Java Notes:

1.Java: Simple program to get, receive, show and store the data. (Object data)

It runs on OOPS concept.

OOPS : Object oriented programming structure;

2.standard notation or coding standards:

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

\*Pascal notation (ProjectName/ Classname)

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

Each word's first letter should be capital

Ex : GreensTechnologySolutionsLimited e:2 EmployeeDetails.

\*Camel notation: methodname/objectname

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

First word first letter should be small remaining Each word's first letter should be capital

Ex : greensTechnologySolutionsLimited ex:2 empName, empMobNo

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

3.project : Collection of required information/data. | Ex: A company (Employee details)

How to create a project: Menu-->File-->New--> Java Project --> Project Name --> finish

Project Structure:

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

thispc--->Localdriver->Folder->file

PROJECT-> PACKAGE-->CLASS-->METHODS & OBJECTS

package format : org.bike or com.employee

right click src file and create packagename in small letters but in com/org

org.cts.testing1

(package name all letter should be in small)

Class: Collection of methods and objects

Method: set of action to be performed

Main method: To execute the set of action or work and to get the output.

Object: Runtime memory allocation. so, object reference is used to call the method.

object syntax (Formulae)

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

Classname ref/objectname = new Classname();

Method call:

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

variablename/refname/object.methodname();

JDK

====

https://download.oracle.com/java/21/latest/jdk-21\_windows-x64\_bin.exe

Eclipse

======

https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2024-09/R/eclipse-committers-2024-09-R-win32-x86\_64.zip

Task

====

<https://greenstechnologys.com/selenium-training-in-chennai.html>

Datatype--> Data-->collection of information

----------

It specifies the type and size of the variable.

variable is used to store the values , name and the memory location.

Data will be differ based on its type and range of the value

Syntax

-------

Datatype variableName = variable Value;

Primitive Datatypes

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

Data Size Digits Wrapper Stores Default

Types (byte) Classes Values

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

byte 1 2 Byte 0

short 2 4 Short Used to 0

Stores Whole

int 4 9 Integer Numbers 0

long 8 19 Long 0

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

float 4 8 Float Used to 0.0

Decimal /

double 8 15 Double Fraction Numbers 0.0

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

char - - Character Single character

('A')

boolean - - Boolean true/false false

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

Non - Primitive Datatypes

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

\*String ("Greens1234@#$0.1/,")

\*default value of String is null

So, if we want to calculate the Byte range use below formula,

Range Calculation formula. n = 8 1byte = 8bits (n)

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

start end

Formulae--> -(2^(n-1)) to (2^(n-1)-1)

Byte = -(2^(8-1)) to (2^(8-1)-1)

= -(2^(7)) to (2^(7)-1)

= -128 to 127

Short = -32768 to 32767

Like this we can calculate for all datatype

for float need to mention the letter f at the last.

for long need to mention the letter l (capital L) at the last.

Polymorphism

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

poly--- many

morphism ---forms

Executing methods in more than one form

1)Method overloading(compile time polymorphism)/ static binding/static polymorphism

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

compile time polymorphism - before executing program itself can write/load business logic.

With the same method name we can create multiple method

But method name differ based upon the arguments

overloading the arguments in different forms

Diff arguments:

1. based on different data type.

2. based on different data type count/number.

3. based on different data type order

same class

same method name

diff arguments

2)method overriding (runtime polymorphism) / dynamic binding/dynamic polymorphism

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

runtime polymorphism - After execution only we come to know the business logic

If u want to change the business logic /re write/override the business logic

in the class means we are using the overriding method.

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

Same method

Same argument

diff class

static binding - doesn't change with respect to time it will be same logic

dynamic binding - logic is different

Java Oops Very Important inheritance, polymorphism,abstraction, Encapsulation

1. Inheritance

we can access one class property from another class (using extends keyword)

memory wastage is low.

2. Type of inheritance

Single inheritance:

Combinations of one parent class and one child class.

One child class is directly get support from one parent class by using extends.

Multilevel inheritance:

Child class is accessing more than one parent class in a tree level structure

Hierarchical inheritance

Combination of one parent and more than one child

Multiple inheritance:

Child class is accessing more than one parent class Parallelly.

Java does not support Multiple Inheritance:

- Syntax error

Priority Probles

Hybrid Inheritance

Combination of Single and Multiple inheritance

cannot support in java.

Java does not support Hybrid Inheritance:

1. Abstraction

2. Types of abstraction

1.Abstraction

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

hiding the implementation details or business logic details.

business logic - execution step/ output.

inside the curly braces all business classes

2.Types of abstraction:

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

partial abstraction (abstract Class)

Fully abstraction(interface)

partial abstraction(abstract class):

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

use a keyword abstract in both class and method

contain both abstract and non abstract methods

we can't create object in abstract class

We can create object only in child class ,it contains keyword extends

fully abstraction(Interface)

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

we call only interface. it has fully abstract methods only.

use a keyword interface instead of class

contain only the abstract methods

In this interface public abstract is default one

we can't create object

We can create object only in child class , it contains keywords implements

Access specifier / modifier

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

it specifies the level of access

private--class level access specifier ( It will support only within the class) red square

public-- we can access from anywhere samepackage(extends,object)+different

package(extends,object) ( green Circle)

default--package level access specifier ( blue Triangle)

it will support only within the package using both object as well as extends(keyword)

Protected--samepackage we can access using both (extends,object) (yellow Diamond)

different package we can access using (extends)

Non -Access modifiers

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

Abstract

static

final

Abstract:

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

1.class level

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

If we declare class as abstract, we can't create object

2.Method level

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

If we declare method as abstract, we can't write business logic

3.Variable level

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

we can't able to declare variable as abstract(compile time error will occur)

Static:

=========

1.class level

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

we can't able to declare class as static

2.Method level

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

same class.....>If we declare method as static no need of object creation we can

directly call by method name

diff class(without extends)....>ClassName.methodname()

diff class(with extends)........directly call by method name

3.Variable level

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

sameclass>If we declare variable as static no need of object creation.

diff class(without extends)....>ClassName.variableName

diff class(with extends)...>directly call by variable name

Final:

=======

1.class level

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

If we declare class as final,we can't Inherit

2.Method level

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

If we declare method as final,we can't override

3.Variable level

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

If we declare variable as final,we can't change the value.

1 ------>Quo

------

2|3

2

======

1 --> Remainder => (%)-modulae

=====

Control Statement

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

statement that determines whether the other statements will be executed or not.

It controls the flow of a program.

Conditional statement

Looping statement

Jumping Statement

Conditional

===========

if

if else

else if ladder

switch

Looping

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

for ->

nested for

while

do while

conditional operator a=10 ,b=10 ,c=10

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

> -> greater than

< -> lesser than.

= -> equals /assign

== -> equal to/compare to

!= -> not equal to 100 != 100 - false

>= -> greater than or equal to

<= -> lesser than or equal to

Ex:

int a =10, b=20;

if

===

if (condition) {

business logic;

}

if else

========

if (condition) {

business logic;

}

else{

business logic;

}

Logical operators

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

&& - AND

|| - OR

a = 10 , b =20

20==0-f & 10>=10-t

&& ---> it will check all the given conditions are true, give the result as true.

If any one condition is fails, the result will be false.

true && true --> true

false && false ---> false

false && true ---> false

true && false ---> false

|| --> if any one condition is true, the result will be true.

If all the conditions are false, the result will be false.

T || F

true || true --> True

false || false ---> False

false || true ---> True

true || false ---> True

Bitwise Operators

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

& - AND

| - OR

Else if

--------

if (condition) {

business logic;

}

else if(condition){

business logic;

}

else if (condition){

business logic;

}

else{

business logic;

}

While

1 2 3 4

I=0 I<5 I I++

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

0 0<5-t 0 0+1=1

1 1<5-t 1 1+1=2

2 2<5-t 2 2+1=3

3 3<5-t 3 3+1=4

4 4<5-t 4 4+1=5

5 5<5-f===> loop terminated

dowhile

1 2 3 4

a=0 a a++ a<5

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

0 0 0+1=1 1<5-t

1 1 1+1=2 2<5-t

2 2 2+1=3 3<5-t

3 3 3+1=4 4<5-t

4 4 4+1=5 5<5-f --> Loop terminated

switch

=======

switch (choice){

case 1/choice:

bus logic;

break;

case 2:

bus logic;

break;

case 3:

bus logic;

break;

default:

bus logic;

}

1 2 3 4

I=0 I I++ I<5

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

0 0 0+1=1 1<5-t

1 1 1+1=2 2<5-t

2 2 2+1=3 3<5-t

3 3 3+1=4 4<5-t

4 4 4+1=5 5<5-f ==> loop terminated

Normal For loop

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

1 2 4

for (initialization;condition;increment/decrement){

3

//Business Logic;

}

initialization -->int a =10

condition ---> a<10-> 10<200 -> t

increment or decrement --> a++==>10+1=11

a-- => 10-1=9

I =0

I++ -->0+1-->1 (Post Increment )

++I--->1+0=1 (Pre Increment )

Nested For loop

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

1 2 8

for(init;Condi;inc/dec){

3 4 6

for(init;Condi;inc/dec){

5

buisnesslogic

}

7

b.l

}

while (Entry level Check loop)

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

intialization;

while(Condition)

{

bus logic;

inc/dec;

}

do while (Exit Check Loop)

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

intialization;

do{

bus logic;

inc/dec;

}

while (condition);

Jumping Statement:

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

break --> for , switch

continue

System.exist

Break; ---> it terminate the iteration

continue; ---> it skips the current iteration

System.exit---> it terminate the code

1 2 3 4 5 6

I=0 I<5 I==3 break I I++

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

0 0<5-t 0==3-f 0 0+1=1

1 1<5-t 1==3-f 1 1+1=2

2 2<5-t 2==3-f 2 2+1=3

3 3<5-t 3==3---> br ==> loop terminated

End

1 2 3 4 5 6

I=0 I<5 I==3 conti I I++

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

0 0<5-t 0==3-f 0 0+1=1

1 1<5-t 1==3-f 1 1+1=2

2 2<5-t 2==3-f 2 2+1=3

3 3<5-t 3==3-t conti 3+1=4

4 4<5-T 4==3-F 4 4+1=5

5 5<5-F===>loop terminated

End

1 2 3 4 5 6

I=0 I<5 I==3 exit I I++

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

0 0<5-t 0==3-f 0 0+1=1

1 1<5-t 1==3-f 1 1+1=2

2 2<5-t 2==3-f 2 2+1=3

3 3<5-t 3==3-t exit-----> it terminate the whole system code

1.String

-------- "qwerty"

\* Collections of character or word enclosed with double quotes is called as String.

\* String is a predefined class.

\* It is presented in java.lang package.

\* It is based on index/Position. ---> 0 - to n-1

INdex 0123

String s ="Java"; n = 4-1 -->3

2.String Methods

String Methods:

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

length();

toUpperCase();

toLowerCase();

indexOf('char');

lastIndexOf('char');

charAt(index);

equals(stringVariable); - Case sensitive

equalsIgnoreCase(stringVariable);

contains("String") - Case sensitive

replace("String1","String2")

startsWith("Welcome")

endsWith("class")

isEmpty()

trim()

substring(index);

""

split(regex); ---> Array

String s = "ram"

Array --> []

=====

We can store multiple values of similar datatypes in a single variable

Similar datatypes

Index based

Index starts from 0 to n-1.

It allow duplicates

1D ARRAY

1 2 3 4

I I<4 I I++

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

0 0<4-t a[0]=200 0+1=1

1 1<4-t a[1]=600 1+1=2

2 2<4-t a[2]=400 2+1=3

3 3<4-t a[3]=500 3+1=4

4 4<4-f ===> Loop terminated

2D ARRAY

1 2 3 4 5 6 7

I=0 I<3 j=0 j<2 a[I][j] j++ I++

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

0 0<3-t 0 0<2-t a[0][0]=2000 0+1=1

1 1<2-t a[0][1]=4000 1+1=2

2 2<2-f--------------------------->0+1=1

1 1<3-t 0 0<2-t a[1][0]=5000 0+1=1

1 1<2-t a[1][1]=8000 1+1=2

2 2<2-f--------------------------->1+1=2

2 2<3-t 0 0<2-t a[2][0]=6000 0+1=1

1 1<2-t a[2][1]=7000 1+1=2

2 2<2-f--------------------------->2+1=3

3 3<3-f====> Loop terminated

n=4 0,1,2,3

[] - array

Array syntax

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

DataType[] variable = new Datatype[length]; // One

Datatype var [] = new datatype[length]; // two

int a [] = new int[4]

a[0] = 10;

a[1] = 20;

a[2] = 30;

a[3] = 40;

Enhanced for loop/ for each

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

old var

for(DataType NewVariablename : StoredVariable){

business logic;

}

for(String x:s)

{

sysout(x);

}

2D Array

========== row column

DataType variable [ ] [ ] = new Datatype [length] [length];

DisAdvantages of Array

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

We can store only the similar datatypes

once we fixed the length we cant modified

memory wastage is high

Compile time (array)

normal enhanced forloop (for each)

index based value based

condition set no condition

initiate from any no intialization

iteration

Chances of error less chance of error

Exception will occur there's no exception

default value:

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

int 0

String null

char ''

float 0.0

boolean false