Java Full Stack Training

14-02-2022

Day 1

**Introduction**

15-02-2022

Day 2

Ip address : the default localhost or nodename or machinename or 127.0.0.1

ipconfig 168.10.132.56

ipAddress : 0 to 255

0.0.0.0

256 pow 4 number of ip address we can create under one network environment.

Day 3

16-02-2022

Version control tool : These tool help to keep the track or record every changes done in files or folders or application or software etc.

There are different types of version control tools are available in market.

1. Local version control : RCS : Revision control system: in work only one machine.
2. Centralized version control : SVN : We can connection local machine to server machine. After finish our task we will push this code to central machine other people pull that code and push it back to central after done the changes.
3. Distributed version control : in distributed version control they provide as local as well as remote repository. Git is a type of Distributed version control.

Git is a distributed sub version control tool use to manage the source code or application.

Distributed version control system keeps track of software or application revision and allow many developer or programmer or user to work on given project within a maintaining connection to common network.

Git is a open source distributed version control system.

Git provided lot of command those command based upon the Unix.

Create the folder in desktop or any drive

Test Repository

First create any type of files ie test files file or folder can be any type.

These files or folder maintain by files system based upon the OS.

Inside that folder you have to open terminal or git bash

To make this folder as a local repository we have to use the command as

git init (.git folder will create after executed this command). First time only once

.git folder hidden folder consider in Unix or Mac.

ls –a (this command we can execute only in unix or git bash)

git status : This command is use to provide the current status of our repository.

Repository : it is just be like a folder or directory maintain by git.

We have add those files from file system to staging area.

git add filename : it is use to add only one file

Or

git add . : all files or folder present in current directory

git status

we want to push this file from staging area to local repository

git commit –m “1st task done”

first time we will get the error.

If you get the error you have to add your name and email id (only one time).

git config --global user.name “Raj”

git config --global user.email “[raj@gmail.com](mailto:raj@gmail.com)”

git commit –m “1st task done”

git status

do the changes ie create new file or folder or do some changes in existing file

git status

git add .

git status

git commit –m “commit msg”

staging area : this area provided by git where the files or folder hold before commit.

remote repository : remote repository is just like a network folder which help to share the data between two teams.

github, gitlabs, aws(code commit) or azure etc.

github remote repository

to connect local repository to remote repository we have to run the command as

git remote add origin URL

git remote add origin https://github.com/Kaleakash/test\_app.git

you have to execute this command only once

git push -u origin main

git push -u origin master

to generate the token

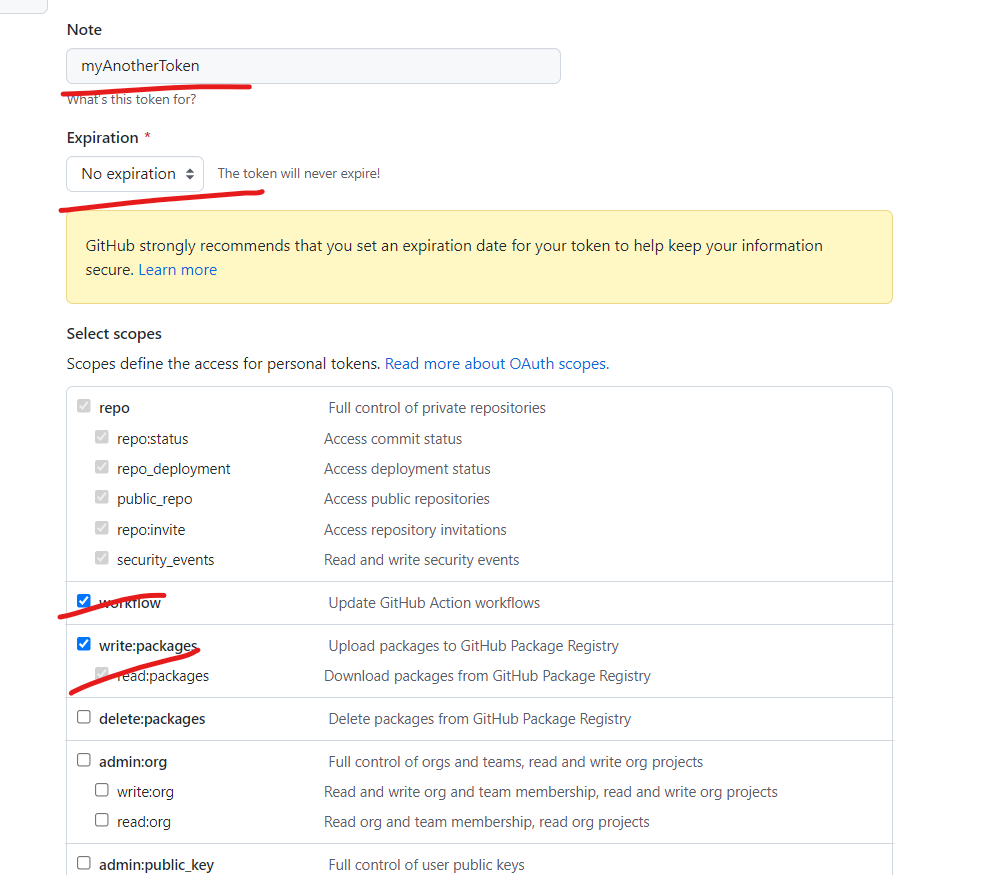
In Github account

Right click in setting option in Remote repository (right side on top of web page +)

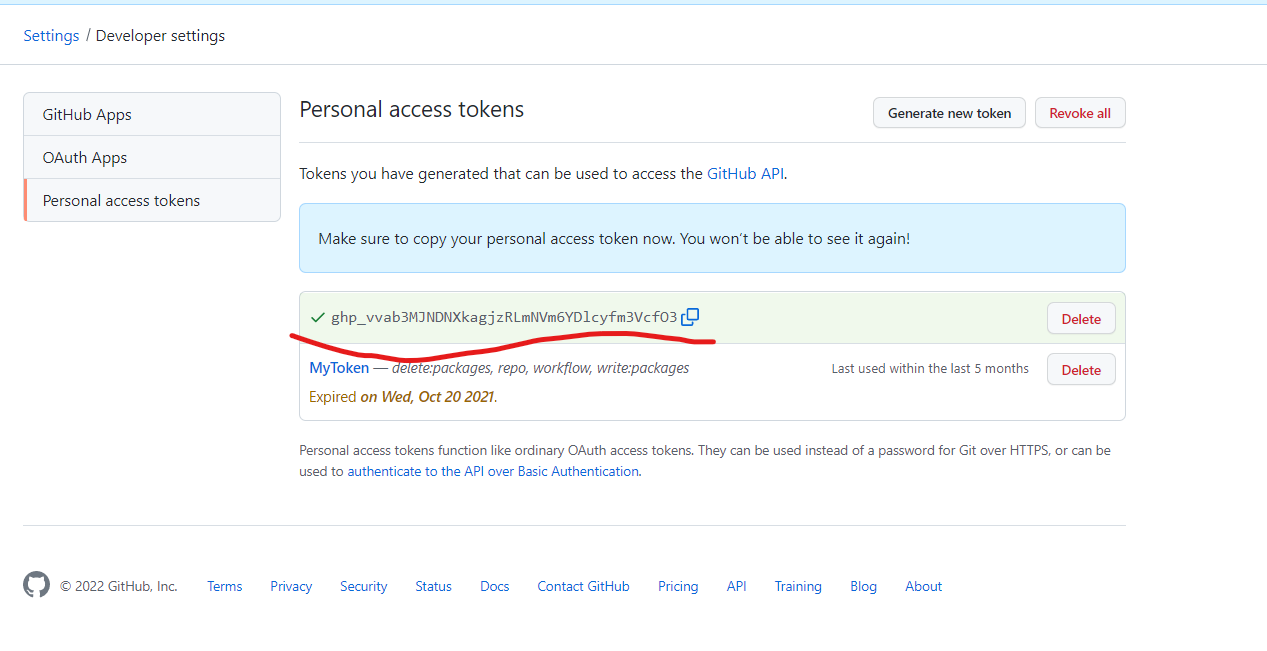
search developer setting option

Click on personal access token option

Generate new token



Then copy the token



Got check local machine connect to which remote repository

git remote show origin

To remove remote origin from local repository run the command as

git remote remove origin

git remote add origin https://github.com/Kaleakash/test\_app.git

git remote add origin <https://token@github.com/Kaleakash/test_app.git>

ghp\_XfNggV9m2yhlbUqDRqpFhjhJq9nuIj211UZM

git remote add origin <https://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@github.com/Kaleakash/test_app.git>

git remote add origin [https:// ghp\_vvab3MJNDNXkagjzRLmNVm6YDlcyfm3VcfO3@github.com/Kaleakash/test\_app.git](https://%20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@github.com/Kaleakash/test_app.git)

git push –u origin main

or ­

git push –u origin master

Day 4

17-02-2022

git branch : git branch is like a pointer which hold more than one commit details.

By default one branch created ie master or main.

To check branch we have to use the command as

git branch

command to create the branch

git branch branchName

to switch from one branch to another branch

git checkout branchName

git checkout –b branchName (it will create the branch and switch to created branch)

if we are planning to do any task don’t d do in main or master (default branch).

If we want to do any task we have create the branch and do that task inside that branch.

Merge the branch

git merge branchName

git branch –D branchName

First check the default branch it may be main or master

git status (status must be clean)

create the branch

you are main or master.

git checkout –b A

do some task like created file, modify file or delete file etc

git add .

git commit –m “msg”

please switch to main or master branch.

Git checkout master or main

git checkout –b B

do some task like created file, modify file or delete file etc

git add .

git commit –m

git checkout master or main

if you want to add the branch A or B or etc code

in main branch or master branch run the command as

git merge branchName

after that you can delete user defined branch

git branch –D branchName

git clone URL : using this command we can download the all git repository in local machine.

This command is use to download existing repository to local repository (first time).

Next time if you want new updated which present in remote repository we have to open the git bash terminal insider a repository folder(.git folder must be present) run the command as or inside a clone folder.

git pull : this command is use to get new updated present in existing repository.

10 min time to push , clone and pull with our own repository

git rebase command

Java :

Java is programming language

Java is object oriented programming language.

Java is platform independent and pure object oriented programming language.

Difference procedure language and Object oriented language.

C

C++

OOPs

Object : any real world entity.

State or properties 🡪 have 🡪 attribute, variable, fields etc

Person

Behaviour 🡪 do/does -🡪 functions / methods

Place

Bank

Animal

Car

Etc

Object is concept.

Class : blue print of object or template of object or user-defined data types which help to create the object or memory.

Syntax of class

class className {

fields;

methods;

}

ClassName referenceName = new ClassName();

Syntax class

class ClassName {

pre-defined main method.

}

class Test {

public static void main(String args[]) {

System.out.println(“Welcome to Java…”);

}

}

Class name must be follow Pascal naming rules.

1. If class contains one word first letter must be upper case.
2. If class contains more than one word. Each word first letter upper case.
3. No space between two word.

Install the java it may be 8, 11, 15 m 16 or 17

Then check in folder

C:\Program Files\Java

C:\Program Files\Java\jdk-11.0.12\bin

Java provide jdk (java development kit) which help to compile and run the program.

javac (Java compiler ) : it is use to compile program

java (java interpreter ) : it is use to run the program

if

java --version

javac

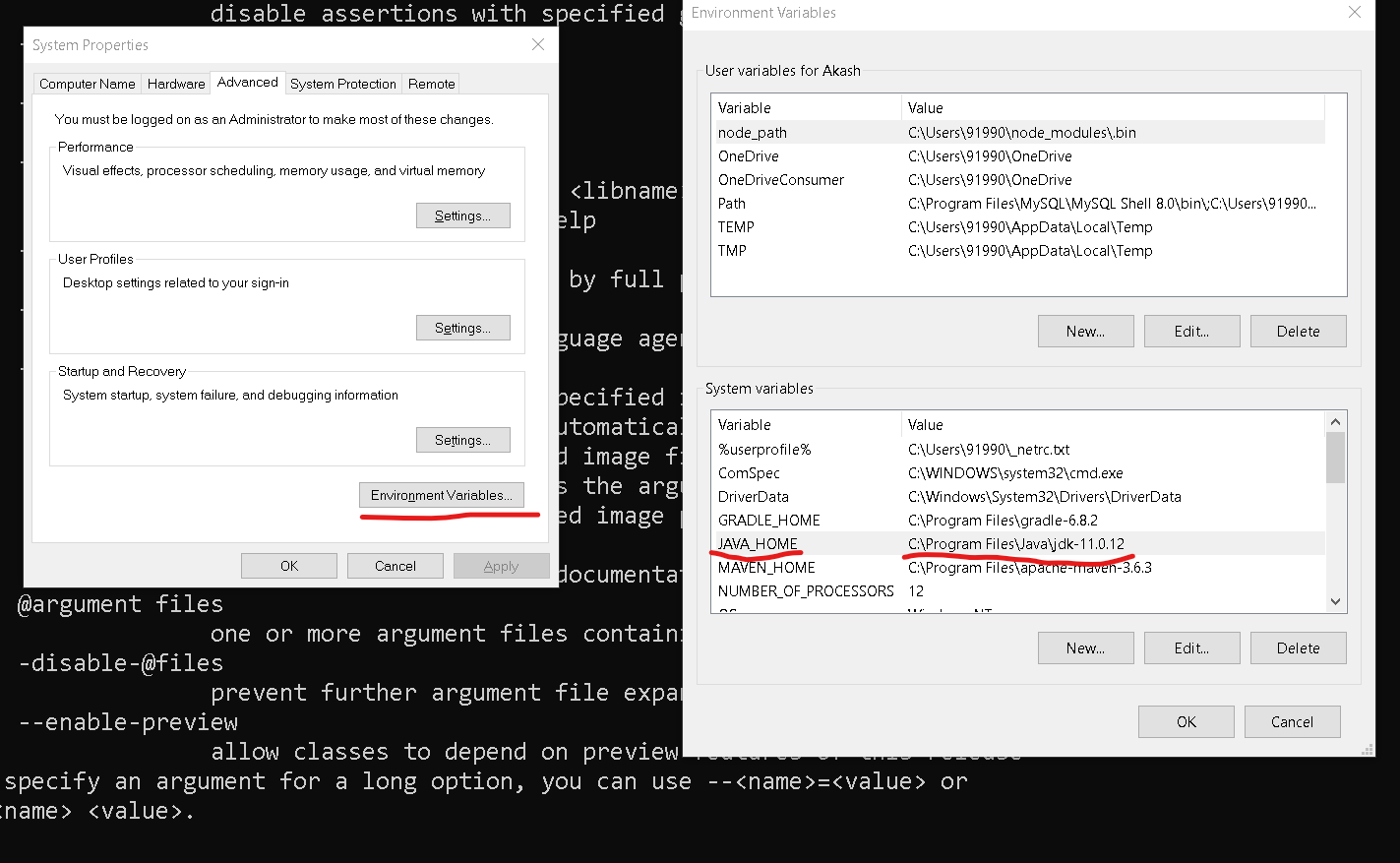
java

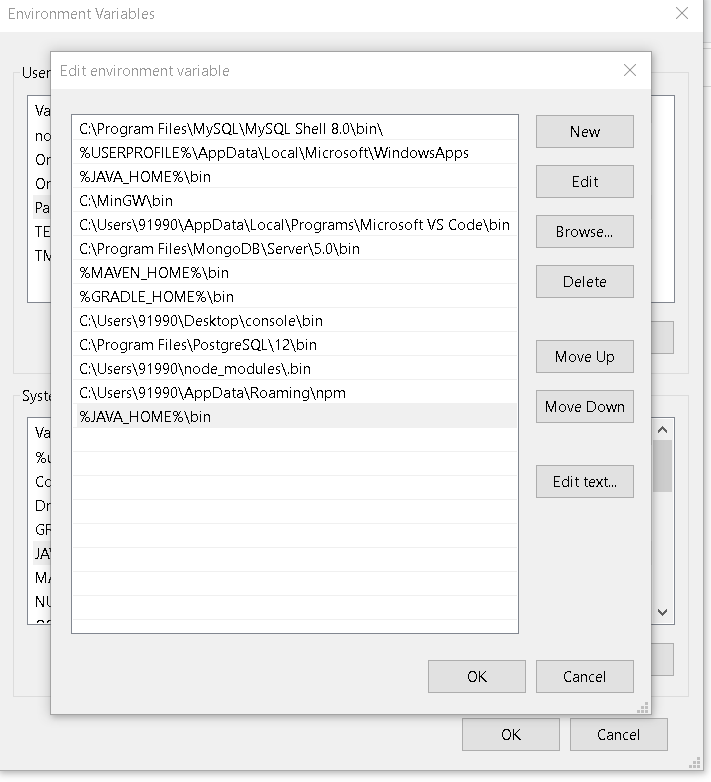
command not found

Then copy path till bin folder with respective version

C:\Program Files\Java\jdk-11.0.12

Right click on my computer icon and properties and search environment variable.





Then open new command prompt

Then check

java --version

javac

java

open any editor

class Test {

public static void main(String args[]) {

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

}

}

Save the program with name Test.java (please save the program with className).

Open the command prompt or terminal

To compile the program

javac Test.java

To run the program

java Test

Day 5

18-02-2022

Welcome Program

class Test {

public static void main(String args[]) {

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

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

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

}

}

Data types : Data type is a type of data which tells what type of value it can hold.

2 types of data types.

1. Primitive type : it is use to store only value
2. Non primitive type or reference data type : it use to store value as well as reference of another data types.

8 types

1. byte 1 byte it is use to store value without decimal point
2. short 2 byte it is use to store value without decimal point
3. int 4 byte it is use to store value without decimal point
4. long 8 byte : it is use to store value without decimal point
5. float 4 byte : it is use to store the value with decimal
6. double 8 byte : it is use to store the value with decimal
7. char 2 byte : single character
8. boolean 1bit : true or false.

Variable declaration syntax

datatype varaibleName;

simple example to declare the variable and display the value

class Test {

public static void main(String args[]) {

int a;

int b=10;

double d =10.20;

boolean res = true;

System.out.println(b);

System.out.println("The value of b is "+b);

System.out.println("The value of res is "+res);

System.out.println("The value of d is "+d);

}

}

Type casting : converting from one data type another data is known as type casting.

2 types

1. implicit : automatically convert
2. explicit : we have to convert.

Int family

--------------------------🡪Implicit -------------------------------🡪

byte short int long

🡨------------------------Explicit ------------------------------------

datatype variableName = (type)variableName;

Simple example of type casting

class Test {

public static void main(String args[]) {

byte a=10;

short b=a; // implicit type castting

System.out.println(a);

System.out.println(b);

short c=10; // 2 byte

//byte d =c;

byte d = (byte)c; // explicit type casting

System.out.println(c);

System.out.println(d);

}

}

Another Example with type casting

class Test {

public static void main(String args[]) {

byte a=127; // range -128 to 127

short b=a; // implicit type castting

System.out.println(a);

System.out.println(b);

short c=129; // 2 byte

//byte d =c;

byte d = (byte)c; // explicit type casting

System.out.println(c);

System.out.println(d);

}

}

Implicit

Int ----------------------float

Explicit

In java every decimal number by default consider as double. Double size is 8 btye.

int to char

chat to int

int char

operator

arithmetic operator : +, -, \*, /, %

conditional operator : >, >=, <, <=, ==, != (relational operator )

logical operator : &&, ||, !

assignment operator : =

increment and decrement : ++, --

pre increment ++a; pre decrement --a

post increment : a++ post decrement a--

ternary operator condition ? true:false

class Test {

public static void main(String args[]) {

int a=10;

System.out.println(a); //a

a++;

System.out.println(a); //a

++a;

System.out.println(a); //a

}

}

Pre-increment and post increment by 1 if we are not using the variable inside some expression or displaying the value.

Pre : increment and assign or display

Post : display or assign and then increment.

operator example

class Test {

public static void main(String args[]) {

int a=10;

int b=20;

int c=10;

int sum = a+b;

System.out.println("Sum of two number is "+sum);

boolean res1 = a>b;

boolean res2 = a==c;

boolean res3 = a>b && a>c; // both condition must be true. then result is true

boolean res4 = a<b || a>c; // any one condition must be true then result is true

System.out.println(res1);

System.out.println(res2);

System.out.println(res3);

System.out.println(res4);

}

}

Increment and decrement operator example

class Test {

public static void main(String args[]) {

int a=10;

int b=a; // 10 assign

System.out.println(b); //b 10

b=a++; // assign and then increment

System.out.println(b); //b 10

b=++a; //b 12

System.out.println(b); //b 12

}

}

class Test {

public static void main(String args[]) {

int a=10;

int b =50;

int res = a>b?a:b;

System.out.println(res);

}

}

if statement

it is use to execute the set of statement depending upon the conditions.

Simple if

If(condition) {

}

If else

**If(condition) {**

**}else {**

**}**

**If else if**

**If(condition) {**

**}else if(condition) {**

**}else if(condition) {**

**}else {**

**}**

**class Test {**

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

**int age=16;**

**if(age>18) {**

**System.out.println("You can Vote!");**

**}else {**

**System.out.println("You can't Vote!");**

**}**

**}**

**}**

switch statement : it is use to execute the block statement base upon user requirements.

class Test {

public static void main(String args[]) {

int label =10;

switch(label) {

case 1: System.out.println("1st block");

break;

case 2: System.out.println("2nd block");

break;

case 3: System.out.println("3rd block");

break;

default : System.out.println("Wrong choice");

break;

}

System.out.println("finish");

}

}

looping

looping is use to execute the set of statement again and again till the condition become false.

While loop

Do while loop

For loop

1 2 4

for(initialization ; condition ; increment/ decrement ) {

body of the loop; 3

}

class Test {

public static void main(String args[]) {

// while loop : Entry loop

/\*int i=1,n=10; //initialization

while(i<=n) { // condition

//System.out.println("Hello");

System.out.println("i="+i);

i++;

}\*/

// do while loop : Exit loop

/\*int i=1,n=10;

do {

System.out.println("i="+i);

i++;

}while(i>=n);\*/

// for loop

Fixed iteration

for(int i=0;i<=10;i++) {

System.out.println("i="+i);

}

}

}

For each loop or enhanced loop

This type of loop we use to retrieve the value from array

**for(datatype variableName : arrayName) {**

**}**

reference data types

1. array : array is a type of reference data type which is use to store the value of same types.

syntax to declare the array in java

datatype arrayName[];

int abc[];

int []xyz;

in every java to retrieve the value from array we have to use the index positon.

Index position start from zero.

Array example declaration, initialization, retrieve value using for loop and Enhanced loop

class Test {

public static void main(String args[]) {

int a;

int abc[];

int b=20;

int xyz[]={10,20,30,40,50,60,100,240,70,57,101,78,46,76,24};

System.out.println(xyz[0]);

System.out.println(xyz[1]);

int leng = xyz.length; // it is use to find the size of the array;

System.out.println("Size of the array is "+leng);

System.out.println("Retrieve the elements using for loop");

for(int i=0;i<xyz.length;i++) { // we can customize base upon our requirement.

System.out.println(xyz[i]);

}

System.out.println("Retrieve the elements using enhanced loop");

for(int n : xyz) { // it is use to retreive from begining to end one by one

System.out.println("Value is "+n);

}

}

}

Memory creation for array

Syntax

datatype []arrayName=new datatype[size];

int []abc=new int[10]; in Java

int abc[10]; in C or C++

in java using new keyword we can create dynamic memory.

Array memory creation and assign the value and display the value.

class Test {

public static void main(String args[]) {

int []abc=new int[10];

System.out.println("Size of array is "+abc.length);

abc[0]=100;

abc[1]=200;

System.out.println("0 position value is "+abc[0]);

System.out.println("1 position value is "+abc[1]);

int temp=100;

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

abc[i]=temp++;

}

System.out.println("Display");

for(int n:abc) {

System.out.println(n);

}

}

}

Taking the value through keyboards

1. using Scanner class
2. using DataInputStream class
3. using BufferedReader class
4. using command line

Scanner : Scanner is a pre-defined class which provide set of methods which help to scan the value through keyboards.

Syntax to create the Scanner class object.

Scanner obj = new Scanner(System.in);

Scanner is a pre-defined class part of util package.

Package is a collection of classes and interfaces.

obj.nextByte()

obj.nextShort()

obj.nextInt();

obj.nextFloat()

obj.nextDouble()

import java.util.Scanner;

class Test {

public static void main(String args[]) {

Scanner obj = new Scanner(System.in);

System.out.println("Enter the value of a ");

int a = obj.nextInt(); // nextInt() is pre-defined method which help to scan integer value through keyboard.

System.out.println("The value of a "+a);

}

}

but no nextChar()

class : class also known as reference data types.

it may be pre-defined or user-defined class.

String is a pre-defined class or reference data types.

obj.next(); This method is use to scan only one word

The string value through keyboards

import java.util.Scanner;

class Test {

public static void main(String args[]) {

Scanner obj = new Scanner(System.in);

String name = "Ravi";

String msg = "Welcome to Java Training";

System.out.println(name);

System.out.println(msg);

System.out.println("Enter the name");

String fname = obj.nextLine(); // it is use to scan string or name through keyboards.

System.out.println("Your name is "+fname);

}

}

Taking array value through keyboards using Scanner

import java.util.Scanner;

class Test {

public static void main(String args[]) {

Scanner obj = new Scanner(System.in);

System.out.println("how many details you want to store");

int n = obj.nextInt();

int []id = new int[n];

String []names=new String[n];

System.out.println("Enter id and name one by one");

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

System.out.println("Enter the "+(i+1)+" Id ");

id[i]=obj.nextInt();

System.out.println("Enter the "+(i+1)+" Name ");

names[i]=obj.next();

}

System.out.println("All details are ");

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

System.out.println("id is "+id[i]+" Name is "+names[i]);

}

}

}

Day 6

21-02-2022

OOPs concept using Java

Download Eclipse for JEE

object : any real word entity.

Properties or state -🡪 have -🡪 fields / variables

Person

Behaviour 🡪do/does 🡪 functions / methods

Place

Bank

Animal

Customer

name, price, color etc

Car

start(), appliedGear(), moving(), stop() etc

class : Blue print of object or template of object or user-defined or reference data type which is use describe object.

syntax to create the object or memory

ClassName refereneName = new ClassName();

Car innova = new Car(); // heap memory.

Simple example for create the object

class Car {

String name;

float price;

int wheel;

void start() {

System.out.println("Car Started..");

}

void appliedGear() {

}

void moving() {

}

void stop() {

System.out.println("Car Stop");

}

}

class CarTest {

public static void main(String args[]) {

Car innova = new Car();

innova.start();

innova.stop();

}

}

ClassName objectRefereceName = new ClassName();

objectReferenceName.method();

Types of variable or fields.

3 types

1. instance variable
   1. The variable which declared outside a method including main method is known as instance variable.
   2. Instance variable hold default value according to their data types. int family 🡪0, float family 🡪0.0, char 🡪 white space, boolean 🡪 false, String 🡪 null.
   3. Instance variable we can access directly in all methods. But method must be part of same class and it must be non static method.
2. local variable
   1. The variable which declared inside a method is known as local variable.
   2. Local variable doesn’t hold default value. we have to initialize.
   3. The scope of the local variable within that block where it declared.
3. static variable

**Example for instance and local variable.**

**class Car {**

**String name;**

**float price;**

**int wheel;**

**void start() {**

**int temp=123;**

**System.out.println("Car Started..");**

**System.out.println("name "+name);**

**System.out.println("price "+price);**

**System.out.println("wheel "+wheel);**

**System.out.println("Temp "+temp);**

**}**

**void stop() {**

**String msg="Welcome";**

**System.out.println("Car Stop");**

**System.out.println("name "+name);**

**System.out.println("price "+price);**

**System.out.println("wheel "+wheel);**

**System.out.println("message "+msg);**

**}**

**}**

**class CarTest {**

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

**Car innova = new Car();**

**innova.start();**

**innova.stop();**

**}**

**}**

Assign the value for instance variable and display the values.

class Car {

String name;

float price;

int wheel;

void displayCarDetails() {

System.out.println("Car Details");

System.out.println("name "+name);

System.out.println("price "+price);

System.out.println("Wheel "+wheel);

}

}

class CarTest {

public static void main(String args[]) {

Car innova = new Car();

innova.displayCarDetails();

innova.name = "crysta";

innova.price = 3000000;

innova.wheel = 4;

innova.displayCarDetails();

}

}

Constructor : It is a type of special method which help to create the object.

Pts

1. Constructor have same name as class itself.
2. Don’t write the return type for constructor not even void also.
3. Constructor no need to call it will call automatically when we create the object.

Empty constructor

class Car {

Car() {

System.out.println("Car class memory created...");

}

void displayCarDetails() {

System.out.println("Car Details method");

}

}

class CarTest {

public static void main(String args[]) {

Car innova = new Car();

innova.displayCarDetails();

}

}

Parameterized constructor

In the life of the object if you want to perform any task only one time that type of code we have to write inside a empty constructor or parameterized constructor.

In the life of the object if you want to perform any task more than one time that type of task we have to write inside a methods.

class Operation {

int a,b,sum;

Operation() {

a=10;

b=20;

}

Operation(int x, int y){

// default value of a and b is 0

//x=a;

//y=b;

a=x;

b=y;

}

void setValue(int x, int y) {

a=x;

b=y;

}

void add() {

sum = a+b;

}

void display() {

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

}

}

class Test {

public static void main(String args[]) {

Operation op1 = new Operation(); op1.add(); op1.display();

Operation op2 = new Operation(); op2.add(); op2.display();

Operation op3 = new Operation(1,2); op3.add(); op3.display();

Operation op4 = new Operation(4,5); op4.add(); op4.display();

Operation op5 = new Operation(); op5.setValue(100,200); op5.add(); op5.display();

Operation op6 = new Operation();

op6.setValue(11,22);

op6.setValue(111,222);

op6.add();

op6.display();

}

}

Encapsulation : binding or wrapping data ( variables / fields ) and code (function / methods) in a single unit is known as Encapsulation.

Ex : class

Example

class Employee {

String name;

float salary;

void display() {

System.out.println("Name is "+name);

System.out.println("Salary is "+salary);

}

}

class Test {

public static void main(String args[]) {

//display();

//salary = 12000;

Employee emp = new Employee();

emp.name = "Raj Deep";

emp.salary = 12000;

emp.display();

}

}

To make property secure we have to use private access specifiers with variable.

If variables are private we can’t change the value of variable directly as well as through object.

If local variable and instance variable have same name local variable hide the visibility of instance variable. If you want to refer to instance variable we have to use the keyword as this.instancevarible name.

Example

class Employee {

private String name;

private float salary;

void setValue(String name, float salary) {

//name = name; localvarible = localvariable

//salary = salary;

this.name = name;

//this.salary = salary;

if(salary<=0) {

this.salary = 8000;

}else {

this.salary = salary;

}

}

void display() {

System.out.println("Name is "+name);

System.out.println("Salary is "+salary);

}

}

class Test {

public static void main(String args[]) {

//display();

//salary = 12000;

Employee emp = new Employee();

//emp.name = "Raj Deep";

//emp.salary = -12000;

emp.setValue("Raj Deep",-12000);

emp.display();

}

}

Inheritance : Inheritance is use to inherits or acquire properties and behaviour of old class to new class.

class OldClass { super class or base class or parent class

properties

behaviour

}

class NewClass extends OldClass{ sub class or derived class or child class.

properties

behaviour

}

Through sub class objet we can access its own properties as well as behaviour and super class properties and behaviour.

Simple inheritance example

class A {

void dis1() {

System.out.println("A class method");

}

}

class B extends A{

void dis2() {

System.out.println("B class method");

}

}

class Test {

public static void main(String args[]) {

A obj1 = new A();

obj1.dis1();

B obj2 = new B();

obj2.dis2();

obj2.dis1();

}

}

Types of inheritance

1. Single inheritance : one super class and one sub class

Class A { }

Class B extends A { }

1. Multilevel inheritance : one super class and n number of sub classes extends one by one

Class A { }

Class B extends A { }

Class C extends B { }

Class D extends C { }

1. Hierarchical inheritance: one super class and n number of classes directly connected to super class.

Class A { }

Class B extends A { }

Class C extends A { }

1. Multiple inheritance : more than one super class and one sub class

Class A { }

Class B { }

Class C extends A, B { } : Wrong in Java : Java doesn’t support multiple inheritance. This type of inheritance we can achieve using interface.

OOPs relationship

1. Is a relationship
2. Has a relationship

Manager is a Employee

Developer is a Employee

ProjectManager is a Manager

Super class must be generic.

class Employee {

id,name,salary

Scanner obj = new Scanner(System.in);

readEmp() : to receive the value

disEmp() : display the value

}

class Manager extends Employee {

numberOfEmp;

Address add =new Address();

readMgr() : to receive details

disMgr() : to display the details.

}

class Developer extends Employee {

techName:

readDev() read developer details

disDev() dis developer details.

}

class ProjectManager extends Manager{

numberOfProject;

readPmgr()

disPmgr()

}

class Address {

city

state

Scanner obj = new Scanner()

readAdd() : receive the address

disAdd() : display the address

}

Has a relationship means inside one class creating the object of another class is known as has relationship.

Has a relationship divided into 3 types

1. Association
2. Aggregation
3. Composition

Association : to make the has a relationship. Inside one class we have to create the object of another class.

class A {

B obj1 = new B(); 0 or 1 or many

}

class B {

A obj2 = new A(); 0 or 1 or many

}

Relationship

1. 1
2. 0
3. Many

May 1

Aggregation : It is a type of association which is known as week association.

class Manager {

Address ladd = new Address(); 1

Address padd = new Address();

}

class Address {

}

Composition : it is a type of association which is known as strong association.

class Student {

StudentHistory sh = new StudentHistory();

}

class StudentHistory {

}

Polymorphism : : One name many forms or many implementation

1. types of polymorphism

Compile time polymorphism

Static binding or early binding

Method Overloading : the method have same name but different parameter list ie type of parameter list or number of parameter list must be different is known as method overloading.

Run time polymorphism

Dynamic binding or late binding

Method Overriding

Day 7

22-02-2022

Simple Example through eclipse IDE

Package is a collection of classes and interfaces. Package name must be in lower case

Method Overriding : Method have same name and same method signature( number of parameter list, type of parameter list and return type must be same). To achieve the method override we require inheritance.

If sub class don’t like code provided super class then sub class can write same method with different logic.

Example

**package** com;

**class** Bike {

**void** speed() {

System.***out***.println("60km/hr");

}

}

**class** Honda **extends** Bike { // re-usability of speed();

**void** color() {

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

}

}

**class** Pulsar **extends** Bike {

**void** speed() { // this class override speed method

System.***out***.println("90km/hr");

}

**void** color() {

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

}

}

**class** Tvs **extends** Bike {

**void** color() {

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

}

**void** speed() {

**super**.speed(); // the code present in super class +

System.***out***.println("10km/hr"); // sub class merge

}

}

**public** **class** MethodOverridingTest {

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

Honda hh = **new** Honda(); hh.speed(); hh.color();

Pulsar pu = **new** Pulsar(); pu.speed(); pu.color();

Tvs tv = **new** Tvs(); tv.speed(); tv.color();

}

}

We can write a method without body but method must abstract.

abstract : abstract is a keyword we can use with method and class but not with variable.

1. abstrat method : The method without body or without curly braces or incomplete method is known as abstract method.

Syntax

abstract returnType methodName(parameterList);

abstract void speed();

1. if class contains abstract method we have to declare the class as abstract.

Syntax

abstract class className {

}

1. Whichever class extends abstract class that class must be provide the body for all abstract method mandatory that class can ignore only if that class itself is abstract class.
2. Abstract class we can’t create the object.
3. Abstract class can contains normal as well as abstract method. it can contains zero or 1 or many abstract method.
4. If method is abstract then class must be abstract but not mandatory class abstract it must be contains only abstract method.
5. Abstract class can contains default as well as parameterized constructor. Because abstract class can contains normal ie instance variable.

final keyword : final keyword we can use with variable, method and class.

1. final variable : to declare constant variable in java we use final keyword. Final variable value we can’t change. If variable is final we have to initialize mandatory.

final int A=10;

A=30; // Error

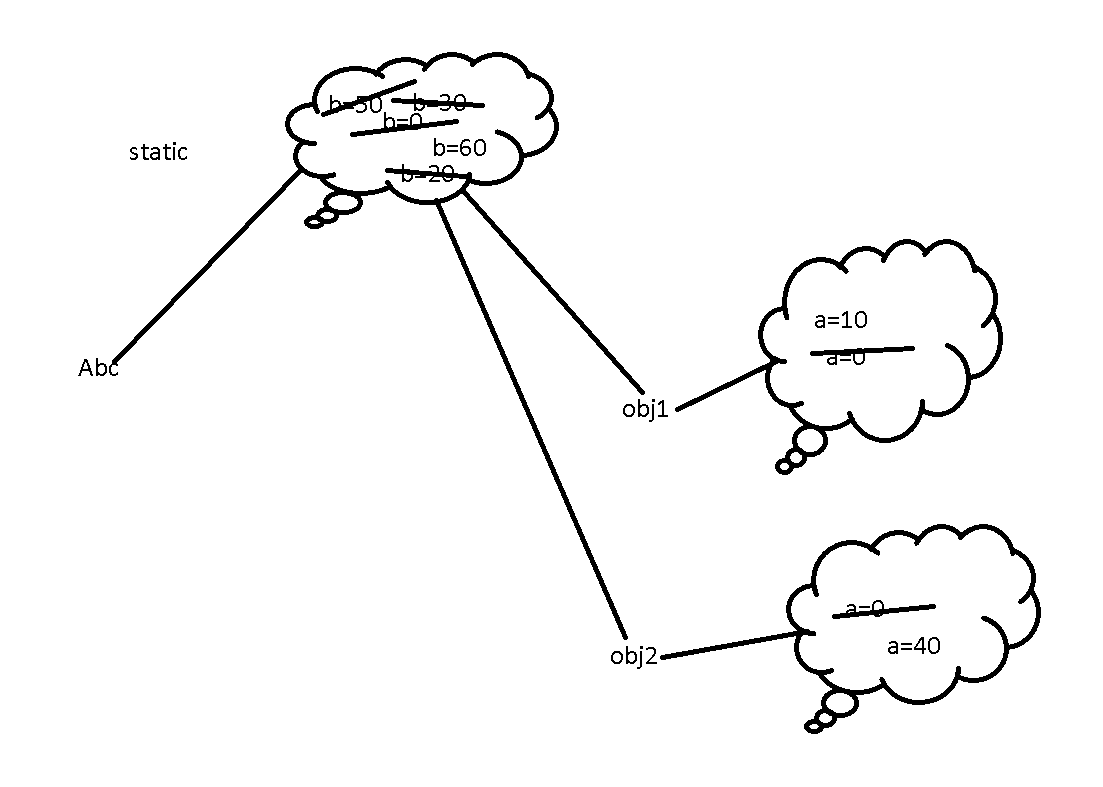
1. final method : if method is final we can’t override that method. But we can use it.
2. final class : if class is final we can’t inherit or extends that class.

static keyword static keyword we can use with variable and method but not with class(if class is inner class we can use static keyword but not for outer class).

1. if variable is static we can assign the value for that variable using class name as well as through object.
2. if method is static we can call that method with the help of class name as well as through object.
3. inside static method we can access only static variable but inside non static method we can access both static as well as non static variable directly.

Every class only one static memory present.

Every class we will get number of instance memory or heap base upon number of object created.



Static is like a global to all object.

Employee

Id, name,salary --🡪 instance

projectId, clientId, managerId 🡪 static

interface : interface is known as 100% pure abstract class till java7.

Syntax to declare the interface

Interface interfaceName {

fields;

methods

}

By default all variable in interface are public, static and final.

By default all methods in interface are public and abstract.

interface Abc {

public static final int A=0;

static final int B=0;

final int C=0;

int D=0;

public abstract void dis1();

public void dis2();

abstract void dis3();

void dis4();

}

Interface contains only final or constant variable as well as abstract method.

Like a class one interface can extends another interface. But interface can extends more than one interface.

Class always implements interface. Class can implements more than one interface.

interface A {

int X=10;

void dis1();

}

interface B {

int Y=20;

void dis2();

}

Interface C extends A,B{ // Multiple inheritance.

int Z=30;

void dis3();

}

class D implements A,B {

}

Whichever class implements any interface what class must be provide the body for all method belongs to that interface.

One class can extends only one class 1 to 1

Interface can extends more than one interface. 1 to many

Class can implements more than one interface. 1 to many

Interface can’t extends or implements to class

Class can’t implements one or more than one class.

Interface can’t implements to another interface.

While providing body for interface method in class we have to write the public access specifiers for interface method.

**Interface Example**

**package** com;

**interface** A {

**int** ***X***=10;

**void** dis1();

}

**interface** B {

**int** ***Y***=20;

**void** dis2();

}

**interface** C **extends** A,B{ // Multiple inheritance.

**int** ***Z***=30;

**void** dis3();

}

**class** D **implements** A,B {

**public** **void** dis1() {

System.***out***.println(" A interface method");

}

**public** **void** dis2() {

System.***out***.println("B interface method");

}

}

**public** **class** InterfaceTest {

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

D obj1 = **new** D();

obj1.dis1();

obj1.dis2();

}

}

Differences between interface and abstract class.

1. abstract class can contains normal as well as final variable but interface contains only final variable.
2. Abstract class can contains normal as well as abstract method but interface contains only abstract method.
3. Normal class can extends only one abstract class But normal class can implements more than one interface.
4. Abstract class can contains default as well as parameterized constructor but interface doesn’t contains default constructor.
5. Using abstract class we can achieve partial abstract using interface we can use 100% abstraction.

Common between interface and abstract class

1. We can’t create the object of interface as well as abstract class.
2. Whichever class extends/implements abstract class as well as interface that class must be provide the body for all abstract method belong to that interface or abstract class.

Day 8

23-02-2022

Run time polymorphism using object creation.

1st Example

**package** abc;

**class** A {

**void** dis1() {

System.***out***.println(" A class own dis1 method");

}

}

**class** B **extends** A{

**void** dis1() {

System.***out***.println(" B class override dis1 method");

}

**void** dis2() {

System.***out***.println(" B class own dis2 method");

}

}

**public** **class** DemoTest1 {

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

// B obj = new B();

// obj.dis1();

// A obj1 = new A(); obj1.dis1(); //obj1.dis2()

// B obj2 = new B(); obj2.dis1(); obj2.dis2();

A obj3 = **new** B(); // sub class object super class reference : it is possible

// // with help of super class refeference we can call only those methods which belong

// // super class or overrided methods.

obj3.dis1();

// //obj3.dis2();

// //B obj4 = new A(); // super class object sub class refeference not possible

}

}

2nd Example

**package** mno;

**abstract** **class** A {

**abstract** **void** dis1();

}

**class** B **extends** A{

**void** dis1() {

System.***out***.println(" B class override dis1 method");

}

**void** dis2() {

System.***out***.println(" B class own dis2 method");

}

}

**public** **class** DemoTest1 {

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

//A obj1 = new A();

//B obj2 = new B(); obj2.dis1(); obj2.dis2();

A obj3 = **new** B(); // creating object of sub class object which is normal class and creating reference of super class

// it can be normal class as well as abstract class.

obj3.dis1(); // obj2.dis2();

}

}

3rd example

**package** xyz;

**interface** A {

**void** dis1();

}

**class** B **implements** A{

**public** **void** dis1() {

System.***out***.println(" B class override dis1 method");

}

**void** dis2() {

System.***out***.println(" B class own dis2 method");

}

}

**public** **class** DemoTest1 {

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

//A obj1 = new A();

//B obj2 = new B(); obj2.dis1(); obj2.dis2();

A obj3 = **new** B(); // creating object of sub class object which is normal class and creating reference of super class

// it can be normal class as well as interface

obj3.dis1(); // obj2.dis2();

}

}

This keyword and super keyword.

If local variable and instance variable have same name local variable hide the visibility of instance variable if you want to refer instance variable we have to use this.varibleName.

If sub class variable and super class variable have same name then sub class variable hide the visibility of super class variable if you want to refer to super class variable we have to use super.varaibleName

**package** com;

**class** A {

**int** x=100;

}

**class** B **extends** A {

**int** x=200;

**void** dis1() {

**int** x = 300;

System.***out***.println("x local variable "+x);

System.***out***.println("x instance variable "+**this**.x);

System.***out***.println("x super class variable "+**super**.x);

}

}

**public** **class** DemoTest {

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

B obj = **new** B();

obj.dis1();

}

}

this() and super()

this() : it is use to do constructor chaining or calling same class constructor or linking one constructor to another constructor.

this() must be first statement inside constructor. Which help to do re-usability of constructor logic.

super(): super() is use to call sub class constructor to super class constructor. By default every sub class constructor super() is present which always call super class empty constructor.

Super() must be first statement inside a constructor.

Access specifiers Access specifiers use to expose the visibility of variable, method and class.

4 types

1. private : We can use private access specifiers with instance variable, static variable, non static method static method, constructor but we can’t use with class and local variable.

scope : private within a same class.

1. default(nothing) or no access specifiers : we can use with all

scope : within a same package.

1. protected : We can use protected access specifiers with instance variable, static variable, non static method static method, constructor but we can’t use with class and local variable.

scope : within a same package other package if it is sub class

1. public We can use private access specifiers with instance variable, static variable, non static method static method and class(in one file it may be notepad or ide editor one only class we can make public other classes must be default) but not with local variable.

scope : public same package as well as other package.

package : package is a collection of a classes and interfaces.

Package divided into two types.

User defined

Pre-defined or built in packages.

education

school college

Attendance Attendance

Package is just like directory or folder. Which two file or program have same name but different purpose or different functionality purpose.

Day 9

24-02-2022

Access specifiers while overriding the methods.

Super class / interface Sub class

Methods

public public

protected public

protected

default (nothing) public

protected

default

private we can’t override that method

Note : private and static method we can’t override.

Pre-defined package

lang package

by default every java program imported lang package.

By default every java program extends Object class.

All pre-defined directly or indirectly extends Object class.

Super class for all classes it may be pre-defined or user-defined Object.

class A {

}

class B extends A{

}

Exception Handling :

Exception is a pre-defined object which occurs when unexpected or abnormal thing happened during the execution of a program. To handle expected thing using some technique is known as exception handling.

Java

compile the program run the program

javac java

compile time error run time error

syntax error

or

type error

void dis() {

system.out.println()

}

Run time error

Error Exception

Error and Exception both are pre-defined classes part of lang package.

The error which generate at the run time which we can’t handle it. Ex: JVM crash, out of memory or software or hardware issue. These are Error.

The error which generate at the run time which we can handle it. Ex : divided by zero, wrong index position etc. These are Exception.

Object class

Throwable class

Error Exception

Checked Exception Un checked exception

IOException RuntimeException

FileNotFoundException ArithmeticException

SQLException NumberFormatException

InterruptedException ArrayIndexOutOfBoundsException

NullPointerException

Etc etc

All unchecked exception internally extends RuntimeException and RuntimeException class internally extetnds Exception class.

All checked exception directly or indirectly extends Exception class.

javap java.pacakgeName.className/interface

javap java.lang.Exception

javap java.io.IOException

javap java.sql.SQLException

To handle both checked exception and unchecked exception java provided 5 keywords.

1. try
2. catch
3. finally
4. throw
5. throws

Unchecked exception

Try and catch block

try{

the code which generate the exception. It may be one line code

or more than one line code.

}catch(Exception e) {

}

We can use try with single catch to handle any type of exception.

Some time if you want to execute or handle specific exception then we can use

Try with multiple catch block

try{

catch(ArithmeticException e) {

}catch(ArrayIndexOutOfBoundsException e) {

}catch(NumberFormatException e) {

}} catch(Exception e) {

}

When you are going use try with multiple catch block all sup class exception must first and super classes exception must be bottom.

In main method or user-defined method we can write

Multiple try catch block or try with multiple catch block or nested try also possible.

try{

}catch(Exception e) {

}

Set of code

try{

}catch(Exception e) {

}

try block : the code which generate the exception it may be single line or multi line we have to keep in try block.

catch block : catch block execute only if any

generate. No exception no catch block.

finally block : it is a type of block which will execute 100% sure if any exception generate or not.

Syntax

try {

}catch(Exception e) {

}finally {

}

File handling

or

Database connectivity program

try {

open the files

read or write operation

}catch(Exception e) {

}finally {

Close the resources if any exception or not.

}

Finally block we can’t use without try block.

try

catch catch catch catch finally

catch finally catch

finally

// try with catch and finally

// try {

// int a=10/0;

// System.out.println("No Exception");

// }catch (Exception e) {

// System.out.println("Catch block");

// }finally {

// System.out.println("Finally block");

// }

// System.out.println("Normal statement");

// try with finally

**try** {

**int** a=10/0;

System.***out***.println("No Exception");

}**finally** {

System.***out***.println("Finally block");

}

System.***out***.println("Normal statement");

throw keyword : throw keyword is use to generate or raise user-defined or pre-defined exception depending upon the requirements. Throw keyword we have to use inside a method.

Syntax

throw new Excxeption();

or

throw new ExceptionSubClass() it can be pre-defined or user-defined

throws keyword : throws keyword is use to throw the exception to caller method exception can be checked as well as unchecked exception.

Syntax

returnType methodName() throws Exception, ExceptionSubClass {

}

Checked exception

Checked exception we have to handle it mandatory using try- catch or throws.

Checked exception check twice compile time as well as run time.

Unchecked exception it check only at run time.

25-02-2022

Multithreading overview

Program : set of instruction to perform a specific task is known a program.

Process : time taken to execute the code or program in execution.

Processor : processor is responsible to execute the code.

Thread : small execution of a code within a process.

Thread also known as light weighted process. Process is heavy weighted. It take more time to do any task.

Java by default is thread base programming language.

Inside main method one default thread always execute.

Java provided pre-defined class ie Thread. Thread is class part of lang package.

Thead.currentThread() : currentThread is a pre-defined method part of Thread class and it is a static method. currentThread() method return type is Thread class reference.

Thread t = Thread.currentThread();

System.out.println(t) : Thread[main,5,main]

Main 🡪 name of the thread

5-🡪 priority of the thread

Priority min 1 and max 10 we can’t set less than 1 and more than 10

Max 🡪10

Norm 🡪 5

Min 🡪 1

Group of the thread : main

t.setName(“My Thread”);

t.setPriority(2);

t.setPriority(Thread.MAX\_PRIORITY); 10

System.out.println(t); Thread[“My Thread”,10,main];

Multi tasking : doing more than one task at time.

Process base

Task1 Task2 Task

Thread base

In Java we can create more than one user defined thread using

1. extends Thread
2. implements Runnable interface.

Using extends Thread class

1. create user-defined class and make that class extends Thread class.
2. create thread class reference (create the object of user-defined class which class extends Thread class).
3. with the help of reference you have to call start() method. start() method is pre-defined method part of thread class which help to make ready to run.
4. start() method internally call run() method which is part of Thread class. The run method which present inside thread class is empty method. so If you want any custom code to execute then we have to override the run method.

annotation : annotation means meta-data. Meta-data means data about data.

Java provided lot of pre-defined annotation. All annotation start with @ followed by annotation name.

@Override one of the annotation we can write on those method which are the methods are overrrided by sub class.

Using implements Runnable interface

1. create user defined class and implements Runnable interface.
2. Runnable interface contains one method ie run() method and it is a abstract method you have to override mandatory.
3. Internally Thread class implements Runnable interface and provide empty body for run method.
4. Now you have to create the reference of thread class. First you have to create the object of user defined class. The class which implements runnable interface and while creating Thread class reference pass the reference of that class which class implements Runnable interface.

Creating more than one thread using same class object.

Some if you want to execute only one thread at time.

Synchronization : synchronization is a concept which help to block or lock or allow only one thread to do all task at time. I is use to allow all resource to use by only one thread at time.

To achieve synchronization in java we can use synchronized keyword.

This keyword we can use with method or block(inside a method we can make more than one block depending upon the requirements).

Thread communication or inter thread communication

wait()

notify

notifyAll();

all these three methods part of Object class.

wait() method is use to make the thread to wait or suspend. Notify() method is use to call back waited thread or resume the suspended thread.

More than one thread must part of same object.

And method must be synchronized.

notifyAll() is use to resume more than one suspend thread

**Consumer and Producer Example using Multithreading.**

**With the help of wait(), notify() and notifyAll()**

Wait(), notify() and notifyAll() method example

**package** com;

**class** Abc **implements** Runnable{

@Override

**public** **synchronized** **void** run() {

Thread t = Thread.*currentThread*();

String name = t.getName();

**for**(**int** i=0;i<10;i++) {

System.***out***.println(name+" "+i);

**try** {

Thread.*sleep*(1000);

**if**(i==5 && name.equals("Raj")) {

wait(); // i == 5 and name is Raj we are calling wait() method belong object.

}

**if**(i==6 && name.equals("Ajay")) {

notify(); // resume suspended thread if any thread waiting

wait();

;

}

}**catch**(Exception e) {}

}

};

}

**public** **class** InterThreadCommunication {

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

Abc obj = **new** Abc();

Thread t1 = **new** Thread(obj);

Thread t2 = **new** Thread(obj);

Thread t3 = **new** Thread(obj);

t1.setName("Raj");

t2.setName("Ajay");

t3.setName("Mahesh");

t1.start(); t2.start(); t3.start();

}

}

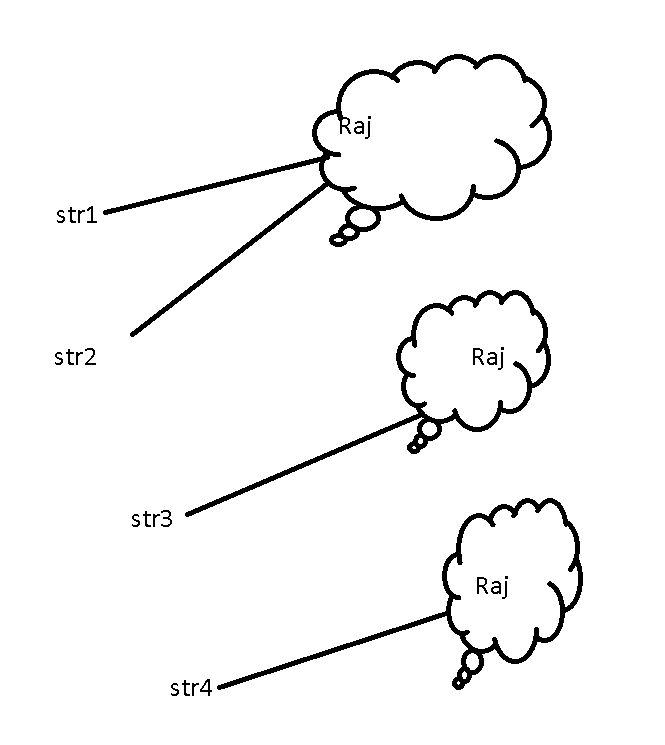
String , StringBuffer and String Builder

String is a type of reference data type or pre-defined class part of lang package.

Syntax to create string class object.

String str1 = “Welcome to Java”; literal style

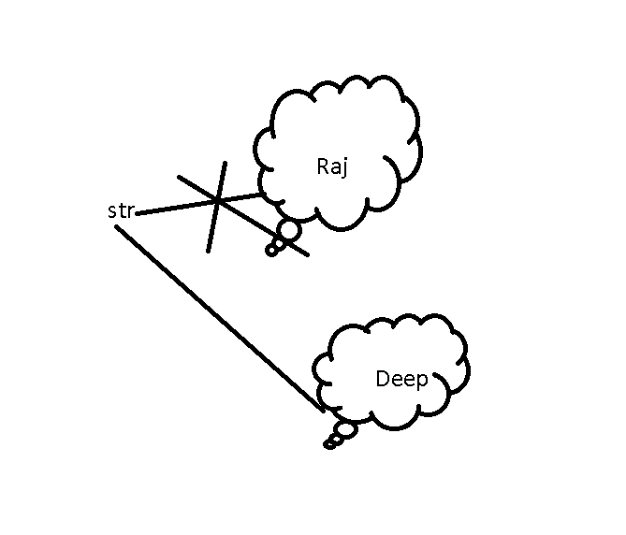
String str2 = new String(“Welcome to java”); using new keyword.



== it check value as well as reference code ie hashcode

equals() : it is use to check the only value.

String class is known as immutable class. immutable means we can’t change.



StringBuffer and StringBuilder are known as mutable string class.

StringBuffer maximum method by default synchronized. Work is safe but slow performance.

But StringBuilder method are not synchronized performance fast but not thread safe.

IO package :

Input and Output operation.

In Java we can do input output operation using two ways.

First way byte wise and second one character wise.

Stream : stream : flow of data or it is a abstraction between source and destination or target.

Stream

Byte char

Input Output Input Output

InputStream OutputSteam Reader Writer

These all four abstract classes part of io package.

javap java.io.InputStream

javap java.io.OutputStream

Byte

InputStream :

1. DataInputStream
2. FileInputStream
3. BufferedInputStream
4. ObjectInputStream

OutputStream

1. DataOutputStream
2. FileOutputStream
3. BufferedOutputStream
4. ObjectOutputStream
5. PrintStream

Char

Reader

1. InputStreamReader
2. BufferedReader
3. FileReader

Writer

1. ­OutputStreamWriter
2. BufferedWriter
3. FileWriter
4. PrintWriter

System.out.println();

System is a pre-defined class

out : it is static reference of PrintStream class.

System.out we will get the PrintStream class reference.

Println() : is it method belong to PrintStream class.

PrintStream ps = Sytem.out;

ps.println(“Welcome to Java”);

System.out is a PrintStream class reference which always refer to standard output device ie console.

Scanner sc = new System(System.in);

System is a class and in a static reference of InputStream which always refer to standard input device ie keyboards.

Byte wise

Source : DataInutStream : keyboard

Destination : PrintStream : monitor or console

**package** com;

**import** java.io.DataInputStream;

**import** java.io.PrintStream;

**public** **class** BytewiseOperation {

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

DataInputStream dis = **new** DataInputStream(System.***in***);

PrintStream ps = System.***out***;

ps.println("Enter the name");

String name = dis.~~readLine~~();

ps.println("Your name is "+name);

}

}

Source : Keyboard

Destination : File

**package** com;

**import** java.io.DataInputStream;

**import** java.io.FileOutputStream;

**public** **class** ByteWiseFileOperation {

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

DataInputStream dis = **new** DataInputStream(System.***in***);

//FileOutputStream fos = new FileOutputStream("abc.txt");// override

FileOutputStream fos = **new** FileOutputStream("abc.txt",**true**);// append the data

System.***out***.println("enter the data");

**int** ch;

**while**((ch=dis.read()) != '@') {

fos.write(ch); // automatically convert

System.***out***.print(ch+" "+(**char**)ch); // console it will not convert

}

fos.close();

}

}

Source : File

Destination : File

Buffer : Buffer is known temporary memory. Which help to improve the speed while doing input and output operation.

Using buffer we can improve 1000 time more performance.

Source file

Program target file

Character wise operation

Source : Keyboard

Destination : Console

**package** com;

**import** java.io.BufferedReader;

**import** java.io.InputStreamReader;

**public** **class** CharacterWiseOperation {

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

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

//InputStreamReader isr = new InputStreamReader(System.in);

//BufferedReader br = new BufferedReader(isr);

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

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

**int** id = Integer.*parseInt*(br.readLine()); // convet string to integer

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

String name = br.readLine();

System.***out***.println("Your id is "+id);

System.***out***.println("your name is "+name);

}

}

Source : File

Destination : File

**package** com;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**public** **class** CharacterWiseCopyFile {

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

FileReader fr = **new** FileReader("abc.txt");

BufferedReader br = **new** BufferedReader(fr);

FileWriter fw = **new** FileWriter("demoinfo.txt");

BufferedWriter bw = **new** BufferedWriter(fw);

**int** ch;

**while**((ch=br.read()) != -1) {

bw.write(ch);

}

bw.flush();

fr.close();

fw.close();

System.***out***.println("File copied...");

}

}

File class

This class is use to check the file properties. As well as we can create empty file, delete file, file present or not we can verify.

do {

1: Display all files present in current directory with ascending order.

2:

Sub option a: create new file take name of the file through keyboards.

b: delete the file ask file name through keyboards.

c: search the specific file present or not.

D: exit (sub option)

3: exit main option

switch() {

}

Regular Expression Overview

Regular Expression help to describe a pattern of text. So regex is a set of symbols and syntactic elements which are used to match pattern in text.

Valid email id, name start with some character, end with some character, contains some character, valid phone number etc.

Regular expression provide pre-defined classes ie Pattern and Matcher.

Pattern class defines a regular expression and Matcher class matches the pattern against a particular character sequence provided by pattern.

Both class doesn’t provided constructor to create the object.

Both classes part of import java.util.regex.\*;

Pattern class provide static method .compile which return Pattern class reference.

To create the Matcher class object we will take the help of Pattern reference and calling matcher which will return Matcher class reference.

First program

**package** com;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** Test1 {

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

Pattern pt =Pattern.*compile*("Java");

Matcher mt = pt.matcher("Java");

**if**(mt.matches()) {

System.***out***.println("Both are equal");

}**else** {

System.***out***.println("Not Equal");

}

}

}

Second Example

**package** com;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** Test2 {

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

String str = "Welcome to Java Training Provided by Simplilearn";

//Pattern pt = Pattern.compile("a"); // search a character

Pattern pt = Pattern.*compile*("Java"); // search a character

Matcher mt = pt.matcher(str);

**while**(mt.find()) {

//System.out.println("Yes");

System.***out***.println("Start "+mt.start());

System.***out***.println("End "+mt.end());

}

}

}

Regular Expression provided few symbols

1. \d any digit 0 to 9
2. \D not 0 to 9
3. \w any character a to z
4. \W not character a to z

Regular Expression qualifiers

? : 0 or 1

+ 1 or many

\* 0 or many

Third example

**package** com;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** Test3 {

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

//String pattern="[abc]"; // start with a or b or c

//String pattern="[^abc]"; // not start with a or b or c

//String pattern ="[a-z]"; // first character must be a to z

//String pattern = "[[a-z] || [A-Z]]"; // first character must be a to z or A to Z

//String pattern = "[a-z,A-Z,0-9]";

//String pattern = "\\d"; // [0-9] equal to 0 to 9 [0-9][0-9]

//String pattern = "\\d{10}"; // valid phone number

//String name = "9876543210";

//String pattern = "\\w{3}\\d"; //[a-z] : 3 character names must be valid

//String name ="Raj1";

//String pattern ="\\w{2,4}"; // min 2 and max 4 character

//String name = "Raju";

//String pattern = "[a-z]"; // only one character and mandatory a to z

//String pattern = "[a-z]?"; // optional 0 to 1

//String pattern = "[a-z]+"; // 1 to many

//String pattern = "[a-z]\*"; // o to many

//String name = "";

String pattern ="[a-z]+@[a-z]+\\.com";

String name ="a@b.com";

Pattern pt = Pattern.*compile*(pattern);

Matcher mt = pt.matcher(name);

**if**(mt.matches()) {

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

}**else** {

System.***out***.println("No");

}

}

}

}while();

01-03-2022

Data Structure : (Collection Framework in Java)

Data Structure is use to organized and store the data in a computer such as manner or way so we can perform any operation on that data like, add, remove, iterate , searching or sorting more efficient way(Data modification).

Algorithms

int a=10;

a=20;

array

int abc[];

store more than one value but same types.

structure

class : class adv of structure.

class Employee {

id, name,salary

}

Employee emp = new Employee();

emp.id=100;

emp.name=”Ravi”;

emp.salary = 12000;

array object

int num[]=new int[100];

Employee employees[]=new Employee[100]; employee zero object created…

employees[0]=new Employee();

employees[0].id=100;

employees[1].id=101

employees[1]=new Employee();

array is known a fixed memory size.

In employees we can store only Employee class object

Array doesn’t provide any pre-defined method to do any operation on that data. Like adding, removing, searching , iterating etc.

Collection Framework (DS)

Collection framework provided set of classes and interfaces which internally connected to each others to do specific task. All those classes by default allow to store any types of values. It means primitive values(wrapper classes object ) and user-defined class object. it provided lot of pre-defined methods which help add, remove, search, sort, iterate very easily.

Collection framework hierarchy

import java.util.\*;

javap java.util.Collection

Collection ---🡪

extends extends doesn’ t extends

Set List Queue Map

All four are interfaces. API (Application Programming interface). It may be3 class, function or interfaces.

Set : it doesn’t allow duplicate. Under the set few api store information in order, unorder and sorted. Set doesn’t provide index or it is not index base.

Set classes :

HashSet : HashSet doesn’t maintain the order.

LinkedHashSet : LinkedHashSet maintain the order. This class internally extends HashSet class.

TreeSet : TreeSet internally implements SortedSet interface. That interface extends Set interface. TreeSet display the element in sorted order by default ascending order. In TreeSet we have to store same types of values. TreeSet provided few extra method like headSet, tailSet, subset()

these classes directly or indirectly implements Set interfaces.

List : it allow duplicate values. It maintain the order. It is index based for few API.

Stack : Stack first In Last Out or Last In First Out.

ArrayList : Normal array allow to store same values But by nature ArrayList allow to store different types values. ArrayList created dynamic memory. We can add, remove any elements from ArrayList very easily which very complex in normal array. ArrayList internally implements RandomAccess interface which help to improve the performance while loading the data.

RandomAccess interface is a type of marker interface. Marker interface it contains zero methods or empty methods.

LinkedList : LinkedList is a type of data structure which use node concept to store the value.

Single linked list

Double linked list

Circular linked list

LinkedList is type of queue.

In Java By default LinkedList class internally created double linked list.

Vector

These classes directly or indirectly implements List interface.

Vector is known as legacy class. By default all methods from Vector are synchronized. If methods are synchronized it is thread safe. But slow in performance.

Queue : it is type of interface by nature if you want to do first in first out concept. Queue allow duplicate.

PriortyQueue :

Which internally implements Queue interface. It retrieve elements from queue base upon priority ie lower priority.

Map : it allow to store value in the form of key-value pairs. Key must be unique and value may be duplicate.

HashMap : it maintain the element in un order format.

LinkedHashMap: it maintain the insertion order.

TreeMap : ascending order as a key. So key must same type. Because TreeMap internally implements SortedMap interface.

Hashtable : it is a legacy class. By default all methods are synchronized. So it is thread safe but performance wise slow.

Above classes directly and indirectly implements Map interface.

Primitive Wrapper classes

byte Byte

short Short

int Integer

long Long

etc etc

Collection framework with generics

CollectionClass/Interface<Type> referenceName = new ClassName<Type>();

Type must be Wrapper classes it Integer, Float, Double, String or user-defined class object.

Collection framework with user-defined object like Employee or Customer

Retrieve the elements from collection framework using for each loop and Iterator and ListIterator

Sorting and Searching using Pre-defined class Arrays and Collections.

Custom Sorting and searching technique

02-03-2022

Retrieve the value from collection one by one using enhanced loop, iterator and ListIterator.

**package** com;

**import** java.util.ArrayList;

**import** java.util.HashSet;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.ListIterator;

**import** java.util.Set;

**public** **class** RetrieveValueFromCollections {

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

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

// we can use enhaced loop and iterator with Set API

Set<String> ss = **new** HashSet<>();

ss.add("Ravi"); ss.add("Ramesh"); ss.add("Lokesh"); ss.add("Ajay");

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

System.***out***.println("using for each loop");

**for**(String str:ss) {

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

}

System.***out***.println("Using iterator interface");

Iterator<String> li = ss.iterator();

**while**(li.hasNext()) {

String name = li.next();

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

}

// we can use enhaced loop and iterator as well as listiterator with List API

List<String> ll = **new** ArrayList<>();

ll.add("Ravi"); ll.add("Ramesh"); ll.add("Lokesh"); ll.add("Ajay");

System.***out***.println("list iterator with forward direction");

ListIterator<String> li1 = ll.listIterator();

**while**(li1.hasNext()) {

String name = li1.next();

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

}

System.***out***.println("list iterator with backward direction");

**while**(li1.hasPrevious()) {

String name= li1.previous();

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

}

}

}

Collection framework with complex object (user-defined class objects).

JavaBean class :

Class with all variable must be private and for every variable we have to provide getter and setter methods. Getter method is use to get the value and setter method is use to set value with terms and conditions.

Whenever we display any class reference using println it internally call toString() of Object class. That toString() method return output as string in the form of [packageName.className@code](mailto:packageName.className@code).

If you want to proper output we have to override toString() method and return proper string output.

**package** com;

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.List;

**import** bean.Employee;

**public** **class** EmployeeTest {

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

Employee emp = **new** Employee();

emp.setId(100);

emp.setName("Ravi");

emp.setSalary(12000);

System.***out***.println("Id is "+emp.getId());

System.***out***.println("name is "+emp.getName());

System.***out***.println("salary is "+emp.getSalary());

Employee emp1 = **new** Employee(101, "Ramesh", 14000);

System.***out***.println("Id is "+emp1.getId());

System.***out***.println("name is "+emp1.getName());

System.***out***.println("salary is "+emp1.getSalary());

Employee emp2 = **new** Employee(102, "Ajay", 16000);

System.***out***.println(emp2);

List<Employee> listOfEmp = **new** ArrayList<>();

listOfEmp.add(emp);

listOfEmp.add(emp1);

listOfEmp.add(emp2);

listOfEmp.add(**new** Employee(103, "Balaji", 16000));

System.***out***.println("Number of employees "+listOfEmp.size());

listOfEmp.remove(1); // using index position

System.***out***.println("Number of employees "+listOfEmp.size());

Iterator<Employee> li = listOfEmp.iterator();

**while**(li.hasNext()) {

Employee e = li.next();

// if(e.getId()==1000) {

// System.out.println(emp.getName());

// }

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

}

}

}

Collection framework provide two pre-defined utilities classes

Ie

Arrays

Collections

Both classes provided lot of pre-defined method ie static method which help to do sorting, searching copy, fill the data.

Arrays we can work on primitive array

Collections we can work on list api data.