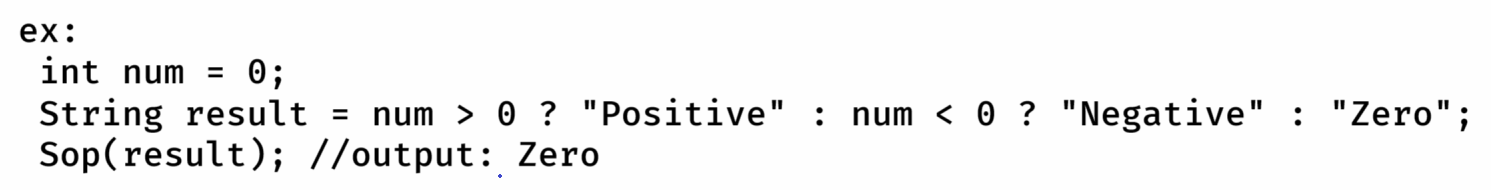


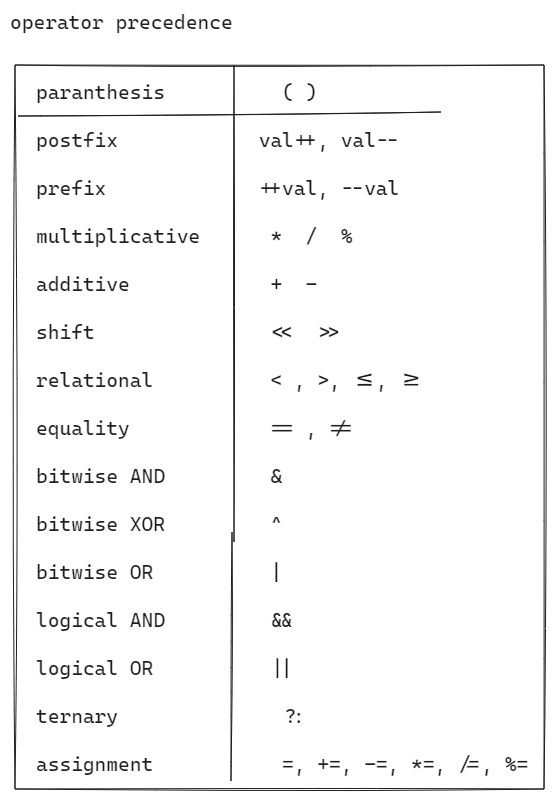
Right Shift ( >> ):

\* It moves the bits of a number to the right by the specified number of positions. Fills the empty bits created at left side with 0.









ex: int x = 3 + 4 % 9 / 2 – 1 \* 8;

Sop(x); //output: -3

ex: int x = 3 + 4 \* 9 / 6 – (2 +7);

Sop(x); //output: 0

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

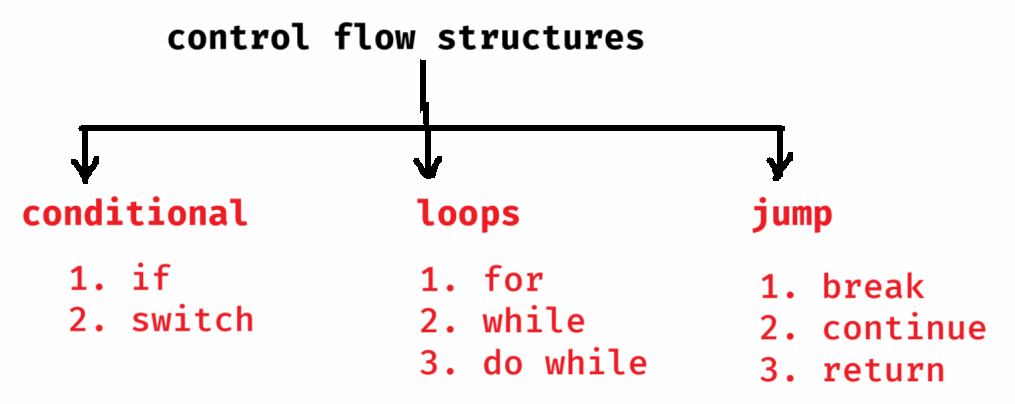
Sop(x); //output: -55

control flow structures

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

* The control flow structures defines, the order in which the statements in a program should be executed.
* There are 3 types of control flow structures.

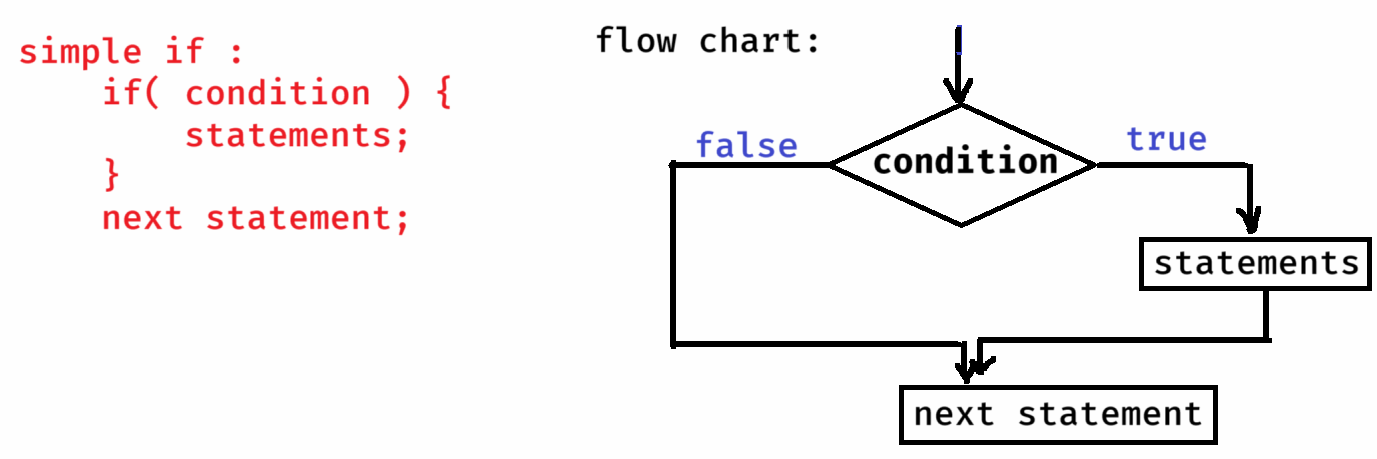
1. conditional/selection statements
2. iterational/loop statements
3. branching/jump statements.



if statements:

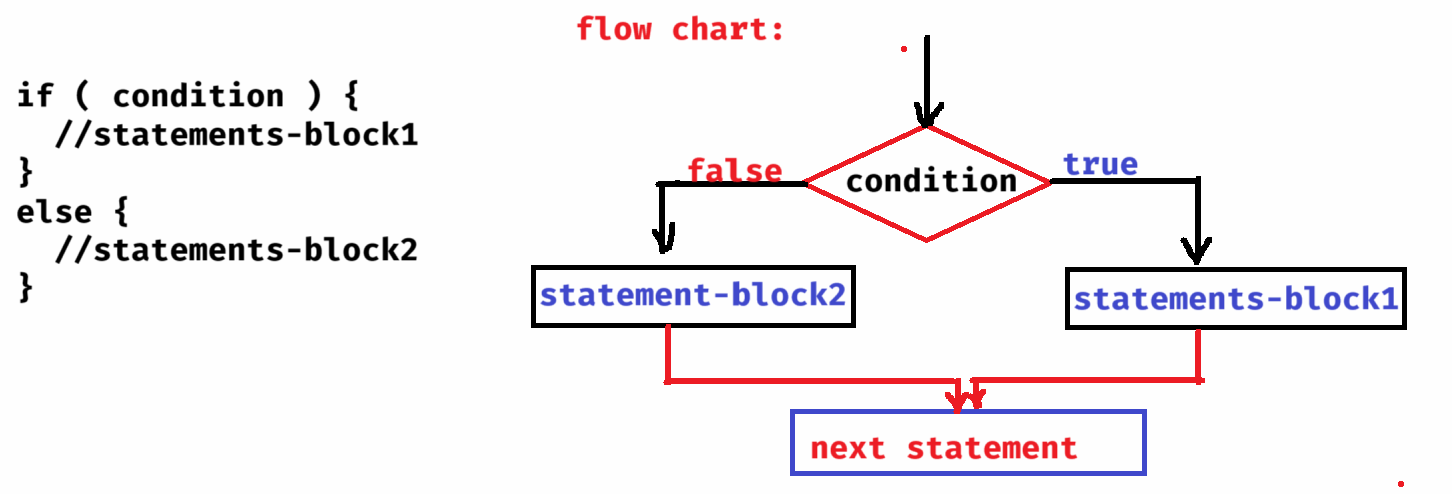
-------------

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

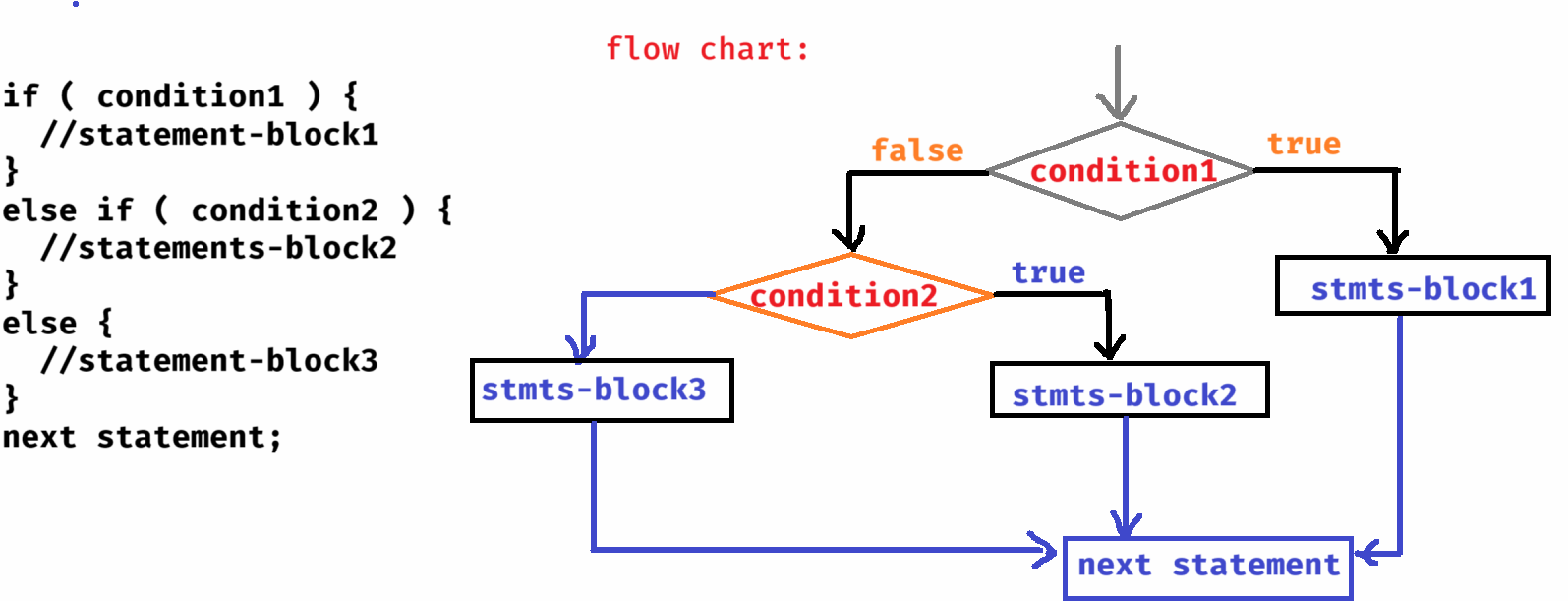


if else syntax:

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

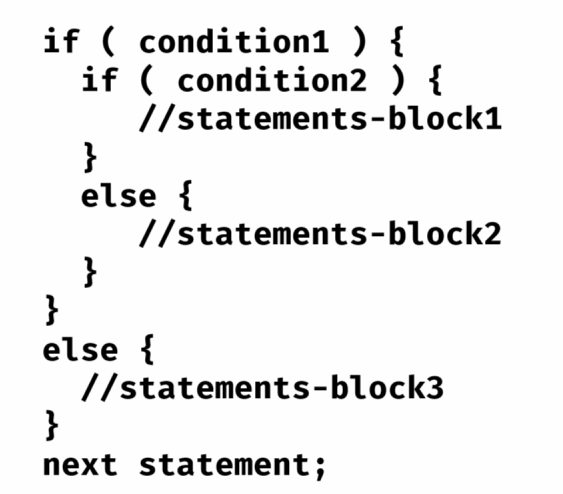


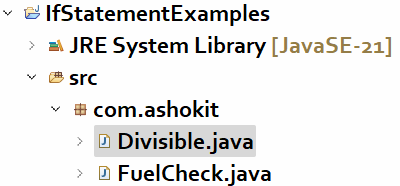
if else ladder syntax:



nested if syntax:

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





FuelCheck.java

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

/\*

\* This program reads fuel level as input and

\* displays a warning message, if it is < 5 liters.

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** FuelCheck {

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

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

System.***out***.println("Please enter fuel level in your car!");

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

**if**( fuelLevel < 5 ) {

System.***out***.println("Warning! Low fuel, please refill immediately!");

}

**else** {

System.***out***.println("Hey, Enjoy driving!!");

}

scan.close();

}

}

Divisible.java

-----------

/\*\*

\* This program reads input as an integer and checks the

\* below conditions.

\* 1. if the number is divisible by 3 then display "Zip"

\* 2. if the number is divisible by 5 then display "Zap"

\* 3. if the number is divisible by 3 and 5 then display "Jar"

\* 4. otherwise, display "Rar"

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** Divisible {

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

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

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

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

**if**( number % 3 == 0 && number % 5 == 0 )

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

**else** **if**( number % 5 == 0 )

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

**else** **if**( number % 3 == 0 )

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

**else**

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

scan.close();

}

}

BiggestOfThree.java

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

/\*

\* This program reads 3 input values and finds the biggest

\* of the three 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("largest number : " + a);

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

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

**else**

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

scan.close();

}

}

TurnOnAc.java

----------

/\*\*

\* This program reads temperature, isRaining and acWorking inputs and

\* applies then below conditions.

\* if temp > 40 && acWorking==true

display "Turn off AC, High temp"

\* else if temp > 35 && acWorking == false

display "Turn on AC"

\* else if isRaining == true && acWorking == true

display "Turn off AC"

\* else

display "No need to turn on AC"

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** TurnOnAc {

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

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

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

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

System.***out***.println("Enter is raining ( true/false) ");

**boolean** isRaining = scan.nextBoolean();

System.***out***.println("Enter is ac working ( true/false )");

**boolean** acWorking = scan.nextBoolean();

**if**( temperature > 40 && acWorking == **true** )

System.***out***.println("Turn off AC, High temperature");

**else** **if**(temperature > 35 && acWorking == **false** )

System.***out***.println("Turn on AC");

**else** **if**(isRaining == **true** && acWorking == **true** )

System.***out***.println("Turn off AC, it's already raining");

**else**

System.***out***.println("No need to Turn on AC");

scan.close();

}

}

DeliveryFeeCalculator.java

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

/\*\*

\* This program should take distance in kms as input and

\* should calculate the delivery fee based on the below conditions.

\* 1. For the first 3kms, free delivery

\* 2. For the next 3kms, delivery fee is Rs.15 per km

\* 3. For the remaining kms, delivery fee is Rs.25 per km

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** DeliveryFeeCalculator {

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

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

System.***out***.println("Please enter distance in kms");

**float** distance = scanner.nextFloat();

**float** deliveryFee = 0.0f;

**if**(distance <= 3)

deliveryFee = 0.0f;

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

deliveryFee = (distance - 3) \* 15;

**else**

deliveryFee = 3 \* 15 + (distance - 6) \* 25;

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

scanner.close();

}

}

LeapYear.java

-----------

/\*

\* This program checks whether a given year is leap year or not.

\* 1. if a year is divisible by 4, divisible by 100 and divisible by 400 then

\* it is a leap year.

\* 2. if a year is divisible by 4, divisible by 100 and not divisible by 400 then

\* it is not a leap year.

\* 3. if a year is divisible by 4, but not divisible by 100 then it is a leap year.

\* 4. if a year is not divisible by 4, then it is not a leap year.

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** LeapYear {

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

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

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

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

**if**(year % 4 == 0) {

**if**(year % 100 == 0) {

**if**(year % 400 == 0) {

System.***out***.println("Leap year");

}

**else** {

System.***out***.println("Not a Leap year");

}

}

**else** {

System.***out***.println("Leap year");

}

}

**else** {

System.***out***.println("Not a Leap year");

}

scanner.close();

/\*

\* using ternary operator

\* String str = (year % 4 == 0 && year % 100 != 0) || year % 400 == 0 ? "Leap year" : "Not a Leap year";

\* System.out.println(str);

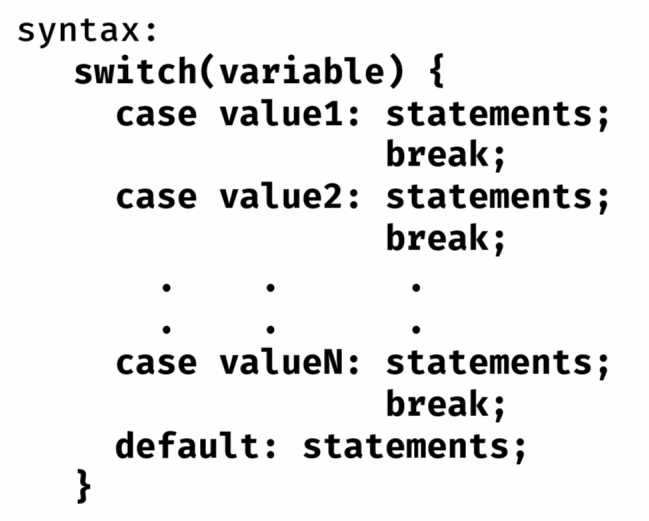
\*/

}

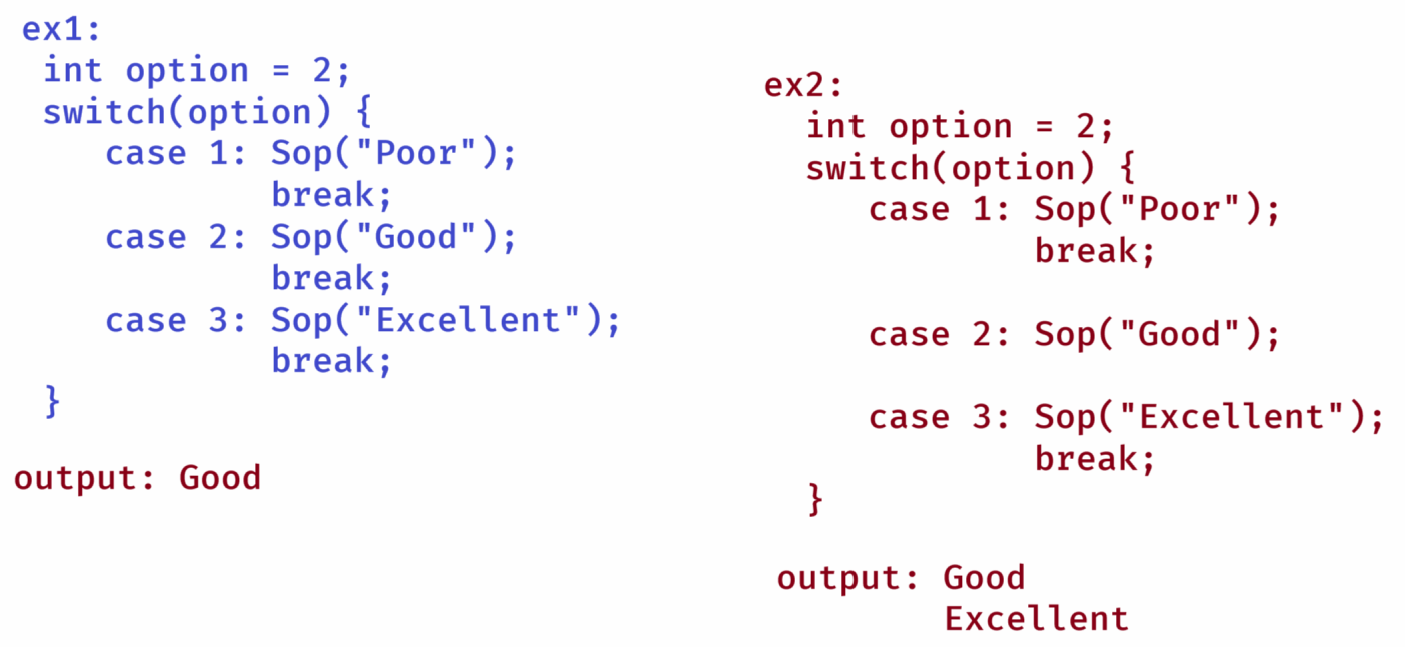
}

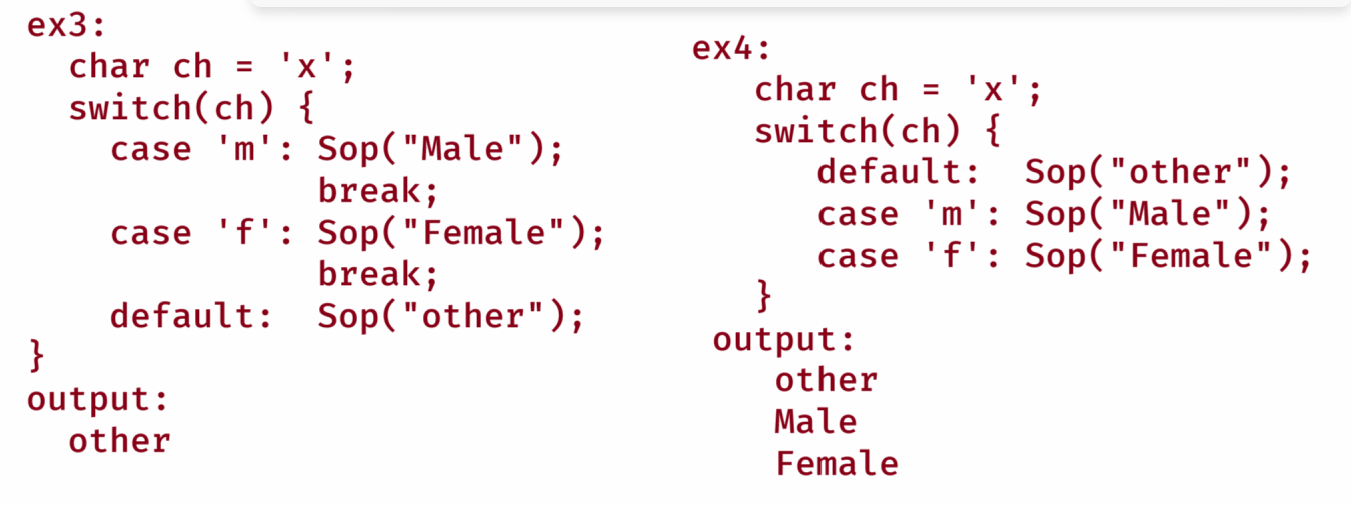
switch statement:

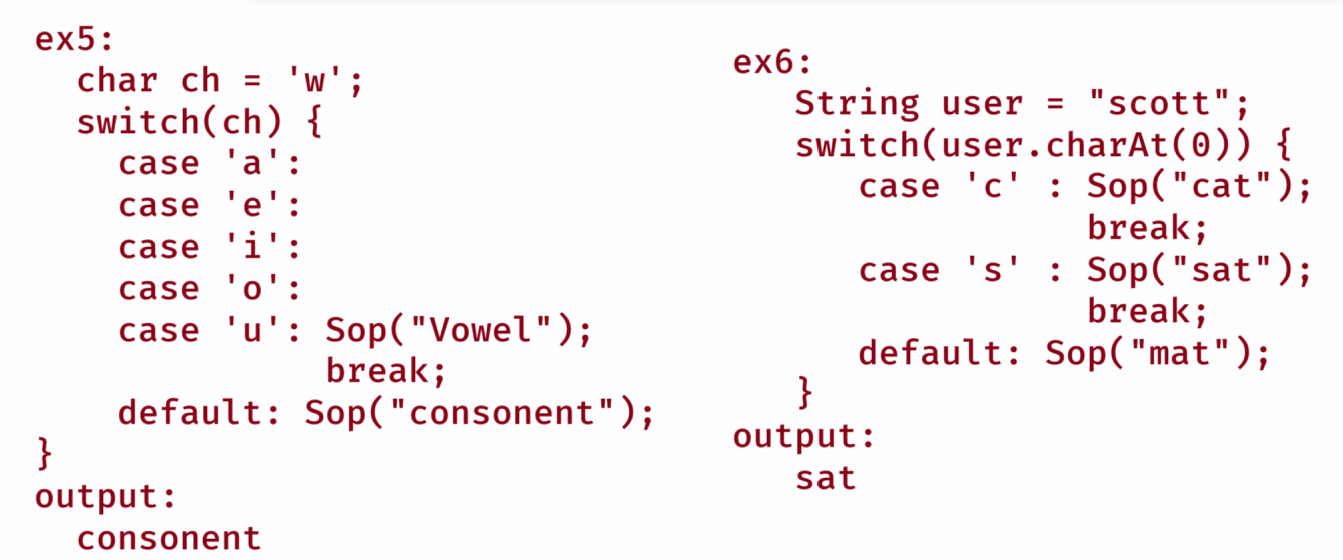
* When you want to write the multiple conditions on the single variable and if that condition is equals(==) then you can switch statement, to make the code more readable and understandable.

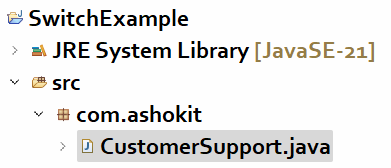


* The variable in switch statement must be either integer type(byte/short/int/long) or character type or a string type. It must not be float/double/boolean.
* A case value also must be integer type or character type or string type.
* default case is optional.









CustomerSupport.java

**package com.ashokit;**

**import java.util.Scanner;**

**public class CustomerSupport {**

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

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

**System.*out*.println("Welcome to Customer Support");**

**System.*out*.println("Press 1 for Account issues");**

**System.*out*.println("Press 2 for Billing issues");**

**System.*out*.println("Press 3 for Technical issues");**

**System.*out*.println("Press 8 to speak with Representative");**

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

**switch(choice) {**

**case 1:**

**System.*out*.println("Redirecting to Account Support....");**

**break;**

**case 2:**

**System.*out*.println("Redirecting to Billing Support....");**

**break;**

**case 3:**

**System.*out*.println("Redirecting to Technical Support....");**

**break;**

**case 8:**

**System.*out*.println("Connecting to a Representative");**

**break;**

**default:**

**System.*out*.println("Invalid choice, Please try again..");**

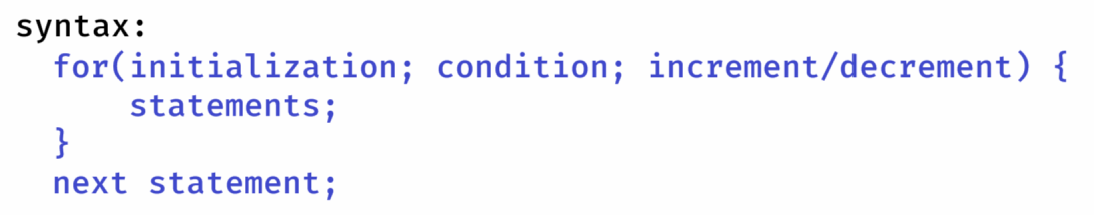
**}**

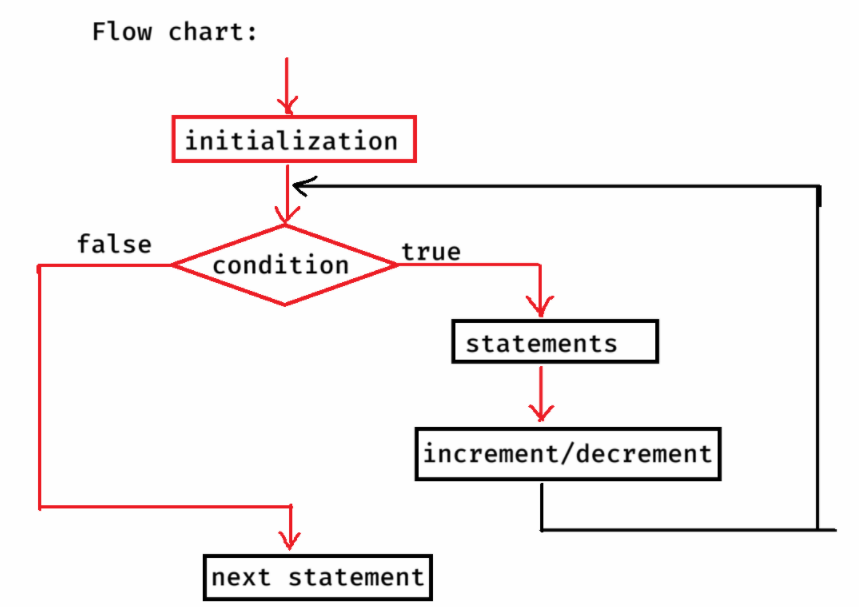
**scanner.close();**

**}**

**}**

for loop:





ex1:

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

Sop(“i = “ + i);

}

output:

i = 1

i = 2

i = 3

ex2:

int i=0;

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

Sop(“i = “ + i);

output:

i = 6

ex3:

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

Sop(“i = “+i);

output: error

ex4:

int i = 1;

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

Sop(“i = “ + i);

}

output: error

ex5:

int i = 1;

for( ; ; ) {

Sop(i);

if(i > 3)

break;

i++;

}

output:1

2

3

4

ex6:

for(int i=1; i<=5; i+=2)

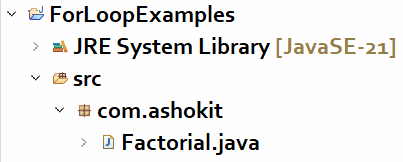
Sop(i);

output:

1

3

5



Factorial.java

----------

/\*

\* This program finds the factorial of a number

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** Factorial {

**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) {

**int** f = 1;

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

f = f \* i;

}

**return** f;

}

}

prime number:

-----------

A Natural number, which contains only 2 factors is called a prime number.

The 2 factors are 1 and itself.

To check whether a given number is prime or not, you can start the loop from 1 and goes upto n, count how many factors are exist for the number.

If that count is 2, then it is prime number. Otherwise, it is not a prime number.

But if the number is large, then the loop will iterate for more times. Hence, the program can give poor performance.

Another logic is, start the loop from 2, goes upto sqrt(n), find is there is any factor for the number. If any factor exist, then we can say that the given number is not a prime number.

If any number doesn’t have a factor from 2 upto sqrt(n), then definitely that number doesn’t contain more than 2 factors. So, we can say that it is a prime number.

Prime.java

--------

/\*\*

\* This program checks whether a given number is

\* prime or not.

\* ex:

\* n = 5,

\* output: prime number

\* n = 8,

\* output: not a prime number

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** Prime {

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

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

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

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

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

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

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

}

**else** {

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

}

scanner.close();

}

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

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

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

**return** **false**;

}

}

**return** **true**;

}

}

perfect number:

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

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

For example, if number =6,

factors excluding 6 are, 1, 2 and 3

sum = 6. So, it is a perfect number.

Perfect.java

---------

/\*

\* This program checks whether a given number is

\* perfect or not.

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** Perfect {

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

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

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

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

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

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

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

} **else** {

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

}

scanner.close();

}

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

**int** sum = 0;

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

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

sum = sum + i;

}

**if** (sum == n)

**return** **true**;

**else**

**return** **false**;

}

}

fibonacci series:

. fibonacci series is a mathematical series.

. The first two terms of the series are 0 and 1

. The next term is a sum of previous two terms.

. For example, fibonacci series of 5 terms are,

0 1 1 2 3

. For example, fibonacci series of 8 terms are,

1. 1 1 2 3 5 8 13

Fibonacci.java

/\*\*

\* This program will display fibonacci series of n terms

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** Fibonacci {

**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();

*printFibonacciSeries*(n);

scanner.close();

}

**private** **static** **void** printFibonacciSeries(**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;

}

}

}

Nested for loops:

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

ex1:

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

{

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

{

System.out.print(j + “ “);

}

System.out.println();

}

output:

1 2

1 2

ex2:

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

{

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

{

System.out.print(j + “ “);

}

System.out.println();

}

output:

1

1 2

1 2 3

ex3:

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

{

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

{

System.out.print(j + “ “);

}

System.out.println();

}

output:

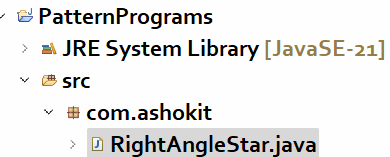
1

1 2 3

1 2 3 4 5

Pattern programs:

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



RightAngleStar.java

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

**/\***

**\* Write a program to print the right angle star pattern.**

**\* example:**

**\* if n = 5,**

**\* output:**

**\* \***

**\* \* \***

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

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

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

**\*/**

**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 display right angle star pattern");**

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

***printPattern*(n);**

**scanner.close();**

**}**

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

**//outer loop : rows**

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

**//inner loop: columns(stars)**

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

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

**}**

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

**}**

**}**

**}**

**InvertedRightAngleStar.java**

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

**/\***

**\* Write a program to print the right angle star pattern.**

**\* example:**

**\* if 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 display inverted right angle star pattern");**

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

***printPattern*(n);**

**scanner.close();**

**}**

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

**// outer loop : rows**

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

**// inner loop: columns(stars)**

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

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

**}**

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

**}**

**}**

**}**

LeftAngleStar.java

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

**/\***

**\* Write a program to print the right angle star pattern.**

**\* example:**

**\* if 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 display left angle star pattern");**

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

***printPattern*(n);**

**scanner.close();**

**}**

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

**// outer loop : rows**

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

**// inner loop: spaces**

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

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

**}**

**// inner loop: columns(stars)**

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

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

**}**

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

**}**

**}**

**}**

PyramidStar.java

**/\***

**\* Write a program to print the right angle star pattern.**

**\* example:**

**\* if n = 5,**

**\* output:**

**\* \***

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

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

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

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

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class PyramidStar {**

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

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

**System.*out*.println("Enter n value to display pyramid star pattern");**

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

***printPattern*(n);**

**scanner.close();**

**}**

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

**//outer loop : rows**

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

**//inner loop1 : spaces**

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

**System.*out*.print(" "); //print one space**

**}**

**//inner loop2: stars**

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

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

**}**

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

**}**

**}**

**}**

DiamondStar.java

**/\***

**\* Write a program to print the diamond star pattern.**

**\* example:**

**\* if n = 5,**

**\* output:**

**\* \***

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

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

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

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

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

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

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

**\* \***

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class DymondStar {**

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

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

**System.*out*.println("Enter n value to display Diamond star pattern");**

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

***printPattern*(n);**

**scanner.close();**

**}**

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

**//outer loop : rows**

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

**//inner loop1 : spaces**

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

**System.*out*.print(" "); //print one space**

**}**

**//inner loop2: stars**

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

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

**}**

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

**}**

**//outer loop : rows-1**

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

**//inner loop1: spaces**

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

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

**}**

**//inner loop2: stars**

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

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

**}**

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

**}**

**}**

**}**

**NumberPattern.java**

**/\*\***

**\* Writer a program to display a number pattern like below.**

**\* if n = 5,**

**\* output:**

**\* 1**

**\* 2 3**

**\* 4 5 6**

**\* 7 8 9 10**

**\* 11 12 13 14 15**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class NumberPattern {**

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

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

**System.*out*.println("Enter n value to display the number pattern");**

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

***printPattern*(n);**

**scanner.close();**

**}**

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

**int x = 1;**

**// outer loop : rows**

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

**// inner loop: numbers**

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

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

**x++;**

**}**

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

**}**

**}**

**}**

**PascalTriangle.java**

**/\***

**\* This program is to display pascal triangle for 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 display pascal triangle");**

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

***printPattern*(n);**

**scanner.close();**

**}**

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

**//outer loop : rows**

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

**//inner loop : spaces**

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

**System.*out*.print(" "); //single space**

**}**

**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;**

**}**

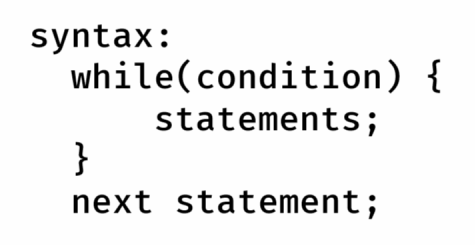
**return fact;**

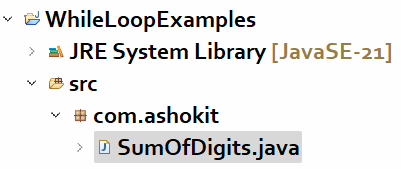
**}**

**}**

while loop

* When you know how many times to iterate the loop, then use for loop.
* When you don’t know how many times, but you want to iterate as long as the given condition is true. Then use while loop.
* In while loop, when the given condition becomes false, the loop will terminate.





SumOfDigits.java

**/\*\***

**\* This program finds the sum of the digits of**

**\* a given number**

**\* ex:**

**\* n = 176**

**\* 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();**

**int k = *findSumOfDigits*(n);**

**System.*out*.println("Sum of the digits = " + k);**

**scanner.close();**

**}**

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

**int sum = 0;**

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

**int lastDigit = n % 10;**

**sum = sum + lastDigit;**

**n = n / 10;**

**}**

**return sum;**

**}**

**}**

Armstrong.java

/\*\*

\* This program is used to check whether a given number

\* is an Armstrong number or not.

\* th

\* Armstrong number : If sum of the n power of each digit

\* of a number is equal to the same

\* number then it is called an Armstrong

\* number.

\* example:

\* number = 153

\* The length of this number is : 3

\* 3 3 3

\* 1 + 5 + 3 = 153 (It is Armstrong number)

\* example:

\* number = 124

\* The length of this number is : 3

\* 3 3 3

\* 1 + 2 + 4 = 73

\* 124 is not an Armstrong number

\*/

**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 check is it an Armstrong number or not");

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

**boolean** flag = *isArmstrong*(number);

**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** isArmstrong(**int** n) {

**int** count = 0;

**int** temp1, temp2;

temp1 = temp2 = n;

**int** sum = 0;

//This while loop counts the no of digits

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

temp1 = temp1 / 10;

count++;

}

//This while loop finds the sum of the powers of the digits

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

**int** lastDigit = temp2 % 10;

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

temp2 = temp2 / 10;

}

**if** (sum == n)

**return** **true**;

**else**

**return** **false**;

}

}

Strong.java

/\*\*

\* This program checks whether a given number is

\* strong number or not.

\* strong number : if the sum of the factorials of each digit

\* is equals to the number itself.

\* ex:

\* number = 145

\* 1! + 4! + 5! = 145 (strong number)

\*

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** Strong {

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

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

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

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

**boolean** flag = *isStrong*(number);

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

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

**else**

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

scanner.close();

}

**private** **static** **boolean** isStrong(**int** number) {

**int** sum = 0;

**int** temp = number;

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

**int** lastDigit = temp % 10;

sum = sum + *factorial*(lastDigit);

temp = temp / 10;

}

**if**(sum == number)

**return** **true**;

**else**

**return** **false**;

}

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

**int** fact = 1;

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

fact = fact \* i;

}

**return** fact;

}

}

Palindrome.java

**/\*\***

**\* Write a program to check whether a given number is palindrome or not**

**\* palindrome : If reverse of a number is equal to the same number**

**\* then it is called palindrome.**

**\* ex:**

**\* number = 12321**

**\* reverse = 12321**

**\* So, it is a palindrome**

**\* ex:**

**\* number = 1021**

**\* reverse = 1201**

**\* So, it is not a palindrome**

**\*/**

**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 number or not");**

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

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

**if (flag == true)**

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

**else**

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

**scanner.close();**

**}**

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

**int temp = number;**

**int reverse = 0;**

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

**int lastDigit = temp % 10;**

**reverse = reverse \* 10 + lastDigit;**

**temp = temp / 10;**

**}**

**if (reverse == number)**

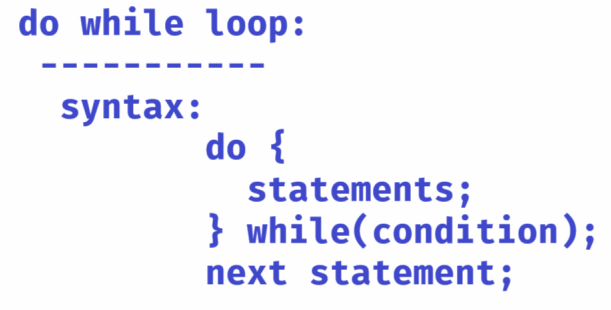
**return true;**

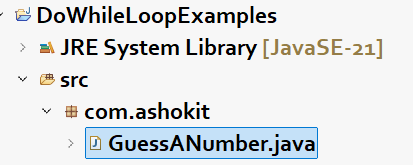
**else**

**return false;**

**}**

**}**





GuessANumber.java

**/\*\***

**\* This program will accept a number from 0 to 9**

**\* from the user.**

**\* It generates a random number between 0 to 9**

**\* and then checks whether both are matched or not**

**\* If matched then displays "Guess is correct",**

**\* otherwise, displays "Guess is wrong".**

**\*/**

**package com.ashokit;**

**import java.util.Random;**

**import java.util.Scanner;**

**public class GuessANumber {**

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

**guessANumber();**

**}**

**private static void guessANumber() {**

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

**int yourNumber = -1;**

**do {**

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

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

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

**scanner.close();**

**Random random = new Random();**

**//generates a random number between 0 to 9**

**int randomNumber = random.nextInt(10);**

**if(yourNumber == randomNumber) {**

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

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

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

**}**

**else {**

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

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

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

**}**

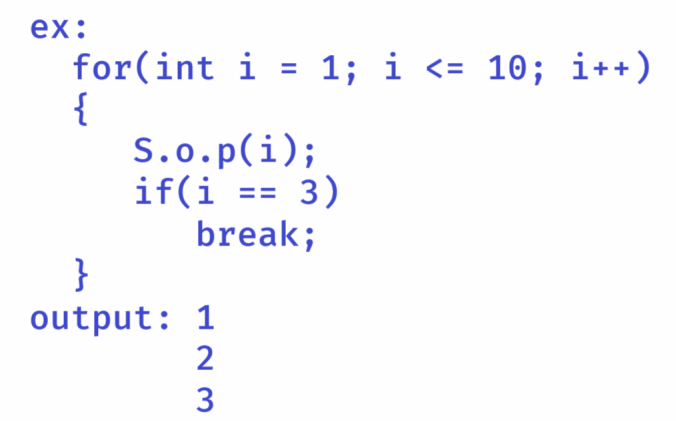
**}**

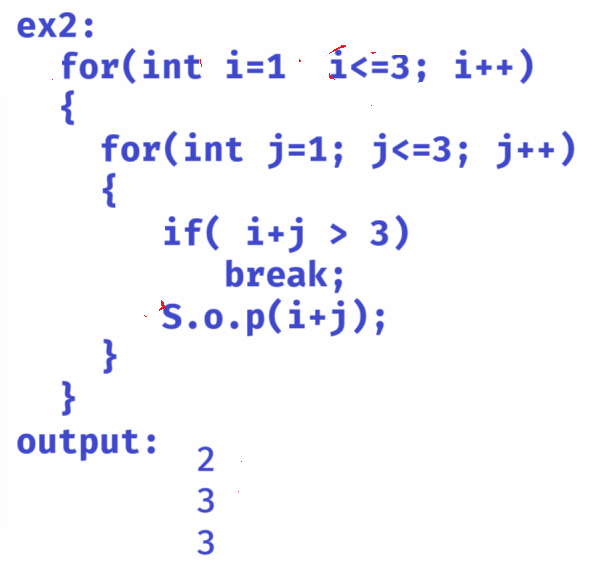
**}**

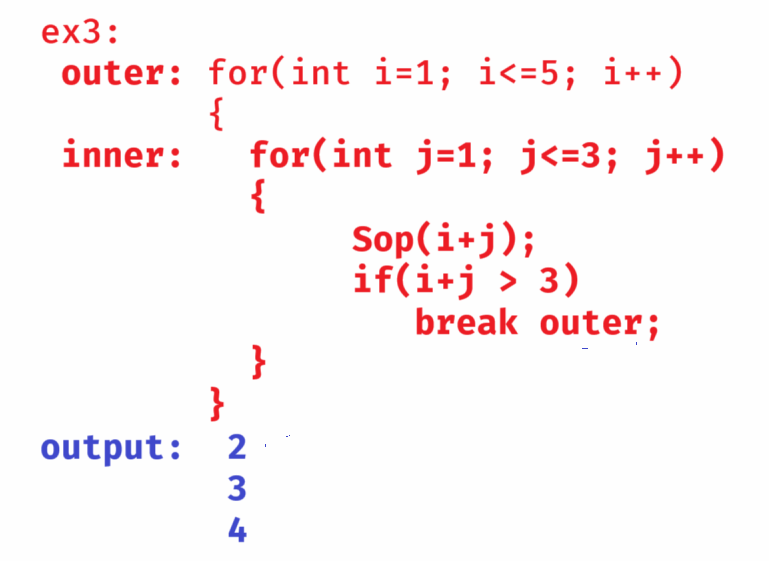
======================

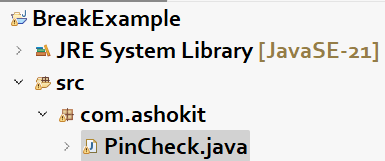
break statement:

* break statement can be used either in switch statement or in a loop(for loop/while loop/do while loop).
* In switch case, we use break in a case, to push the control out of the switch statement, after executing a case.
* In a loop, we use break statement, to push the control out of the loop, based on a condition









PinCheck.java

/\*\*

\* This program checks the user entered pin for a maximum

\* of 3 times.

\* Within 3 attempts, if pin is correct, then break will

\* break the loop.

\* If not, the loop will display a message that pin is not correct.

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** PinCheck {

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

*pinVerification*();

}

**private** **static** **void** pinVerification() {

**int** correctPin = 1234;

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

**int** attempts;

**for**(attempts = 1; attempts <= 3; attempts++) {

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

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

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

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

**break**;

}

**else** {

System.***out***.println("Pin is incorrect!");

System.***out***.println("Remaining attempts : " + ( 3 - attempts));

}

}

**if**(attempts == 4) {

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

}

scanner.close();

}

}

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

continue:

* **continue statement is used in a loop, to skip the remaining code for the current iteration and to proceed the control for the next iteration.**

**ex1:**

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

**{**

**if(i==3)**

**continue;**

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

**}**

**output: 1**

**2**

**4**

**5**

**ex2:**

**outer:**

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

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

**if(j==3)**

**continue outer;**

**S.o.p(“i=”+i+”, j=”+j);**

**}**

**}**

**output:**

i=1, j=1

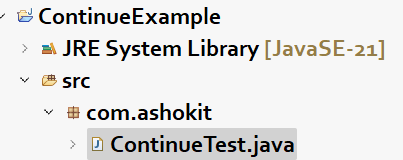
i=1, j=2

i=2, j=1

i=2, j=2

i=3, j=1

i=3, j=2



**ContinueTest.java**

**/\*\***

**\* This program should add five positive integers**

**\* to the sum variable.**

**\* If the input value is negitive then continue the loop**

**\* for next iteration.**

**\* If 5 numbers are added to the sum, then break the loop.**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class ContinueTest {**

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

**//connect Scanner object to the keyboard**

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

**int sum = 0;**

**int count = 0;**

**while(true) {**

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

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

**if( x < 0)**

**continue;**

**sum = sum + x;**

**count++;**

**if(count == 5)**

**break;**

**}**

**System.*out*.println("sum of 5 positive integers = " + sum);**

**scanner.close();**

**}**

**}**

return statement:

* return statement is used to move the control from the current method to its caller method.
* if you write the return statement in Java main method then the control goes back to the JVM.

ex1:

class A {

p s v m(String[] args)

{

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

{

if(i==3)

return;

S.o.p(i);

}

}

}

output:

1

2

ex2:

class A {

p s v m(String[] args) {

int k = square(10);

S.o.p(k);

}

private static int square(int i) {

return i \* i;

}

}

output: 100

ex3:

class A {

p s v m(String[] args) {

boolean flag = isEven(15);

S.o.p(flag);

}

private static boolean isEven(int i) {

if(i % 2 == 0)

return true;

return false;

}

}

output: false

Arrays

* **A variable can store a single value.**
* **if you want to store multiple values, then you need multiple variables.**
* **if multiple variables created in a program, then the program code becomes complex and also the performance of the program will be decreased.**
* **Because, JVM will allocate the memory for the variables at different memory locations, so while fetching the values from the variables, the JVM has to search for the variables in the entire memory.**
* **So, a solution to the above problems is use arrays.**
* **An array is a single variable, but it can store multiple values and also the memory for the values will be allocated at the contigeous memory locations.**
* **A real-time example is, suppose you want to store last 7 days prices of a stock in a Stock Market Application. Here, you have to use an array to store the prices.**
* **Another real-time example is, supose you want to play 7 songs in a music player application. Here, you have to add the 7 songs to the playlist. Here, playlist is an array which stores the songs.**

**array declaration, creation and initialization:**

**datatype[] arrayvariable;**

**(or)**

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

**for example:**

**int[] arr;**

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

**for example:**

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

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

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

**For example:**

**arr[0] = 17;**

**arr[1] = 6;**

**arr[2] = 9; initialization**



**arr[3] = 1;**

**arr[4] = 12;**

**we can also combine declaration, creation and initialization together.**

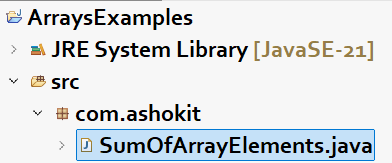
**int[] arr = new int[] { 9, 45, 2, 0, 32 };**

**(or)**

**int[] arr = { 9, 45, 2, 0, 32 };**

**limitations of an array:**

1. **array has a fixed size. You can’t increase/decrease the size after array creation.**
2. **array can store only homogeneous elements.**
3. **array doesn’t have pre-defined methods to perform insert/update/remove/serach/sort operations.**

****

**SumOfArrayElements.java**

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

**/\*\***

**\* This program is to find the sum of array elements.**

**\* This program first reads the size of an array from**

**\* the user and then reads the elements from the user.**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class SumOfArrayElements {**

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

**//connect Scanner to the keyboard**

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

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

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

**//create array**

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

**//read the array elements**

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

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

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

**}**

**int sum = *findSum*(arr);**

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

**scanner.close();**

**}**

**private static int findSum(int[] arr) {**

**int sum = 0;**

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

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

**}**

**return sum;**

**}**

**}**

**LinearSearch.java**

/\*\*

\* This program is to search for an element

\* in the array with linear search.

\* Linear search means, compare the searching element

\* with each element in the array. If matched then

\* display that element is found.

\* If array doesn't have this element then

\* display that element is not found.

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** LinearSearch {

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

// connect Scanner to the keyboard

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

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

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

// create array

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

// read the array elements

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

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

arr[i] = scanner.nextInt();

}

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

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

*linearSearch*(arr, key);

scanner.close();

}

**private** **static** **void** linearSearch(**int**[] arr, **int** key) {

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

**int** index = -1;

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

**if** (arr[i] == key) {

flag = **true**;

index = i;

**break**;

}

}

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

System.***out***.println("\u001B[32m" + "Element found at index : " + index);

}

**else** {

System.***out***.println("\u001B[31m" + "Element not found");

}

}

}

**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, and break 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.

**BinarySearch.java**

**package com.ashokit;**

**import java.util.Arrays;**

**import java.util.Scanner;**

**public class BinarySearch {**

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

**// connect Scanner to the keyboard**

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

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

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

**// create array**

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

**// read the array elements**

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

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

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

**}**

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

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

***binarySearch*(arr, key);**

**scanner.close();**

**}**

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

**// sort the array in ascending order**

**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] == key ) {**

**flag = true;**

**break;**

**}**

**else if (arr[mid] < key) {**

**low = mid + 1;**

**}**

**else {**

**high = mid - 1;**

**}**

**}**

**if(flag == true)**

**System.*out*.println("\u001B[31m" + "element found!");**

**else**

**System.*out*.println("\u001B[35m" + "element not found!!");**

**}**

**}**

**RemoveDuplicates.java**

**package com.ashokit;**

**public class RemoveDuplicates {**

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

**int[] arr = { 2, 5, 0, 1, 2, 3, 1, 5 };**

***removeDuplicates*(arr);**

**}**

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

**int[] arr2 = new int[arr.length];**

**int index = 0;**

**arr2[0] = arr[0];**

**index++;**

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

**boolean flag = false;**

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

**if (arr2[j] == arr[i]) {**

**flag = true;**

**break;**

**}**

**}**

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

**arr2[index] = arr[i];**

**index++;**

**}**

**}**

**System.*out*.println("Elements in original array : ");**

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

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

**}**

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

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

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

**System.*out*.print(arr2[j] + " ");**

**}**

**}**

**}**

**for each loop:**

* **In Java, we have for loop and for each loop.**
* **For both the loops, the keyword is “for” only.**
* **Normal for loop is used for repeating a loop of a number of times like 5 times or 10 times like that.**
* **for each loop is used to repeat over the elements of an array or over the elements of a collection object.**

**syntax:**

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

**{**

**//logic**

**}**

**ex:**

**int[] arr = { 3, 6, 5, 1};**

**int sum = 0;**

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

**sum = sum + x;**

**}**

**Sop(sum);**

Bubble sort:

* Bubble sort is a basic sorting algorithm for sorting the elements.
* It compares the adjacent pair of elements. If the first element is greater than second element then swap them.
* In each pass, one largest element will be bubbled to the end of the array.
* if an array has n elements, then the array elements are sorted in n-1 passes.

ex:

[ 3, 8, 5, 1, 0]

pass1:

3 > 8 (false)

8 > 5 (true) – swapped

[3, 5, 8, 1, 0]

8 > 1 (true) – swapped

[3, 5, 1, 8, 0]

8 > 0 (true) – swapped

[3, 5, 1, 0, 8] – 8 is bubbled.

pass2:

3 > 5 (false)

5 > 1 (true) – swapped

[3, 1, 5, 0, 8]

5 > 0 (true) – swapped

[3, 1, 0, 5, 8] – 5 is bubbled.

pass3:

3 > 1 (true) – swapped

[1, 3, 0, 5, 8]

3 > 0 (true) – swapped

[1, 0, 3, 5, 8] – 3 is bubbled

pass4:

1 > 0 (true) -- swapped

[0, 1, 3, 5, 8] – 1 is bubbled

* Now the array elements are sorted.

BubbleSort.java

**package** com.ashokit;

**public** **class** BubbleSort {

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

**int**[] arr = { 3, 8, 1, 5, 0 };

*bubbleSort*(arr, arr.length);

}

**private** **static** **void** bubbleSort(**int**[] arr, **int** length) {

System.***out***.println("The elements of array before sorting");

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

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

}

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

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

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

{

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

{

**if** (arr[j] > arr[j+1])

{

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

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

System.***out***.println("The elements of array after sorting");

**for**(**int** x : arr)

{

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

}

}

}

Two-Dimention arrays:

* A one-dimention array contains a single row with multiple columns.
* A two-dimention array contains multiple rows and multiple columns.
* So, we can think a two-dimention array like a matrix.

syntax:

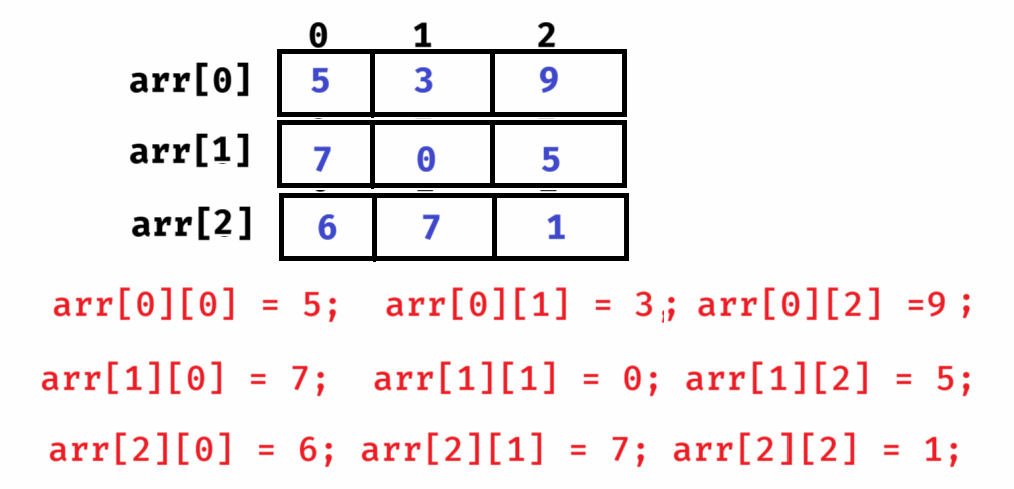
int[][] arr = new int[3][3];

(or)

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

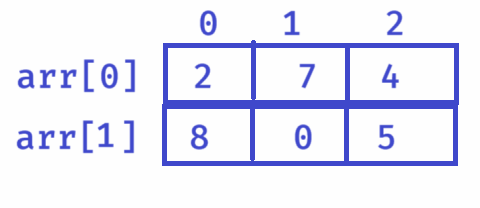
(or)

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



ex: static initialization

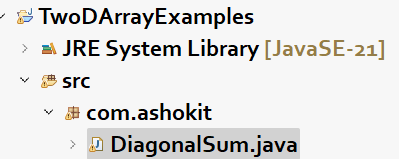
int[][] arr = { {2, 7, 4}, {8, 0, 5} };



ex:

int[][] array = { {5, 0, 2}, {3, 6}, {3, 9, 1, 2} }; //jogged array

//jogged array means, each row can have different number of columns.



DiagonalSum.java

**/\*\***

**\* This program reads the input elements into 2D array**

**\* from the user, then finds the sum of left and right**

**\* diagonal elements separately.**

**\*/**

**package com.ashokit;**

**import java.util.Scanner;**

**public class DiagonalSum {**

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

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

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

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

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

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

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

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

**System.*out*.println("Invalid input");**

**return;**

**}**

**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();**

**}**

**}**

***calculateDiagonalSum*(arr, rows, cols);**

**scanner.close();**

**}**

**private static void calculateDiagonalSum(int[][] arr, int rows, int cols) {**

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

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

**{**

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

**{**

**if (i == j)**

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

**if (i + j == rows - 1)**

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

**}**

**}**

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

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

**}**

**}**

MatrixAddition.java

package com.ashokit;

public class MatrixAddition {

public static void main(String[] args) {

int a[][] = { {2, 4}, {6, 7}, {8, 9} }; // size = 3 \* 2

int b[][] = { {0, 3}, {1, 6}, {5, 9} }; // size = 3 \* 2

*addition*(a, b);

}

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

int r1 = a.length;

int c1 = a[0].length;

int c[][] = new int[r1][c1];

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

{

for(int j = 0; j < c1; j++)

{

c[i][j] = a[i][j] + b[i][j];

}

}

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

{

for (int j = 0; j < c1; j++)

{

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

}

System.*out*.println();

}

}

}

work:

Prepare Matrix multiplication program.

String handling

String is a datatype or class?

Ans: String is a class in java.lang package.

java.lang is the default package, which is imported automatically into every java program. You no need to import this package explicitly.

* There are 2 ways that a string can be created.
* 1. String literal
* 2. using new keyword

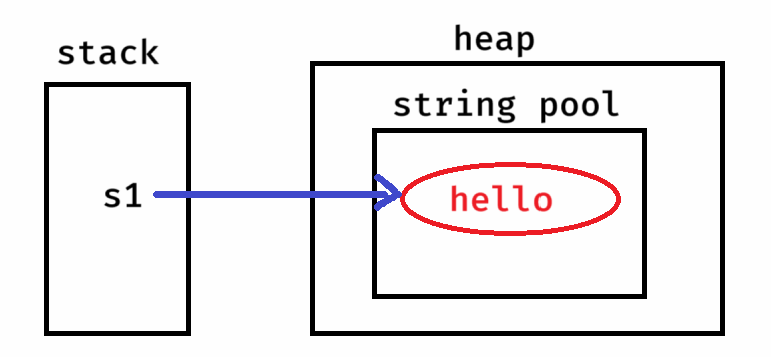
ex:

String s1 = “hello”; //String literal

String s2 = new String(“hello”); //using new keyword

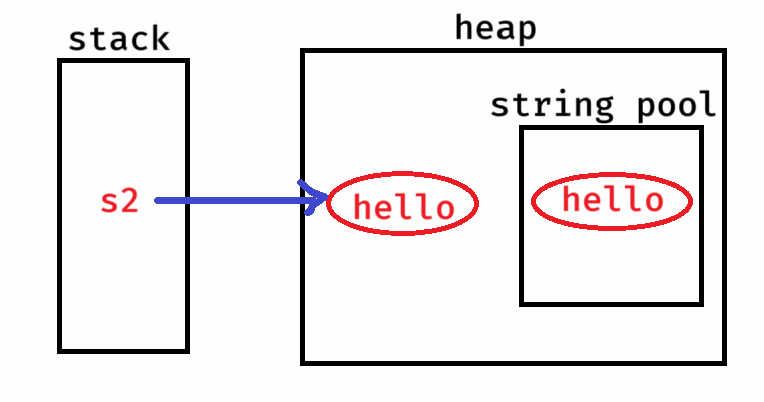
String s1 = “hello”;

1. JVM will check, is already a string object exist with the same content in the string pool.
2. If yes, then JVM will point this variable to that existing object only.
3. If no, then JVM will create a new string object in the string pool, then points this variable to that object.



String s2 = new String(“hello”);

* JVM will check, is already a string object exist with the same content in the string pool.
* If yes, then JVM creates a new string object in the heap, then points this variable to the object in the heap.
* If no, then JVM creates a new string object in the string pool, then creates a new string object in the heap, then points this variable to the object in the heap.



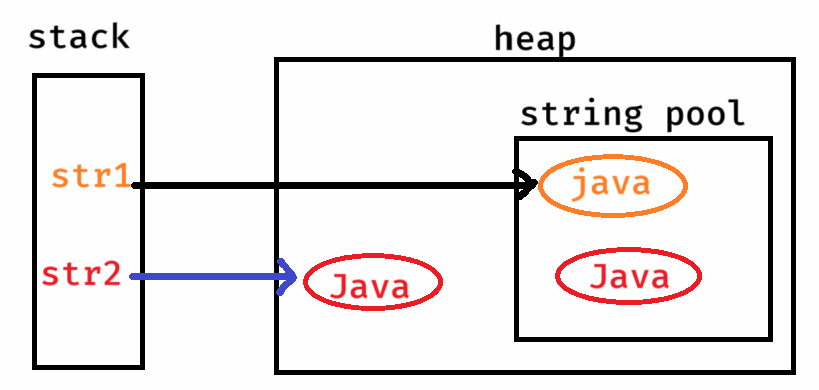
ex:

How many objects are created for the below code?

String str1 = “java”;

String str2 = new String(“Java”);

Ans: 3 objects



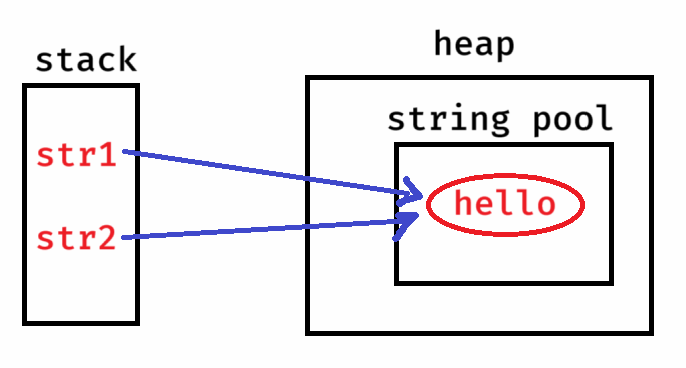
ex:

How many objects are created for the below code?

String str1 = “hello”;

String str2 = “hello”;

Ans: 1 object.



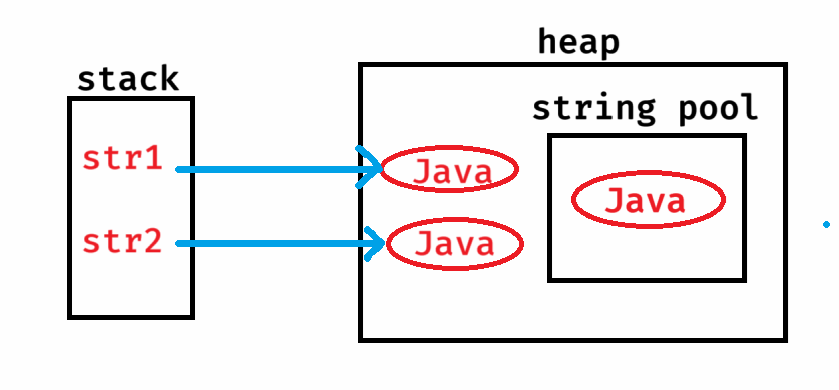
ex:

How many objects are created for the below code?

String str1 = new String(“Java”);

String str2 = new String(“Java”);

Ans: 3 objects



ex:

How many objects are created for the below code?

String s1 = new String(“Java”);

String s2 = new String(“Shekher”);

Ans: 4 objects.

ex:

How many objects are created for the below code?

String s1 = new String(“Ashokit”);

String s2 = s1;

Ans: 2 objects.

string length:

length – it is attribute

length() – it is method

ex1:

String[] strArray = { “Taman”, “Jessica”, “John” };

Sop(strArray.length); //output: 3

Sop(strArray.length()); //error

ex2:

String[] strArray = { “Sophia”, “Clara”, “Anushka” };

Sop(strArray[2].length); // error

Sop(strArray[2].length()); //output: 7

strings comparision:

* strings comparision can be done in 4 ways.

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

* == operator, compares whether the two variables are pointing to the same object or not. If yes, returns true. Otherwise, returns false.

ex1:

String s1 = “hello”;

String s2 = “hello”;

Sop(s1 == s2); //output: true

ex2:

String str1 = new String(“Java”);

String str2 = new String(“Java”);

Sop(str1 == str2); //output: false

ex3:

String t1 = new String(“Apple”);

String t2 = t1;

Sop(t1 == t2); //output: true

* equals() method will only compares the content. If the content is same in the objects, returns true. Otherwise, returns false.

ex1:

String s1 = “hello”;

String s2 = new String(“hello”);

Sop(s1.equals(s2)); //output: true

ex2:

String t1 = new String(“Dell”);

String t2 = new String(“dell”);

Sop(t1.equals(t2)); //output: false

* equalsIgnoreCase() method will only compares the content, by ignoring the case.

ex1:

String str1 = new String(“Apple”);

String str2 = new String(“apple”);

Sop(str1.equalsIgnoreCase(str2)); //output: true

* compareTo() method will also compares the content only, but returns the output as an int value.
* if the content is equal then returns 0.
* if the first string is less than second string then returns -ve value.
* if the first string is greater than second string then returns +ve value.

ex1:

String g1 = new String(“dell”);

String g2 = new String(“dell”);

Sop(g1.compareTo(g2)); //output: 0

ex2:

String s1 = new String(“Dell”);

String s2 = new String(“dell”);

Sop(s1.compareTo(s2)); //output: -ve integer

ex3:

String str1 = new String(“dell”);

String str2 = new String(“Dell”);

Sop(str1.compareTo(str2)); //output: +ve integer

finding a character/finding index:

charAt(index) : returns character at the index

indexOf(char) : returns index of the char

ex1:

String str = “Lenovo”;

Sop(str.charAt(4)); //output: v

Sop(str.charAt(6)); //IndexOutOfBoundsException

ex2:

String k1 = “cat sat”;

Sop(k1.indexOf(‘s’)); //ouput: 4

Sop(k1.indexOf(‘t’)); //output: 2

Sop(k1.lastIndexOf(‘t’)); //output: 6

ex3:

String str = “cat sat on mat”;

Sop(str.indexOf(‘b’)); //output: -1

Sop(str.indexOf(‘t’, 3)); //ouput: 6

reading a part of a string:

substring(beginIndex, endIndex): returns a string from beginIndex to endIndex – 1.

ex:

String str1 = “The sky is blue”;

String str2 = str1.substring(4, 10);

Sop(str2); //output: sky is

substring(beginIndex) : returns a string from beginIndex to the last character.

ex:

String s = “The sky is blue”;

String s2 = s.substring(4);

Sop(s2); //output: sky is blue

trim() : removes white spaces before the first character/after the last character.

ex:

String user = “ scott”;

Sop(user.equals(“scott”)); //output: false

Sop(user.trim().equals(“scott”)); //output: true

toLowerCase() and toUpperCase():

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

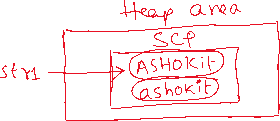
* toLowerCase() converts the letters of a string value to the lower case.
* toUpperCase() converts the letters of a string value to the upper case.
* In Java, String is an immutable object. It means, once the object is created, it will not allow to make any changes to its data.
* So, when you perform any operation on a string object, the result will be stored into a new object.

ex1:

String str1 = “ASHOKit”;

str1.toLowerCase();

S.o.println(str1);



output: ASHOKit

ex2:

String str1 = “HELLO”;

str1 = str1.toLowerCase();

S.o.println(str1);

output: hello



ex3: String str = “java”;

S.o.println(str.toUpperCase());

S.o.println(str);

output: JAVA

java

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

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");

String s3 = s2.intern();

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

System.***out***.println(s1 == s3); //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



Palindrome.java

/\*\*

\* This program is to check whether a string is palindrome or not

\* ex:

\* madam -- it is palindrome

\* level -- it is palindrome

\* test -- it is not a palindrome

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** Palindrome {

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

// connect Scanner with the keyboard input

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

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

String str = scanner.nextLine();

**boolean** flag = *checkPalindrome*(str);

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

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

**else**

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

scanner.close();

}

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

// convert string to lowercase

str = str.toLowerCase();

// convert 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;

}

}

Q) String str = new String();

How many objects are created?

Ans: one object in the heap.

When JVM starts, it creates an empty string in the string pool at the starup only. So, again empty string is not created in the pool.

Q) String str = new String(null);

How many objects are created?

Ans: throws NullPointerException

Q) String str = null;

How many objects are created?

Ans: 0

VowelsCount.java

/\*\*

\* This program counts the vowels and consonants

\* in a given string.

\* ex:

\* str = "ashokit@123"

\* output:

\* vowels count : 3

\* consonants count : 4

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** VowelsCount {

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

//connect Scanner to the keyboard input

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) {

//convert the string into lowercase

str = str.toLowerCase();

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

//convert the string into char[]

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

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

**if** (Character.*isLetter*(ch[i])) {

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

vCount++;

**else**

cCount++;

}

}

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

System.***out***.println("Consonants count : " + cCount);

}

}

Anagrams.java

/\*\*

\* This program checks whether 2 strings are anagrams or not.

\* Anagrams: means, the strings must contain same characters

\* but could be in different order.

\* ex:

\* str1 = "Listen" str2 ="Silent"

\* output: Anagrams

\* str1 = "State" str2 = "Taste"

\* output: Anagrams

\* str1 = "Java" str2 = "Kava"

\* output: Not Anagrams

\*/

**package** com.ashokit;

**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("They are anagrams");

**else**

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

scanner.close();

}

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

//convert the strings to lowercase

str1 = str1.toLowerCase();

str2 = str2.toLowerCase();

//convert the strings into char[]

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

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

//sort the arrays

Arrays.*sort*(ch1);

Arrays.*sort*(ch2);

**if**(ch1.length != ch2.length)

**return** **false**;

**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.

\* ex:

\* String str = "Missisipi";

\* output:

\* m-1

\* i-4

\* s-3

\* p-1

\*/

**package** com.ashokit;

**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();

*findFrequency*(str);

scanner.close();

}

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

//convert the string to lowercase

str = str.toLowerCase();

//convert the string to char[]

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

//sort the array

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);

}

}

}

StringBuffer class

* StringBuffer class is provided in java.lang package.
* StringBuffer class is also provided to perform operations on string values.
* StringBuffer object is mutable, but threadsafe object.
* StringBuffer object is mutable, which means, the object will allow to make changes its content/data.
* StringBuffer doesn’t allow two threads exactly at the same time to modify its conent. So, it is a threadsafe object.
* StringBuffer object can be created only with new keyword.

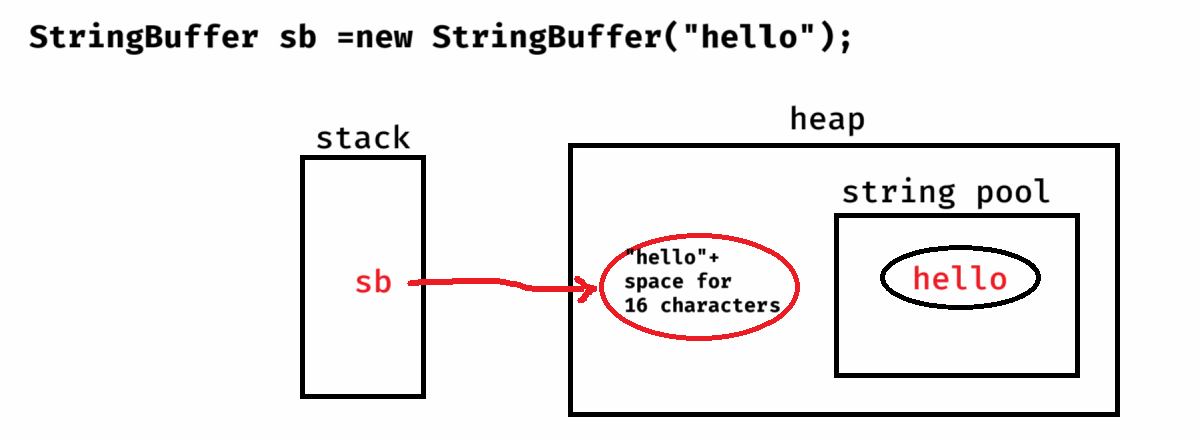
ex:

StringBuffer sb = “Hello”; // error

StringBuffer sb = new StringBuffer(“Hello”); //correct

* When you create a StringBuffer object, first a string object is created in the string pool, if not not exist, then a new StringBuffer object is created in the heap memory.
* The StringBuffer creates additional space for 16 characters in the object.

ex:



* In StringBuffer class, we have length() and capacity() methods.
* length() returns the number of characters in the StringBuffer object.
* capacity() returns the length + addition space.

For example,

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

System.out.println(sb.capacity()); // output: 21

example:

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

System.***out***.println("length : " + sb.length()); //output : 5

System.***out***.println("capacity : " + sb.capacity()); //output: 21

sb.append("hellohellohelloh"); // 16 characters appended.

System.***out***.println("length : " + sb.length()); // output: 21

System.***out***.println("capacity : " + sb.capacity()); //output : 21

sb.append("h"); // 1 more character is appended

System.***out***.println("length : " + sb.length()); // output: 22

System.***out***.println("capacity : " + sb.capacity()); //output: 44

/\*

\* If the length of a StringBuffer object, exceeds the capacity

\* then the capacity will be resized as

\* old capacity \* 2 + 2

\*/

* Suppose, a StringBuffer object is created, like
* StringBuffer sb1 = new StringBuffer();
* then how many objects are created?
* Ans: 1 object is created with empty string as value in the heap.
* For the below code, how many objects are created?

StringBuffer sb1 = new StringBuffer(“hi”);

sb1.append(“ admin”);

Ans: 2 objects.

* For the below code, how many objects are created?

StringBuffer sb1 = new StringBuffer(“dell”);

StringBuffer sb2 = new StringBuffer(“dell”);

* Ans: 3 objects.

Comparing StringBuffer objects:

StringBuffer sb1 = new StringBuffer(“Java”);

StringBuffer sb2 = new StringBuffer(“Java”);

S.o.println(sb1.equals(sb2)); // output: false

Note: In StringBuffer, equals() method doesn’t compare the content in the objects. It just checks whether the two variables are pointing to the same object or not.(Like == operator).

* To compare the StringBuffer objects, you need to convert them into String objects.
* String str1 = sb1.toString();
* String str2 = sb2.toString();
* S.o.println(str1.equals(str2)); // output: true

some key methods of StringBuffer:

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

1. delete(start, end): removes characters from start index to

end-1.

If start or end indexes are negitive, or

start is greater than end, then

StringIndexOutOfBoundsException will be

thrown.

1. insert(index, string): Inserts the characters of the given

string at the given index.

If index is invalid, then

StringIndexOutOfBoundsException will be

thrown.

1. replace(start, end, string): replaces the characters between

start to end-1 indexes with the given string.

1. reverse() : It will reverse the sequence of characters in the

given string value.

// The code to check for string palindrome with reverse() method.

String original = “madam”;

StringBuffer sb = new StringBuffer(original);

sb.reverse();

String reversed = sb.toString();

if(original.equals(reversed))

Sop(“It is palindrome”);

else

Sop(“It is not a palindrome”);

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 come.
* So, it 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.

RemoveDuplicates.java

/\*\*

\* This program removes the duplicate characters

\* from a given string.

\* ex:

\* str = "Lilliput"

\* output: Lilput

\* ex:

\* str = "Missisipi"

\* output: Misp

\*/

**package** com.ashokit;

**import** java.util.Scanner;

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

**private** **static** String removeDuplicates(String str)

{

//create a StringBuilder

StringBuilder builder = **new** StringBuilder();

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

{

**char** ch = str.charAt(i);

**if** (builder.indexOf(String.*valueOf*(ch)) == -1)

{

builder.append(ch);

}

}

**return** builder.toString();

}

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

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

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

String str = scanner.nextLine();

String str2 = *removeDuplicates*(str);

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

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

scanner.close();

}

}

problem:

Write a program to reverse each word of a given string.

ex:

str = “The sky is blue”;

output:

ehT yks si eulb

Compression.java

/\*

\* This program is to perform string compression.

\* example:

\* str = "ababbcdaddc"

\* output: a1b1a1b2c1d1a1d2c1

\*/

**package** com.ashokit;

**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();

String compressedString = *compress*(str);

System.***out***.println("after compression : " + compressedString);

scanner.close();

}

**private** **static** String compress(String str) {

**int** count = 1;

StringBuilder builder = **new** StringBuilder();

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

**if** (i + 1 < str.length() && str.charAt(i) == str.charAt(i + 1)) {

count++;

} **else** {

builder.append(str.charAt(i)).append(count);

count = 1;

}

}

**return** builder.toString();

}

}

NonRepeatingCharacter.java

/\*\*

\* This program is to find the first non repeating character

\* of a given string.

\* example:

\* str = "swiss"

\* output: w

\*/

**package** com.ashokit;

**import** java.util.Scanner;

**public** **class** NonRepeatingCharacter {

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

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

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

String str = scanner.nextLine();

*findNonRepeatingCharacter*(str);

scanner.close();

}

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

// **TODO** Auto-generated method stub

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

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

**char** ch = str.charAt(i);

**if**(str.indexOf(ch, i+1) == -1 && str.lastIndexOf(ch, i+1) == -1)

{

System.***out***.println("The first non-repeating character : " + ch);

flag = **false**;

**break**;

}

}

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

System.***out***.println("No non-repeating character found");

}

}

Problem:

Write a program to sort the characters in a string, without using

Arrays.sort()

(Hint : use bubble sort logic)