Core Java Handwritten Notes

Created By : @coders_notes

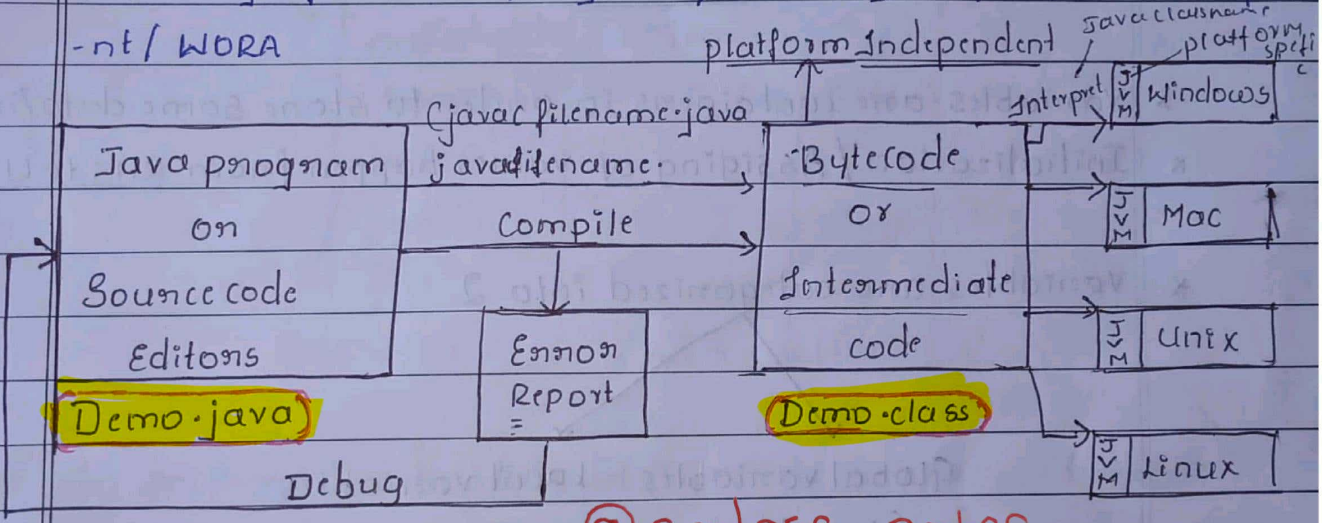
papergrid

Date: 21/01/20

Java

- ✱ Java is a high level programming language.
- ✱ Programming language is a medium to interact with s/m
- ✱ Highlevel language is a language in a normal English i.e Human understandable form
- ✱ James Gosling was a person who introduced Java
- ✱ The company which started java is SunMicros/m (system)
- ✱ Currently Java is owned by oracle.

Working of a Java Program / How is java platform independe -nt / WORA



@codees_notes

JVM : Java virtual Machine (Platform dependent)
WORA : Write Once Run AnyWhere

Bytecode is an intermediate which is neither low-level nor highlevel language so it uses jvm → ① convert to machine level
② Execute line by line

- ✱ Firstly we build the java program using editors and save it with the extension .java
- ✱ Once we are done developing the program we need to compile it

Scanned with CamScanner

* compilation is a process in order to check if there are any errors in my java program or not
* If compilation is unsuccessful we get error report based on error report we need to debug the program
* If compilation is Successful we generate bytecode which is intermediate code / platform independent code
* Extension of all bytecode is .class
* This bytecode can be executed on all platform i.e all operating sh

Variables

* Variables are containers in order to store some data/inform
* Initialization / Assiging of values happen from RHS to LHS
* variables are categorized into 2

$$\begin{tikzcd}
 & \text{variables} \arrow[dl] \arrow[dr] & \\
\text{Global variables (Member)} \arrow[d] \arrow[dr] & & \text{Local variables} \\
\text{Static variable} & \text{Nonstatic variable} &
\end{tikzcd}$$

Example : Age = 25 $\boxed{25}$
LHS RHS Age

Height = 5.5 $\boxed{5.5}$
Height

V → LHS RHS

Datatypes

* Data types are used to indicate or specify the type of data stored into a variables
* Datatypes are categorized into 2
  1) primitive Datatype
  2) Nonprimitive Datatype
* In order to store non decimal numeric values we have the following datatypes
* The difference between those datatypes is Memorysize

Data types	Memory Size	
	Bytes	Bits
byte	1	8
short	2	16
int	4	32
long	8	64

* In order to store decimal numeric values, we have the following datatypes

Data.T	Memorysize	
float	4	32
double	8	64

↑ widely used and default datatype for decimal.

* In order to store true/false we have following datatype

Data.T	Memorysize	
	Bytes	Bits
Boolean	1	8

* In order to store a single character, we make use of char
* char data should be enclosed within 'singlequotes(' ')
* The Memory size of char is 2 bytes = 16bits

Note: All the above mentioned 8 data types are together called as primitive data types.

String: It is a ~~dataStri~~ type to store a sequence of character

* String data has to be enclosed within "double quotes"

Note: Java is Case Sensitive where in lower case letters are not equivalent to upper case letters/values ($a \neq A$)

22/01/2020

Variable Declaration

Syntax:

```
datatype VariableName;
 int age;
double salary;
```

Variable Initialisation

Syntax:

```
VariableName = value;

age = 20;            [20]
                     Age

salary = 5000.69     [5000.69]
                     salary
```

Variable Declaration & Initialization

Syntax:

$$\overbrace{\underbrace{\text{datatype VariableName}}_{\text{Declaration}} = \text{Value};}^{\text{Initialization}}$$

```
boolean x = true;
            false;
```

String subject = "java";
= "ljsp2020", (string can store number, characters but it should be within " " quoted)
char gender = 'M';

Structure of a java program.

1) Class
2) Main method
3) Print Statement

(i) class : Class className

Scope of class

```
{
public static void main (String[] args)
{
        System.out.println(----);
}
}
```

Scope of main method

The filename should be same as classname during file Saving className.java

Ex:1) class Firstprogram

```
{
public static void main (String[] args)
{
System.out.println("Hello world").
}
}
```

Output: Open cmd

```
cd desktop
cd java programs
javac Firstprogram.java -(compiln)
java Firstprogram       -(Interprtn)
Hello world!!!
```

// cd:- change directory to senter to the save files

Ex:2)
```
Class Demo
 {
public static void main (String[] args)
 {
 System.out.println (10);
 System.out.println (45.67);
 System.out.println (true);
 System.out.println ('2');
 System.out.println (" 1jsp@2020");
 }
}
```

Output :
```
javac Demo.java
java Demo
 10
45.67
true
 2
1jsp@2020
```

Ex 3) Write a java program to follow the below statement / Scenarios
i) Create class called as student
ii) Define main method
iii) Under main method initialize 2 variables called as name and age entering those respective values

→
```
Class Student
 {
 public static void main (String [] args)
 {
     String name= "Bhagya";
      int age = 23;
      System.out.println ("name");
       system.out.println (age);
}}
```

O/P
```
Bhagya
23
```

Note: In java, in order to perform Concatenation we make use of '+' operator

Q)
```
class Employee
{
public static void main (String[] args)
{
 int id =101;
 String name = "Jerry";
 double salary= 123.45;
System.out.println(" Employee Id:" +id);
System.out.println(" Employee name is:" +name);
System.out.println(" Employee Salary =" +salary);
System.out.println (id+"  "+name+"  "+salary);
}
```

@codees_notes

Output
```
javac Employee.Java
java Employee
Employee Id=101
Employee Name is : Jerry
Employee Salary= 123.45
101 Jerry 123.45
```

23/1/2024

Operators

1) Arithmetic Operators
2) Assignment Operators
3) Relational / Conditional / comparsion Operators
4) Logical Operators
5) Unary Operators

1) **Arithmetic Operators**

+	:	Addition
−	:	Substraction
*	:	Multiplication
/	:	Division
%	:	Modulus

5 → division
2 | 10
  | 10
  | 0 → modulus

Ex :- 10/2 = 5   10%2 = 0

Example :

```
Class ArithmeticOperators
{
public static void main (String[] args)
 {
     int x=10;
     int y=20;
     int sum = x+y;
     int diff = x-y;
 System.out.println("Sum = "+sum);
 System.out.println (" Difference is " + diff);
 System.out.println (y*5);
 System.out.println (30/3);
 System.out.println (30%3);
```

Output :-

```
Sum = 30
Difference is = -60
100
10
0
```

@codees_notes

Date: / /

2) Assignment operators

```
=
+=
-=
*=
/=
%=
```

@codees_notes

```
int a=5;
    a=a+10   or  a+=10
    a=15
```

$\boxed{\begin{array}{c}\cancel{5}\\15\end{array}}$ a

@codees_notes

```
int  x=30
     x-=20
     x=x-20
     x=30-20
     x=10
```

$\boxed{\begin{array}{c}\cancel{30}\\10\end{array}}$ x

Ex:
```
Class Assignment Operator
 {
 public static void main (String[] args)
  {
   int x=10;
   System.out.println ("value of x is : " +x);
       x+=20;
   System.out.println("value of x is : " +x);
   System.out.println (" = = = = = ");
   int a= 6
   system.out.println ("value of a is: " +a);
       a*=5;
   system.out.println ("value of a is: " +a);
   }
 }
```

O/P

Pathname → cmd
= Assigning
== comparsion

papergrid
Date: / /

Output

```
Value of x is 10
value of x is 30
= = = = = =
value of a is 6
value of a is 30
```

3) Relational / conditional / comparsion Operators

<	: lesser than
>	: Greater than
<=	: lesser than or equal to
>=	: Greater than or equal to
==	: Equals to
!=	: not equal to

Note : Comparsion Operators will always return boolean Values i.e (true/false)

Ex:
```
Class comparsion operator
{
public static void main (String[] args)
  {
    int x=10;
    int y=20;

    boolean result1 = x<y;
    boolean result2 = y>x;
System.out.println(result1 + "\t" + result2);
System.out.println(" = = = = ");
System.out.println( x<=10);
System.out.println( 3 >= 4);
System.out.println(" - - -- ");
System.out.println( x==10);
System.out.println (y==30);
System.out.println ("- - - - ");
System.out.println (X!=10);
System.out.println (y!=30);
```

@codees_notes

O/P
```
true true
- - - -
true
false
- - -
true
false
- - - -
false
true
```

4) logical operators

AND => &&
OR => ||
NOT => !
} return => boolean $\xrightarrow{}$ true = 1, false = 0

AND		&&
T	T	T
T	F	F
F	T	F
F	F	F

OR		||
T	T	T
T	F	T
F	T	T
F	F	F

NOT	!
T	F
F	T

Ex:

```
Class logical operators
    {
    public static void main (String[] args)
    {
    int x=10;
    int y=20;
   boolean result1 = x<y && y>x;
   boolean result2 = x<y && x==1;
 System.out.println (result1);
 system.out.println (result2);
 system.out.println ("---").
 system.out.println (1<2||2>10).
 system.out.println (2<1||2==3);
 system.out.println ("---");
 system.out.println(!true);
 system.out.println(!false);
 system.out.println("---");
 System.out.println(!(1<2));
  }
 }
```

O/P
```
true
false
---
true
false
---
false
true
---
false
```

5) Unary Operators

++ (increment by 1)

-- (Decrement by 1)

$$\begin{tikzcd} & \text{pre increment} \\ ++ \arrow[ur] \arrow[r] & \text{post increment} \end{tikzcd}$$

$$\begin{tikzcd} & \text{pre decrement} \\ -- \arrow[ur] \arrow[r] & \text{post decrement} \end{tikzcd}$$

Ex :
```
Class Unary
{
public static void main (String[] args)
  {
      int x=5;
    System.out.println("x:"+x);
      x++;
    System.out.println("x:"+x);
      ++x;
    System.out.println("x:"+x);
      x--;
    System.out.println("x:"+x);
      --x;
    System.out.println("x:"+x);
      }
  }
```

@codees_notes

Output =>
```
x:5
x:6
x:7
x:6
x:5
```

(1) int x= 10;
int y=x++;
post increment
↓
First assign, then increment

$x\ \boxed{\cancel{10}\ 11}\quad y\ \boxed{10}$

(2) int a= 5
int b= ++a;
pre-Increment
↓
First increment, then assign

$a\ \boxed{\cancel{5}\ 6}\quad b\ \boxed{6}$

③ int i=2
j=i--;
post decrement
First assign, then decrement
i $\boxed{\cancel{2}\ 1}$ j $\boxed{2}$

ⓐ int a=5
int b=++a;
preincrement
First increment, then assign

④ int p=50;
int q=--p;
pre-decrement
First decrement, then assign
p $\boxed{\cancel{50}\ 49}$ q $\boxed{49}$

@codees_notes

Ex) class unary
```
 {
 prsM (String [] args)
  {
    int q=123;
    int w=q++;
    system.out.println(q+"  "+w);

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

       int o=444;
       int p=--o;
   system.out.println(o+" "+p);

  }
}
```

@codees_notes

Decision / conditional statements

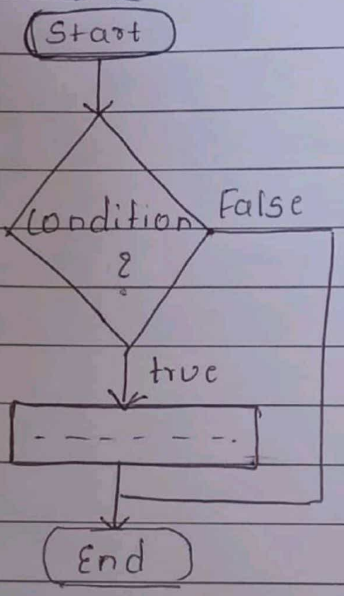
These statements are used to take some decision based on the condition specify. The different decision statements are as follows

1) Simple if
2) if else
3) if else if
4) Nested if
5) switch

1 Simpleif

Simple if is a decision making statements wherein we execute a set of instructions only if the condition is true

Ex : Flow chart



Syntax :

```
if (condition)
{
  statement 1;
  - - - - -          } Set of instructions
  statement n;
}
```

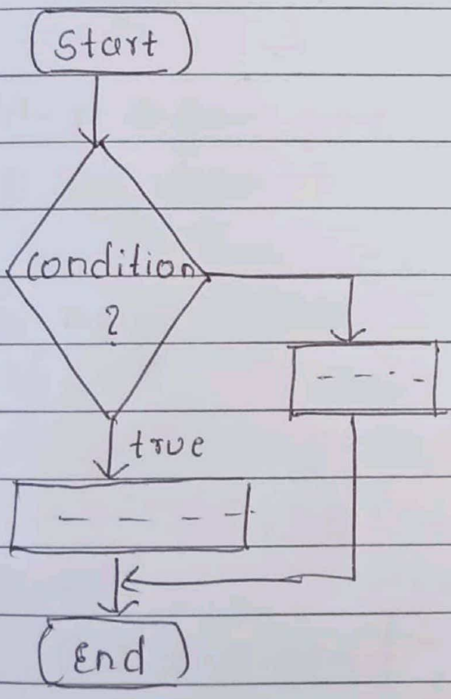
Ex :- class SimpleifDemo   @codees_notes

```
{
psvm (String [] args)
{
 System.out.println ("START");
}
     int a=10;
     int b =10;
if (a<=b)
{
system.out.println (a+" is less than or equal to" +b);
}
system.out.println("End");
}
}
```

② **if else:-** @codees_notes

x if else is a decision making statement where in, if the condition is true we execute if block. Otherwise if the condition is false we execute else block:-

flow chart



Syntax:

```
if (condition)
{
statement 1;    } set of
- - - -         } instructions
statement n;    }
}
else
{
-- --
}
```

Ex

```
class IfelseDemo
 {
  psvm(string[] args)
 {
   s.o.pln("START");
    int x=100

   if (x<=10)
   {
s.o.pln(x+" is lesser than or equal to 10");
   }

 else
  {
  s.o.pln(x+" is greater than 10");
 }
  s.o.pln("------");

 if(true)
 {
 s.o.pln("HI");
 }
else
 {
 s.o.pln("BYE");
}
 s.o.pln("----")

 if(false)
 {
 s.o.pln("WELCOME");
 }
else
{
s.o.pln("Thank you");
}
s.o.pln("end");
}
}
```

o/p:-
```
START
100 is greater than 10
- - - - - - - .
 Hi
- - - - -
Thank you
 END
```

Q) Write a java program to find a number is +ve or -ve

→
```
class Number
 {
  psvm (String[] args)
  {
     int x=5;
     if (x>0)
    {
     s.o.pln ("x is a positive number");
    }
     else
      {
       s.o.pln ("x is a negative number");
      }
    }
```

Date
Page

Q) Write a java program to find a number is even or odd

→
```
class Number1
{
psvm (String[] args)
 {
   int num=4;
   if(num % 2==0);
   {
   s.o.pln (num+" is a Even number");
   }
   else
    {
     s.o.pln (num+" is a odd number");
    }
}
```

@codees_notes

3) Write a java program to find maximum of 2 numbers

→
```
class Number2
 {
  psvm (String[] args)
   {
     int x=5;
     int y=10;

     if(x>y)
  {
   s.o.pln (x+" is a larger than"+y);
  }

  else
   {
   s.o.pln(x+ " is a lesser than" +y);
  }
}
```

@codees_notes

27/1/2020

3) if - else-if

if-else-if is a decision making statement where in we can check multiple conditions

Syntax :
```
if (condition)
{
  ---
  --
}
else if (condition)
{
  ----
}
else if (condition)
{

}
else
{          optional
}
```

Ex :
```
class IfElseIfDemo
{
 psvm (String[] args)
 {
   if num = 250;
    if (num <=10)
    {
 s.o.pln (num+ " is lesser than or equal to 10");
    }
```

@codees_notes

```
else if (num<=20)
 {
 s.o.pln (num+" is lesser than or equal to 20");
 }
elseif (num<=30)
 {
 s.o.pln (num+" is lesser than or equal to 30");
 }
 else
 {
 s.o.pln ("Above Conditions didnot match");
  }
 }
}
```

O/P: Above Condition didnot match

0-34-F
35-59-F
60-100-FC

Ex

```
class IfElseIfDemo1
{
psvm (String[] args)
{
 int marks=25;
 if (marks>=0 && marks<=34)
 {
   s.o.pln ("fail");
 }
  else if (marks>=35 && marks<=59)
 {
   s.o.pln (" First class");
 }
else if (marks >=60 && marks <=100)
{
    s.o.pln("FCD");

 }
 else
  {
    s.o.pln("Invalid Marks");
   }
  }
}
```

O/P : Fail

(4) **Nested if :**

Nested if is a decision making statement where in, one if is presented inside another if condition

Syntax:

```
        if (condition)
         {
          if (condition)
            {
          [    - - -
               - -
          ]  }
            else
            {
            [
             }
```

Ex ① class NestedIfDemo

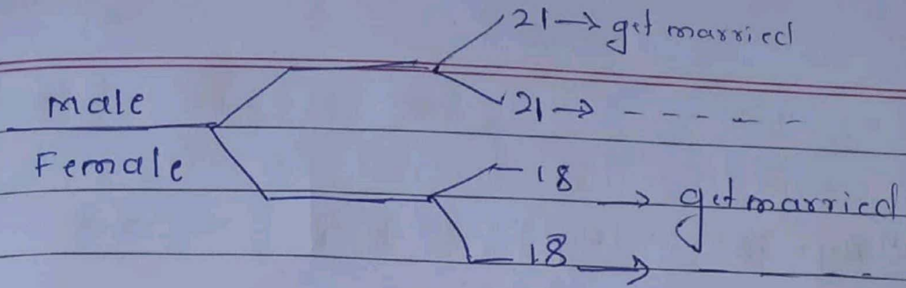
```
{
 psvm (string[] args)
 {
   char id= 'b';
   int password=123;

   if (id=='a');
   {
     s.o.pln (" user id is valid");
       if (password ==123)
       {
         s.o.pln (" password is valid");
         s.o.pln (" login is Successful");
       }
     else
       {
        s.o.pln(" password is invalid");
                (" login is unsuccessful");
       }
     }
     else
       {
         s.o.pln (" user id is Invalid");
                 (" login is Unsuccessful");
       }
   }
```

Output :- ① Userid is Invalid
login is unsuccessful

② User id is valid
Password is valid
login is Successfu

Ex



```
class NestedIfDemo1
{
  psvm(String[] args)
  {
    Char gender='M';
    int age = 24;

    if(gender=='M');
    {
      S.o.pln("Male");
       if(age>=21)
       {
         S.o.pln("Age is :" +age);
         S.o.pln("Get married & hopefully stay happy");
       }
       else
       {
         S.o.pln("age is :" +age);
         S.o.pln("Have patience");
       }
    else if(gender=='F')
     {
       S.o.pln("Female");
        if(age>=18)
        {
          S.o.pln("Age is :"+age);
        }
      else
     {
     s.o.pln ("Age is :" +age);
     }
   }
     else
     {
      s.o.pln ("Gender is Invalid");
     }
   }
```

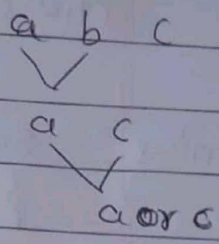
O|P :- Gender is Male
age : 21
Get married & hopefully stay happy
stay happy

(2) Gender is Invalid

8) Write a java program to find largest of 3 numbers
→

```
class largest of threeNumbers
    {
     psvm(string[] args)
      {
         int a =10;
         int b= 5;
         int c= 3;
     s.o.pln ("a:"+a+"b:" +b+"c:"+c");
          if (a>b)
          {
            if (a>c)
             {
              s.o.pln("a is largest");
             }
else
 {
  s.o.pln("c is largest");
 }
}
else if (b>c)
 {
  s.o.pln("b is largest");
 }
else if (c>b)
 {
  s.o.pln("c is largest");
 }
 else
  {
   s.o.pln("Invalid");
  }
 }
}
 s.o.pln("----");

 if (a>b && a>c)
 {
   s.o.pln("a is largest").
 }
else if (b>a && b>c)
  {
  s.o.pln("b is largest");
  }
else if (c>a && c>b)
  {
   s.o.pln("c is largest");
  }
else
 {
 s.o.pln("Invalid")
  }
 }
}
```



28|1|2020

(5) **Switch statement** :-

Switch is a conditional statement generally used for character comparsion

Syntax :-

```
switch (choice/input)
 {
   case 1 : - - - - - -
              break;
   Case 2 : - - - - - -
              break;
        - - -
          :
          :
   case n : - - - -
   default :
```

@codees_notes

Ex :

```
class switchDemo
{
psvM (string[] args)
  {
    int choice=3;
      switch (choice)
  {
case 1 : s.o.pln ("In case 1");
          break;
case 2 : s.o.pln ("In case 2");
          break;
Case 3 : s.o.pln (" In case 3);
          break;
   default : s.o.pln ("Invalid choice");
  }
 }
}
```

O/P :- In case 3

Note: **Break**: is a keyword which is used to transfer the control outside the currently executing block

Ex:
```
class monthvalidation
{
 psvM (String [] args)
  {
    s.o.pln ("start");
    char month = 'Z';
     switch (month)
    {
     case 'J' : s.o.pln (" In January ");
                     break;
     case 'F': s.o.pln (" In February");
                     break;
     case 'M' : s.o.pln (" In March");
                     break;
     default : s.o.pln ("Invalid month");
       }
       s.o.pln ("End");
    }
 }
```

Output:
```
start
Invalid Month
 End
```