***CORE JAVA NOTES BY VIJAY SIR***

----------------------Introduction--------------------------------

- Java is a object-oriented, general-purpose, high-level programming language.

developed by James Gosling and his team at Sun Microsystems and released in 1995.

- Oracle Corporation aquired SunMicrosystems in January 27, 2010.

- SunMicrosystem was conducting exams based on Java SCJP (SunCertifiedJavaProffessional)

Now Oracle conducting those exams in the name OCJP (OracleCertifiedJavaProffessional)

- The Initial name of Java is Oak. Later they changed it to Java (NameofTheCoffeeFlavour).

- you can use Java programming for developing mobile applications, enterprise web application, embedded systems Ex: Elevator(lift), WashingMachines, CarMusicPlayers, DeskTopApplications,BigDataTechnologies(DataProcessing) - ApacheHadoop and ApacheSpark(BigDataFrameworks), ScientificApplications, GameDevelopment.

- Java Programming laguage is also called as Java platform.

- Java is a high level programming language.

- Open Sourcing: In 2006, Sun Microsystems released the source code of Java under the GNU General Public License (GPL) as OpenJDK, allowing developers to view, modify, and contribute to its development.

High Level programming language

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

-Programmer friendly

-easy to understand

Ex: C++, Java, Python, JavaScript, Ruby

Low Level Programming Language

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

- Machine friendly

- difficult to understand

Note: Low level programming language provides a close representation of the hardware and allows direct control over the system's resources.

Ex: Assembly Language, COBOL, FORTRAN

- Machine can only understand 0s and 1s

- Since computers can't able to understand high level programming languages, compilers and interpreters are used to for translation, translating from high level to machine level code.

Translators

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

Converting the code from high level language to machine laguage.

Translators two types

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

1. Compilers

2. Interpreters

Compilers VS Interpreters

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

1. Compiled Programming Languages -> C, C++, Java

2. Interpreted Programming Languge -> Python

3. In case of compilers: compiler takes the entire code at once and it will generate the machine level code.

4. In case of Interpreter: interpreter converts the high level laguage to the machine level code line by line. NOT AT ONCE

JAVA IS BOTH COMPILED AND INTERPRETED -> THAT WILL BE DECIDED BY THE JVM(Java Virtual Machine)

- JDK (software development kit) that includes everything needed to develop, compile and debug java applications, includes various development libraries.

-JRE (SoftwarePackage) provides necessary runtime environment for executing Java applications.it includes JVM as well.

-JVM - Provides an execution environment for JavaAppliations.

Summarry:: The JDK is used by developers for Java Application Development. The JRE is required for end-users to run JavaApplications, and the JVM is responsible for executing Java bytecode and provides plarform independence.

- Java is object-oriented Programming Language NOT object-based

- Python is object based, Interpreted. Everything in case of python is object only.

-Java is Developed By James Gowsling in the year of 1995 while he was working at Sun Microsystems.

-The initial name of Java is Oak, Later they changed it to "Java" Java means, a flavor of coffee.

-James Gowsling is also called as Father of Java Programming Language.

Feature of Java programming Language

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

1. Object Oriented

2. Platform Indepent

3. Secured

4. Auto Memory Management

5. Exception Handling Mechanism

6. Multi-Threaded Programming Language

7. High Performance

8. Portable

1. Object Oriented

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

An object is a real world entity, that is having properties and behaviours.

Ex: Dog -> Object

properties -> name, age, gender, weight, breed, color

behaviours -> run, eat, sleep, bark, bite, chase a cat

This type of developement is not possible in case of C programming laguage(it is procedural language)

2. Platform Independent

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

Java is Write Once and Run Anywhere(WORA) that means achitechture neutral.

This is possible because of Byte Code.

Initially Java Compiler will compile the java program and produces a ".class" file.

Inside that .class file byte code will be available. that byte code (that .class file) is platform independent.

There is an internal mechanism called JVM (Java Virtual Machine) that is available inside the JDK(Java Development Kit) software itself.

The JVM will be converting that Byte code to the corresponding machine level code.

The byte code that we have developed by using any operating system, can ran on any other operating system. There is no need of changing anything to source code. THIS IS NOT POSSIBLE IN CASE OF C

-JDK itself is not platform independent but the byte code is platform independent.

3. Secured

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

- We can avoid security issue in case of Java, bcz in java there is no pointers concept.(in case of pointers concept, if we miss manage the memory by using pointers then there will be a securiy issue, hackers might breach the data and also pointers concept is very confusing) C PROGRAMMING LAGUAGE USES POINTERS BUT NOT JAVA

4. Auto Memory Management

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

- In case of Java There is a auto-memory management by using Garbage collector.

5. Exception Handling Mechanism

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

- To maintain the normal flow of execution we have Exception Handling

THIS IS NOT AVAILABLE IN C

6. Multi-Threaded Programming Language

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

- To increase the efficiency of the CPU and parallel execution.

THIS IS NOT AVAILABLE IN C

7. High Performance

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

- By using JIT(Just In-Time Compiler)

8. Portable

-----------

- Refere to Achitecture Neutrality (byte can be easily run on any platform) we can take the byte code and run it on any plaform.

- Most of the companies will be choosing Java to build their business applications.

The Lastest of Java is 20

- The Latest edition of Java might not be supported by all the platforms(AWS, Azure, Google Cloud)

- The industry standard and which is supported by all the platforms is Java 8

- That is why we will download and install and work with Java 8

- We will be also learning about all the new features and new concepts that has been introduced in the latest editions of java.

Java SE - Standard Edition

Java SE Development Kit 8u371

u == update

JDK software is available for

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

Linux

macOS

Solaris

Windows

According to the system architecture you need to select the installer

it can \*86 (32 bit) or \*64 (64 bit) operating system, or processor

check - right click on this PC -> under system type you will find that.

\*86 is compatible with only 32 bit system architecture if the system architecture is 32 bit only.

if your system is 64 bit then both \*86 and \*64 is compatible.

select this file jdk-8u371-windows-x64.exe

Along with JDK software - > JRE and also JVM

JDK - Development purpose

JRE - Production purpose

JVM - Provides an environment to run(execute) byte code

(Inside the JRE JVM will be there)

- Currently Java No.1 Development Platform(Business and Banking Applications)

- So many devices are running by using Java. Ex: Car Music Systems, lifts, Washing Machines(as embedded softwares), Android TVs, Smart Phones etc..

to find out the java version that is installed in our system trigger

java -version

software version analysis

major minor updates or patches

1 . 8 . 0\_371

After the installation of JDK the first thing we need to do is setting the path environment varible.(or updating the path environment variable)

D>Developmentsoftwares>JDK8>bin || copy this folder location from the address bar

Right click on this pc -> properties -> AdvancedSystemSettings -> EnvironmentVariable - UserLevel -> select the path variable if already existing

and click on edit -> click on new -> paste the copied path of bin folder -> click on ok and in all the remaining windows.

if the path variable is created yet -> under user variable section -> click on new ->variable name 'path' uppercase of lowercase or mixed case but dont give any spaces in the beginning or in the middle or at end.

->variable value -> paste the copied bin folder location

close the previously opened command prompt(before the path setting or path updation). open a new command prompt and trigger javac

Need of setting the path

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

in order to specify the commands location(java, javac) to the command window we need to set the path.

Installing EditPlus for the development and Learning Java Concepts through programs

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

1. Go to https://www.editplus.com/ -> Select Download from top menu

you will get these two options

EditPlus 5.7 Evaluation Version

● Download EditPlus 5.7 (64-bit) == if your system is 64 bit architecture, then download this.

● More options... ==> Select this if your system is 32 bit architecture. then download.

2. Before Installing, create a folder called EditPlus inside the DevelopmentSoftwares folder.

(in windows to create a new folder shortcut cntl + shift + n)

while developing the folder name or file name follow the CamelCase Ex:(EditPlus). the first letter of each word will be in the upper-case without giving any space.

Note:: All the develoment softwares we are installing in the D drive(any drive other than C drive) because we need to lower the OS load, because the OS drive is already having OS files. its not recommended to give more load to that drive.

3. Start the installation by double-clicking on editplus.exe file(downloaded).

(Note: '.exe' is an executable file in the windows system)

4. Installation Windows

1. Accept the Licence Agreement.

2. click on yes for the windows notification(for making the changes to the system).

3. Next window: specify directories where you want to install. in this window just leave all the selected check boxes as it is, then for the program directory, already the installer is pointing default loacation 'C:\Program Files\EditPlus' instead going with this default location select the folder location that we already created 'D:\Developmentsoftwares\EditPlus' for this select the three '...' which is visible after the path indicator.

- Expand This PC -> Expand D drive -> DevelopmentSoftwares -> Then Select EditPlus folder -> then click on OK

- installation program directory should be looking like this 'D:\Developmentsoftwares\EditPlus'

- click on 'Start Copy'

4. In the Next window you will get a success message. Just click on OK.

5. Go to desktop(cntrl + D(all the opened windows will be minimized, desktop will be visible)).

6. Double click on EditPlus application shortcut

7. When you open the EditPlus for the first time after installation you will get License AgreementWindow Again, you just need to click on Yes to accept.

8. You will get Set Directories window - this one is related to the Internal EditPlus files storage.

you just need to click on OK

9. you will get a small window saying that 'Setup will overwrite the existing syntax files' Continue?

Just click on Yes.

10. EditPlus is not 100% free but we can able to use as free for the life time with minor task.

11. you will get Enter Registration Code window -> Trial -> you will get a small window called evaluation version. you just need to drag that window to the bottom of the screen, till it is not visible any more. Then maximize the main window. close the side bar(file Explorer) by dragging to the left side.

12. We need to make some configuration to the EditPlus.

click on Tools -> Preferences -> select General -> Fonts -> FontFamily = consolas, FontStyle = Bold, FontSize = 22

on the left hand-side categories, Select File Uncheck the option 'create the backup file while saving'(To avoid EditPlus creating backup files for all the files). Then click on Apply and click on OK.(To save the changes)

Developing our first java program using simple notepad

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

1. On the desktop create a folder called JavaDemo.

2. Open the windows NotePad(click on windows icon(bottom left corner)start typing notepad and pin to task bar. and click on the notepad icon on the task bar)

3. To increase the font size in the notepad press cntl + mouse(scrollUp).

4. start developing the HelloWorld Java Program

class HelloWorld

{

public static void main(String[] args)

{

System.out.println("Hello World!");

}

}

5. copy the class name 'HelloWorld' -> File -> Save As -> Desktop > JavaDemo -> Enter the file name

'Just paste class name whichever you copied'-> compulsary include the '.java' extension.

Note: For the Java files .java is the extension(Java files will be identified with the .java file extension). -> click on save.

6. Open the command prompt -> make the prompt to point to the 'Desktop > JavaDemo' Folder

Note: Folder == Directory by using command cd Desktop/JavaDemo or if it is not working use the command cd OneDrive/Desktop/JavaDemo.

Note: cd (change directory)

7. To compile the java program we need to trigger the command called javac, this will be invoking the java compiler to check for the syntax correctness of the java file.

Note: javac (java compiler)

Ex: javac JavaFileName.java == javac HelloWorld.java

8. If the program is syntactically proper, then a '.class' file will be generated in the same folder (JavaDemo). Otherwise we will get syntax error. we need to resolve that error, save the file and again we need to trigger 'javac JavaFileName.java'.

9. check the JavaDemo folder for '.class' file. select that .class file and Open that file by right clicking and select EditPlus.

Note: if you cant able to see the extenstions of the files then click on view -> select the check box of file name extensions.

We cant able to read the contents of the .class file. this is exactly what we call the byte code.

inside the .class byte code will be there. we need to execute the .class file.

10. For Executing the .class file -> use the java command 'java name\_of\_the\_class\_file'

Ex: java HelloWorld

11. From the same location 'C:\Users\vijay\Desktop\JavaDemo>' trigger 'java HelloWorld'

you will be getting 'Hello World!' as the output

Note: you should not include any .class or .java extension while executing the class file.

12. Once you trigger java HelloWorld -> JVM(Java Virtual Machine) will be invoking the main method of HelloWorld class and executing the print statement which is producting Hello World! as the output.

Some More Configuration to the EditPlus

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

1. Problem - if we are having more content in the same line then we need to scroll to the right-hand side.

Sol - To avoid this problem we need to turn on word-wrap

steps - click on Document -> Select word-wrap

2. To get more Editor space turn off -> Ruler, that is visible on top of the Editor space.

steps: click on view -> uncheck 'Ruler'

3. To get the FullScreenWindow -> just press F11(toggle for interchange)

Note: if you making any changes to the java file(even minor changes like adding or removing the space) then compulsory you need to save that file and again you need to compile that file before executing with java command.

Windows NotePad VS EditPlus

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

1. While saving the files Auto Extenstion will not be applied in case of notepad.

2. There is no syntax highlighting feature(color highlighting) in notepad so we may not be easily able to recognize keywords, identifiers, inbuilt classes.

3. There are no extra fetures in notepad which are available in the EditPlus like LineNumber, Ruler, Easy navigation between files using inbuilt package explorer and so many other tools that are very much required for the efficient and faster development of programs.

Task --> Develop 5 Programs with the class and file name as

1. HelloWorld1 -> output should be Hello World 1!

2. HelloWorld2 -> output should be Hello World 2!

3. HelloWorld3 -> output should be Hello World 3!

4. HelloWorld4 -> output should be Hello World 4!

Note: No need of opening and closing the command prompts again and again for each and every program you can just use the same command prompt for all the programs.

Standard Way of Developing Java Programs

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

1. Create a folder in any of the drive other than OS drive, Name it as JavaCourse.

2. Inside the JavaCourse create another folder LanguageFundamentals -> application1 -> src & classes

3. src stands for source(folder)

4. We are creating a seperate folders. for keeping .java files in the src folder and to keep generated .class files in the classes folder.

Reason: We are seperating '.class' files from '.java' files.

'.java' files in a seperate folder called 'src'.

'.class' files in a seperate folder called 'classes'.

This is the standard way of developing our java programs.

First Approach

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

C:\Users\vijay>G:

G:\>cd LaraBatches

G:\LaraBatches>cd JavaCourseJune

G:\LaraBatches\JavaCourseJune>cd LanguageFundamentals

G:\LaraBatches\JavaCourseJune\LanguageFundamentals>cd application1

G:\LaraBatches\JavaCourseJune\LanguageFundamentals\application1>cd src

G:\LaraBatches\JavaCourseJune\LanguageFundamentals\application1\src>

- Open the EditPlus File -> New -> Java

- you will be gettting the default code (recommended to delete(ctrl + A and backspace) and type again).

- File -> Save As -> Save it in the 'G:\LaraBatches\JavaCourseJune\LanguageFundamentals \application1\src>' folder with FileName as A (.java extension is optional here bcz already editplus selecting Save as type to .java)

-compile the file using javac FileName.java

-if our program syntactically proper then .class file will be generated. In this case .class file is generated in the same folder(src).

- To get the FileNameExtenstions in the windows FileExplorer Just View -> Select the checkbox for

FileNameExtenstions.

Type: JAVA File == JAVA Source File (Java Source Code File)

CLASS File == Compiled version of Java Source File (.class file) Which contains byte code.

we cant able to read it, so we are chieving security. This file will be processed by the JVM.

This file is platform independent(specifically the content of this file).

- Both the files .java and .class are present in the same src folder itself, which is not a standard way develpment.

Second Approach

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

C:\Users\vijay>cd G:\LaraBatches\JavaCourseJune\LanguageFundamentals\application1\src

C:\Users\vijay>G:

G:\LaraBatches\JavaCourseJune\LanguageFundamentals\application1\src>

Third Approach

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

1. select and copy the address of src folder location from the windows explorer.

Ex: G:\LaraBatches\JavaCourseJune\LanguageFundamentals\application1\src

2. open a new command prompt type cd and paste the content you just copied.

Ex: cd G:\LaraBatches\JavaCourseJune\LanguageFundamentals\application1\src

3. if you are still not that drive and folder, just type the name of the drive and colon Ex: G:

Fourth Approach

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

1. Go to the src folder location in the windows explorer and clear the address bar content.

2. Just type cmd in the upper case or lower case and press enter.

In this location command prompt will be opening.

3. to increase the font size of the command prompt just press cntrl + mouse(scroll up)

Note: - create a new java file in editplus press Alt + F and right arrow and select java

- To save file cntrl + S

Note - to clear the command prompt window just trigger the command 'cls'

To seperate the .class files from the .java files while compiling

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

1. javac -d ../classes B.java

javac == java compiler

-d == option we are using along with the javac command.

this option is for interacting with the file system.

../classes ==> currently we are in the src folder. to go back to the previous folder application1 folder, and select classes folder. In this location we need to keep the generated .class file.

B.java ==> the fileName we wish to compile.

\*\*Note: you need to trigger this command from the src location only\*\*

Runnnig the .class file

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

1st Approach

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

src>cd ../classes

application1\classes> java B

(java classFileName)

To go back to the src

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

cd ../src

2nd and best Approach (easier)

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

java -cp ../classes B

java ==> to invoke the JVM

-cp ==> option 'classpath' -> it will be seaching for .class files in the directory.

../classes B ==> currently we are in the src folder. to go back to the previous folder application1 folder, and select classes folder. there you will find class file, that file we need to execute.

Task ==> Create 4 java files and compile and execute using the steps showed above

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

1. C.java

2. D.java

3. E.java

4. F.java

Note: To rename the file or folder press F2 in windows.

public class demostration

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

1. public is a keyword in java.

Note: - keywords are having speacial meaning to the java language itself.

- Keywords are also called as reserved words.

There are about 50 keyword are available in Java 8

1. abstract

2. assert

3. boolean

4. break

5. byte

6. case

7. catch

8. char

9. class

10. const(NOT USED)

11. continue

12. default

13. do

14. double

15. else

16. enum

17. extends

18. final

19. finally

20. float

21. for

22. goto(NOT USED)

23. if

24. implements

25. import

26. instanceof

27. int

28. interface

29. long

30. native

31. new

32. package

33. private

34. protected

35. public

36. return

37. short

38. static

39. strictfp

40. super

41. switch

42. synchronized

43. this

44. throw

45. throws

46. transient

47. try

48. void

49. volatile

50. while

\*\*Note: There are 3 reserved literals -> 1.null 2. true 3. false \*\*

\*\*Note: each keyword is having a specific functionality\*\*

\*\*Note: 'goto' and 'const' are NOT USED(deprecated) even though these two keywords are present in the Java 8, we are not using those keywords. due to the guidlines provided by the Java Developers. because of language improvements reasons\*\*

- while compiling a java file you should compile it with java file name itself, NOT the class name which is present in the java file.

- In a Java file if you are declaring a class with public keyword then that public class name should be the file name, otherwise you will get error.

- If we are not declaring a class as public then file name can be anything.

while compiling we need to compile with filename but while executing we need to execute with the class name which generated inside the classes folder.

- The initial task of the JVM is to invoke the main method of that class.

- Main method is considered as starting point of a program(execution starts from the main method)\*\*Generally

Note: before the main method we can also execute some of the members which are static members.

class B

{

}

-defined

-concrete

-implemented

-complete body

Note: the above program is syntactically proper.

Note: public keyword is used as an access specifier.

- if the class is not having a main method we cant able to run that class.

- in one java file we can able to develop any no. of classes.

- if none of the classes are declared with public keyword then file name can be anything.

- if at least one class is declared as 'public' then that class name must be the file name.

- in one java file maximum 0 or 1 public class is allowed not more than that.

for the folder seperation in windows '\' will be used

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

G:\LaraBatches\JavaCourseJune\1.LanguageFundamentals\2.public\_class\classes

in the windows only but in the command prompt while the user specifying the path should be be using '/' thats only recognized in the command prompt for navigation by the user

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

src>java -cp ../classes T

Note: even some of the cross platform softwares will supporting both the characters '/''\'

Note: In the browser address bar we will be using '/' for seperation

Ex: https://www.google.com/chrome/download

Note: Linux will be using '/' for directory seperation

NOTE: Learn typing ==> https://www.speedcoder.net/

you should maintain at least 60WPM

IDENTIFIERS - In Java

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

Defintion: - To identify something by using a name in java programming we can use identifiers.

- An identifier is a name given to a class, method, variable and other programming elements to uniquely identify them within their context.

Navigation using command prompt

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

G:\LaraBatches\JavaCourseJune\1.LanguageFundamentals\2.public\_class\src>cd ..

G:\LaraBatches\JavaCourseJune\1.LanguageFundamentals\2.public\_class>cd ..

G:\LaraBatches\JavaCourseJune\1.LanguageFundamentals>cd 3.identifiers

G:\LaraBatches\JavaCourseJune\1.LanguageFundamentals\3.identifiers>cd src

G:\LaraBatches\JavaCourseJune\1.LanguageFundamentals\3.identifiers\src>cd ..

G:\LaraBatches\JavaCourseJune\1.LanguageFundamentals\3.identifiers>cd ..

G:\LaraBatches\JavaCourseJune\1.LanguageFundamentals>cd 2.public\_class

G:\LaraBatches\JavaCourseJune\1.LanguageFundamentals\2.public\_class>cd src

G:\LaraBatches\JavaCourseJune\1.LanguageFundamentals\2.public\_class\src>cd ../../3.identifiers/src

G:\LaraBatches\JavaCourseJune\1.LanguageFundamentals\3.identifiers\src>cd ../.../2.public\_class/src

The system cannot find the path specified.

G:\LaraBatches\JavaCourseJune\1.LanguageFundamentals\3.identifiers\src>cd ../../2.public\_class/src

G:\LaraBatches\JavaCourseJune\1.LanguageFundamentals\2.public\_class\src>cd ../../3.identifiers/src

G:\LaraBatches\JavaCourseJune\1.LanguageFundamentals\3.identifiers\src>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Def: A name Java programming to identify something by using an unique name is called identifier Ex:class name, method name, variable name

Rules and possibilities for developing Identifiers

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. STANDARD: Follow the camel case while developing class name.Ex: HelloWorld

Note: class name is also an identifier.

2. Identifier name must not start with a digit(number). Ex:1HelloWorld ==> not possible

3. Identifier name can have a digit in the middle.Ex: Hello2World ==> possible

4. Identifier name can have a digit at the last in the identifier name. Ex: HelloWorld3 ==> possible.

5. In the identifiers except $ and \_ there are no other special characters are allowed.

Ex: \*HelloWorld ==> not possible

\_\_\_Hello\_\_\_ ==>possible

$$Hello$$ ==>possible

\_\_Hello$$ ==> possible

6. Alphabets (a-z & A-Z)

7. digits (0-9)

8. keywords cant be used as identifiers.

9. we can develop the identifier name with any length.But keep it simple and meaningful.

10. we cant have the space in the middle of the identifier.

11. we can able to use inbuilt class names and interfaces names as identifiers.

but it is not recommended because it leads to the confusion.

Note: its possible to create a java file without any content in that file. Compilation will be success.(developing empty java file is possible), but you dont get the class file.

print VS println

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. In case of 'println' cursor will be coming to the "next line". i,e after printing the content of the print statment, because of the 'ln' cursor comes to the next line.

2. println is method

Note: System is class, out is an object, println is a method.

3. In case of 'print' cursor will not comes to the "next line". i,e after printing the content of the print statment, because of 'just print and without ln' cursor will be staying in the same location.

Note: print is also a method

Note: In General we call print and println statements as 'print statment' but there will a different method for 'next line' and 'without next line'

Note: 'next line' is also called as 'new line'

Note: print and println both will be interacting with the "standard output stream"

There are maily three streams in Java

1. Standard Input Stream

2. Standard Output Stream

3. Standard Error Stream

- In Java ';' indicates that end of a statement.

Note: if you are not ending a statement with a semicolon then you will be getting CTE.

- CTE(Compile Time Error or Compilation Error)

-Note: console == command prompt == terminal

\*\*\*\*\*\*\*\*\*\*\*Literals in Java\*\*\*\*\*\*\*\*\*\*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

- Literal is a values that can be assigned to a variable.

- Literals represent fixed values that cannot be changed during the exection of the program.

Java Supports several types of literals

1. Integer Literal - A number without any decimal points(number of -ve or +ve)

2. Floating Literal - A number which has a decimal point

3. Character Literal - A single character within a pair of single quots

4. Boolean Literal - true and false

5. String Literal - multiple characters within a pair of double quots

6. Null Literal - null is also a literal, that can be assigned to a reference variable.

Basic MathOperators

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

- we can perform mathematical operations inside the SOP

syntax: operand operator operand

Ex: 10 + 10

20 - 10

- in case of '/' you will get quotient value

- in case of '%' you will get remainder

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Datatypes and Variables\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Datatype is nothing but a type of data

In Java we have 2 types

1. primitive datatypes

2. non-primitive datatype

primitive datatypes ==> 8

1. byte

2. short

3. int

4. long

5. float

6. double

7. boolean

8. character

non-primitive datatypes ==> reference type, user type

\_\_\_\_\_\_\_\_variables\_\_\_\_\_\_\_\_

- In order to store the varying data we need a variable

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_In case of primitive datatypes\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SYNTAX: datatype variableName assignmentOperator literal;

Ex: int i = 10;

==> int i = 10;

Note: = is the assignment operator. left side assignment.

Note: variable is also an identifier

int i; ==> variable declaration

i = 10; ==> initialization

System.out.println(i); ==> usage

\_\_\_\_JavaConvention\_\_\_\_

class name ==> always starts with a uppercase letter and follows camel case.

Ex: HelloWorld

variable and method name ==> always starts with a lowercase letter and follows camel case.

Ex: thisIsTheVariableIdentifierExample

Ex: thisIsTheMethodNameExample()

Note: Java compiler ignores the white spaces.

we can use the whitespaces for better readiability.

- By default the decimal values are considered as double type.

- By default in java decimal values are considered as double types only not float type.

- to supply the decimal values for float datatype just include either 'f' or 'F' after the decimal value. then compiler will be considering that value as float value.

\_\_\_\_\_\_\_\_\_\_\_

int i; ==> declaration

i = 10; ==> first time initialization

System.out.println(i);==> usage

i = 20; ==> re-initialization

System.out.println(i); ==>usage

i = 0; ==> re-initialization

System.out.println(i); ==>usage

i = -100; ==> re-initialization

System.out.println(i); ==>usage

Note: a variable can be re-initialized any no. of times.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

byte < short < int < long < float < double

byte ==> -128 to 127

short ==> -32768 to 32767

int ==> -2147483648 to 2147483647

long ==> -9223372036854775808 to 9223372036854775807

float ==> 1.40239846e-45f to 3.40282347e+38f

double ==> 4.94065645841246544e-324 to 1.79769313486231570e+308

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_String concatination\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

String can be added to anything. Anything can be added to String

- In a single line by using ',' as seperator we can declare any no. of varables, and initialize as well. but that should be of the same type.

Note: c4 = c6 = 'c'; ==> compound assignment

- before the usage of local variables compulsary it should be initialized.

- local variables must be initialized before usage.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_String-Concatination\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

- Concatination is the process of adding anything with the string or adding string with anything.

- Concatination referes to the process of combining or joining two or more strings together to create a single string.

- we can combine strings with other data types like numbers or variables.

- In java the most common way to concatenate strings is by using the '+' operator.

Note: by default the whole number is considered as int values only. and we should follow the int range otherwise you will get integer number is too large.

Note: Java is case sensitive.

Ex: variable and VARIABLE or VaRiAbLe ==> all are not same. They are different.

(result1 == result2 == result3);

true == false == false

false == false ==> true

Note: we can just declare the local variables without initializing it as long as yo dont use them.

if you are trying to use local variables without initializing then you will get an error.

Note: local variables are local to that declared methods only, they will not be visible outside of the declared methods.

in otherwords local variables scope is limited to the declared methods only.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_comments-in-java\_\_\_\_\_\_\_\_\_\_\_\_\_

- comments are developed for the information puropose about a particular line of code or code snippet or about the whole document.

- comments are ignored by the compiler.

- comments are used for documentation and explanatory purpose.

- comments not executed as part of the program's logic and do not affect the program functionality(execution).

There are three of comments in java

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

1. single line comment

Ex: //This is a single line comment

2. multiline comment(block comment)

Ex: /\* any no. of lines of code \*/

3. documentation comment(for generating API documentation kind of things)

/\*

\*some info

\*some info

\*some info

\*some info

\*/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_UNARY OPERATORS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

INCREMENT AND DECREMENT

PRE-POST PRE-POST

- in case post (increment, decrement) - the modified value will be affected from the next usage not from current usage(immediate usage)

i, e wherever the operation is going on there the value remains same, from the next usage the modified value will be applied.

- in case of pre(increment, decrement) - the modified value will be immedeately

applied.

i, e wherever the operation is going on there itself the modified value will be applied.

Access Modifiers or Access Specifiers

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

In Java we have four access specifiers

1. private -> narrower

2. default(package, friendly)

3. protected

4. public -> wider

1. private

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

- In Java, the 'private' keyword is an access modifier used to restrict the visibility(scope) of a class, method(member functions), field(data members, variables)

- private members scope(visibility) is limited to only that class.Not outside.

- If you are trying to access it outside you will get a CTE.

- constructors(arg, no arg) can also be declared with access modifiers.

- if we declare a constructor as private then, we cant able to call that constructor from outside of the declared class.

- if you are calling the private constructor within the declared class itself then, its syntactically proper.

- constructors are not be inherited to the subclasses but they are involving in the inheritance process.

class Q

{

private Q()

{

System.out.println("Q()");

}

}

class R extends Q

{

}

- for the above program we will be getting error, bcz

In each and every class there will be a default constructor in the R class also that default constructor first statement( super() ) is calling super class no arg constructor and which is private, so that is why the CTE.

- if you are not providing any access specifier for a constructor then it is with the default access specifier.

- in case of SIB(static initialization block), and in case of IIB(instance initialization block) we cant declared them with any access specifier.

if you do you will get error.

- if we have two java files with the different classes, and they are in the same folder, assume you are tyring to access one java file class within the other java file, compiler can able to easily identify that java file and compile that java file as well.

Note: - only if the class is declared with default or public.

- if you are trying to access the members of that class, those members

must be default, or public but not private!

the only scope of default members

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

DEFAULT - within the class, within same package sub-class,

within the same package non-sub class,

within the indirect sub-classes of the same package.

\*\*\*\*\*\*\*\*Anywhere in the same package is possible\*\*\*\*\*\*\*\*

- in software rst represents reStructuredText

- default members doesnt inherits to the different package sub-class.

- if you are using package statement and import statment then,

first you must keep package declaration then only import declaration

- while importing a package if you metion '.\*' that indicates that you are importing all the files from that particular package. Ex: import lara.\*;

- but if you wanted to import only a particular class you can use the syntax import packageName.ClassName; Ex: import lara.N; \*\* This type of import is recommended bcz instead importing all the files that will be the unneccessory load. \*\*

- once you import a package in the file, that package or package members are availble to the entire java file itself(where import statment is declared).

Scenario :: if you are importing a particular class from a package then that class only will be considered for that entire java file instead of already available class in the same package.

if you are not using import statement then only it will be considering the current package class with the same name while you are refering.

- For the local variable access specifiers are NOT APPLIACABLE.

only for static or non-static variables

- protected members can be accesses outside of the package inside the subclass but

-Whichever the class is Inheriting (the subclass) in another package, by using that class object reference only you can access the protected member.

-you cant access protected member by using the super class(from the other-package which we have mentioned for the extending) object reference.

Ex: package rst;

class G extends lara.A

Note: By using G class object reference you can access the protected member but not by using the A class object reference.

- protected members can be inherited to the indirect subclasses in another package, but the ground rule is that you must create an object to that indirect subclass only, by using the reference of that indirect subclass object only you can access the protected member.

Summary

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

1. public ==> The public access specifier allows unrestricted access to the class, method, variable, or constructor from any other class or package.

2. private ==> The private access specifier restricts access to only within the same class.it prevents other classes from accessing private members.

3. protected ==> The protected access specifier allows access to the members within the same package and only subclass of outside the other package.

4. Default ==> if no access specifier is specified, it is considered as the default access specifier. it allows access to members wihin the same package but not from outside the package.(WITHIN PACKAGE LEVEL ONLY)

Important Points

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

1. Default access specifier is also called as (PACKAGE-PRIVATE)

2. Access specifiers can be applied to Interfaces as well

3. Access specifiers are part of java's encapsulation mechanism, which helps in achieving data hiding and abstraction.

4. public members are accessible from anywhere, both within and outside the class or package.

5. Access specifiers are not applicable to local variables (methods or blocks)

6. Access specifiers are not applicable to method or constructor parameters(arguments) variables.

7. Access specifiers play a role in defining the visibility of members during inheritance, composition, and polymorphism.

8. It is good practice to use the most restrictive access specifier that fulfills the requirements. This helps to control access.

9. Access specifiers can be combined with non-access modifers such as static, final, abstract, synchronized, etc. These modifiers further define the behavior and characterristics of the members.

Abstraction

\_\_\_\_\_\_\_

- Abstraction referes to the process of (mechanism) "HIDING THE IMPLEMENTATION DETAILS AND JUST SHOWING THE FUNCTIONALITY"

- Showing only essential things to user and hiding the internal details.

1. We can make class as abstract.

2. Abstract class can become a member of the java file.

3. For abstract class also compiler will be generating a '.class' file.

4. By making class abstract we can achieve 0-100% of abstractness.

5. We are achieving data hiding through abstraction.

6. Abstract class can have abstract methods as well as non-abstract methods.

7.if the method is having a body thats called implemented, defined, concrete method.(it should be having { } )

8. If the method is not having a body then thats called non-implemented or abstract method.

9. If you have at least one abstract method inside a class then class MUST be declared as abstract.

10. But.. An abstract classs can have both abstract methods as well as non-abstract methods.

11. Abstract class is not 100% abstract since in the abstract class we can have both abstract method and non-abstract methods.

12. Encapsulation VS Abstraction

1. In case of encapsulation we are making data members as private, so that no-other classes can get a direct access to those members.

we have a control over the data, we can make it read-only, write-only, or both. We acheiving data hiding(from other classes) also.

2. In case of abstraction we are making member functions(methods) as abstract. We are just hiding the implementation(internal details) of that method(which is unncessarry in that class). We are acheving data hiding here as well(but which is unncecessary for that class only), but ofcourse we are implementing that method in an another class which is definately neccessary for that class.

13.By using abstraction we can achieve better code readability as well as reducing complexeties(unnecessary code, which is an advantage).

14.abstract methods must not have a body(implemention, definition, or {})

15. Abstract class cant be instatiated(we cant create an object to the abstract class).

Note: Multiple Inheritace is not possible in case of classes in java(one class cant be able extends more than one class at a time)

Note: Abstract class can contain constructors. because abstract class allows defined methods and non-defined methods

Important :: Avoid keeping a class as abstract if there are no abstract methods are present on that class otherwise that class cant be instantiated(cant able to create the object)

\*\*Interface\*\*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

- We can achieve 100% abstraction if we want in case of Interfaces.

- Interface also considered as class internally, but having different functionality than normal class.

- Interface can become a member of the Java file.

- Compiler will be generating .class file for the interface as well.

- Interface provides a way to achieve abstraction and as well as multiple inheritance which is not possible by normal class.

- Interface is by default abstract, So abstract keyword is optional

- By default interface variables are PUBLIC STATIC FINAL, if you are not giving all these modifiers compiler only will be adding this modifiers.

- Methods are by default PUBLIC, ABSTRACT

- In java JDK8 normal methods compulsary abstract

Note: with return type and just method name that method name must be abstract.

- Interface will not be allowing normal defined methods.

- Constructors are not allowed in interfaces. bcz contructors are defined.

- Inside the interface SIB and IIB blocks are not allowed, bcz they are defined.

- Interfaces cant be instatiated(cant create an object to interface)

- For Interface we can just create a reference variable

- reference variables can hold 'null' value.

- while implementing the interface methods in the class you must follow the arguments.

- while implementing the interface methods in the class you must follow the return type as well.

relational keywords

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

class TO class ==> extends

class TO interface ==> implements

interface TO class ==> extends

interface TO interface ==> extends

- One class can able to implement multiple interfaces at a time.

- We are achieving multiple inheritance by using interfaces.(by implementing multiple interfaces at a time)

- One class extend another class as well as can implement any no. of interfaces at a time.

- if you want to use extend and as well implements then FIRST GO FOR EXTENDS THEN IMPLEMENTS

- Interface to interface extends

- one interface can extends any no. of interfaces by comma as seperator

- from java8 we can have any no. of static and default methods which are implemented(having a body, concrete methods). that is the feature of java8 with respect to interface.

- The valid access modifiers for interfaces are 'public' and default access(no explicit access modifier). when no access modifier is specified, it is considered the default access, which allows access to the interface within the same package.

- interface methods cannot be declared with the 'protected' or 'private' access modifiers. the access modifiers avialble for interface methods are 'public' and the default access(no explicit access specifiers)

- An interface in java is a contract that defines a set of methods that implementing classes must provide. by default, all methods declared within an interface are implicitly public. this means that methods are accessible to all classes and interfaces, regardless of their package.

- by default for interface and class the scope will be limited to that package only.if you want to access interface and class across the packages you must declare it as public.

- The members of interface are by default public.

- the members of a class by default package scope only.

- we are using static and default method inside the interface to improve the functionality and flexibility of the interface.

- The inclusion of static methods in interfaces allows the definition of utility methods that are related to the interface but do not depend on any specific implementation. these methods can be called directly on the interface itself, without the need for an instance of a class implementing the interface. this is useful for providing common functionality that can be shared by multiple implementing classes.

------

- One of the main reasons for introducing default methods in java interfaces was to ensure backward compatibility when new methods are added to existing interfaces. in prior versions of java, if a new method was added to an interface, all classes implementing that interface would break unless they were updated to implement the new method. this limitation caused issues when working with libraris or APIs that needed to evolve without breaking existing code.

- By introducing default methods, existing classes implementing the interface dont have to modify their code. they automatically inherit the default implementation of the new method. this allows interface to evolve over time by adding new methods while preserving compatibility with existing implementations.

-----

- The 'abstract' keyword is used to indicate that a class or method is incomplete and must be implemented by a subclass.abstract methods dont have a body and their implementation is deferred to the subclass.

- On the other hand, the 'static' keyword is used to define class-level methods or variables that can be accessed without creating an instance of the class. static methods belong to the class itself and are not associated with any specific instance.

- when combining 'static' and 'abstract' there is a conflict in terms of implementation, 'abstract' methods require the subclass to provide an implementation, while 'static' methods are associated with the class itself and do not have an inheritance relationship with subclasses.

\*\*Therfore abstract and static are illegal combinations\*\*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-------------------------POLYMORPHISM------------------------------------

POLYMORPHISM(derived from the greek word)

POLY - MANY

MORPHISM - FORMS or SHAPE

- Polymorphism is also one of the 4 basic pillars of Object oriented Programming language.

- Polymorphism refers to the ability of something to present in different forms or states.

- An object can take on different forms or exhibit different behaviours depending on the context in which it is used.

- Polymorphism allows for code reusability, flexibility, and extensibility.

- Polymorphism in java is typically achieved through two mechanisms

1. Method overloading

2. Method overriding

- Polymorphism is like having a single name but multiple forms. imagin you have pet, and it can be a dog, a cat, or even a bird. you give your pet a name, lets say "buddy" now, no matter what type of pet buddy is, you can stil call it by the name "buddy".

- imagin that you want to travel to Hyderabad, we can go through BUS, TRAIN, PLAN

- Ex: you may have different classes like Dog, Cat, Bird that all inherit from a common class called 'Pet'. Each of these classes can have their own unique methods or behviours, but when you refer to them using the common "Pet" class, you can access their shared behaviours.

\_\_\_\_\_\_\_\_MethodOverloading\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Rules to be followed

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

1. Methods name must be same.

2. Arguments must be different.

3. Method overloading is an example of compile time polymorphism.

4. Return type can be same or different.

5. Even though the no. of arguments are same, datatpe of that argument must be different.

6. Access specifiers can be same or different.

7. Overloading we can achieve in the same class typically but we can also achieve in the sub-class as well.

**Generics:**

**Why do we need Generics in Java?**

In Java we need generics for type safety purpose. Because we don’t know what type data will be supplied by the user.

**Type Safety:** Generics enable compile-time type checking, ensuring that the code operates on the correct types. This helps detect type errors at compile-time rather than runtime.

Array objects are by default type safe. i. e if we declare string array we can insert only string objects only. By mistake if we r trying to insert any other elements we will get compile time error.

**Ex:** String [] s = new String[200];

s[0] = "abc";

s[1] = 100; 🡺 CE

s[2] = true 🡺 CE

Collection objects are not typesafe by default. It will be allowing any type of elements to the collection container. If we want to restrict the datatype then we must use the generic concept.

**Ex:**

ArrayList list = new ArrayList();

list.add(“abc”);

list.add(10); 🡺 No Compile Time Error.

If we want to restrict the datatype to a particular type then

**Ex:**

ArrayList <String> list = new ArrayList <String>();

list.add(“abc”);

list.add(10); 🡺 compilation error

While retrieving Array elements there is no need to perform typecasting.

String[] s = new String[20];

S[0] = “abc”;

String name = s[0]; 🡺 No typecasting required here.

Because it has been stored in the form of String only.

But while retrieving the elements from ArrayList container compulsory we should perform typecasting.

ArrayList <String> list = new ArrayList <String>();

list.add(“abc”);

list.add(“10”);

String name = list.get(0); 🡺 CE

String name = (String)list.get(0); 🡺 No CTE

Because while storing the elements to the ArrayList container it will be stored in the form of objects. Whatever the type of data you supply that will up casted to the Object type and stored inside the ArrayList container (any collection object container). While we are fetching that element we must downcast that object to the corresponding datatype then only we can supply to the corresponding variables.

To resolve the above 2 problems (type safety, typecasting) generics concept has been introduced in the JDK version 1.5

If we intend to create ArrayList object to hold any String Objects we have to define as follows.

ArrayList<String> list = new ArrayList<String>();

Now we can only store string objects.

For this ArrayList we can only add String objects. if we are adding any other type we will get compile time error.

l.add(“valid”); 🡺 No CTE

l.add(new Integer(10)); 🡺 CTE

**Example without using Generics**

package pack1;

import java.util.ArrayList;

public class A

{

public static void main(String[] args)

{

ArrayList list = new ArrayList ();

list.add("abc");

String content = (String)list.get(0);// compulsory we need to downcast. Because it will be stored in the form of Object.

}

}

**Example with the usage of Generics**

package pack1;

import java.util.ArrayList;

public class A

{

public static void main(String[] args)

{

ArrayList <String> list = new ArrayList <String>();

list.add("abc");

String content = list.get(0);// no need of type casting because it is already stored in the form of String.

}

}

* By using generics we can define parameter for the collection. These parameterized collection classes are nothing but “Generic collection classes”.

package pack1;

import java.util.ArrayList;

import java.util.List;

public class A {

public static void main(String[] args)

{

List<String> list1 = new ArrayList<String>();//for Base type and derived type type casting is applicable but not for parameter type

//ArrayList<Object> list2 = new ArrayList<String>();

}

}

**Program CoreJava/Generics/app1/src/Jupiter.java**

* LineNo. 1 For the Interface also we can use the Generic.
* LineNo. 3 For the method parameter also we can specify the datatype as Generic type.
* LineNo. 4 For the method return type also we can specify the Generic type.
* LineNo 6 while implementing the Jit interface we are specifying the type for the Jit interface <Generic>
* LineNo.8 to LineNo. 14 wherever we have the Generic type there and all we must supply String as a datatype because while implementing we have specified the Generic type as String type.
* Likewise we can able to implement the same interface by using multiple classes but while implementing the interface you suppose to give Generic datatype then all the methods that are implementing inside the class must take that datatype only.

**Program CoreJava/Generics/app1/src/Manager11.java**

* LineNo1. Generic type A extends P
* Generic type A should be the subtype of P
* LineNo10 to 13 all the all the parameters for class Z is a subclass to the P type so there is no issue.
* LineNo14. String is not a subclass to the P type so you will be getting an error.

**Program CoreJava/Generics/app1/src/Manager12.java**

* All the classes which is subclass to the Number class we have for the Generics type A
* LineNo.11 you will get error because String is not subclass to the Number.