Lecture 4- Java program's

* # STRUCTURE of JAVA FILE

"source code that we write will be saved using extension .java".

- Everything written in .java file must be in classes or we can say that every file having .java extension is a class.

- A class with same name as file name must be present in .java file.

# first alphabet of class name can be in upper case. It is the naming convention of class name. however, it is not compulsory to do so.

- class which is having same name as file name must be public class.

- A main function / method must be present in this public class, main is a function from where the program starts.

# How To Run Java program?

Step 1]

* CONVERTING .java TO .class (Terminal):

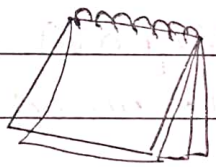
- using javac compiler we can convert .java file to .class → command to convert .java to .class

command ⇒ javac filename.java

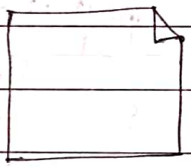
e.g. let name of the file is Main,

command ⇒ javac Main.java

- Above command create a .class file e.g. (Main.class) which contains bytecode.



javac Main.java →



(Main.class)

.java file

.class file

Step 2]

* RUNNING THE PROGRAM

By using java and name of file we can run the program.

command ⇒ java filename

e.g.

command ⇒ java Main

# Components of Java program?



PROGRAM :

→ Public :- It is an access modifier which allows (1st line) to access the class from anywhere.

→ class :- It is a name group of properties & function.

→ Main :- It is just the name of class as same as the name of file.

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

(class) (var) (fn/method)

→ Public (2nd line) :- It is used to allow the program to use main function from anywhere.

→ static :- It is a keyword which helps the main method to run without using objects.

→ void :- It is a keyword used when we do not want to return anything from a method / function.

{ String System Main main
capital capital capital small

→ main :- It is the name of method.

→ String [] args :- It is a command line argument of string type array.

{# if main method is written without String[] args ? The program will compile but not run because JVM will not recognize the main() method. JVM looks for main() method with a string type array as a parameter. }

→ System :- It is a final class defined in java.lang package.

→ out :- It is a variable of printstream type, which is public and static member field of system class.

→ println :- It is a method of printstream class, It prints the arguments passed to it and adds a new line. print can also be used here but it prints only arguments passed to it. It will not add a new line.

# package?

→ It is just a folder in which java files lies.
→ It is used to provide some rules and stuff to our programs.

# PRIMITIVE DATA TYPES?

→ primitive data types are those data types which is not breakable.

Ex:-
String is not a primitive data type so we can break this data type into char
i.e. String "Gautam" can be divided into 'G' 'a' 'u' 't' 'a' 'm'.

But primitives data type are not breakable.
we cannot break a char, int etc.

→ List of primitive data types in java are:-

| Data types | Description | Example |
|---|---|---|
| int | Int is used to store numeric digits | int a = 26; |
| char | char is used to store character | char c = 'A'; |
| float | float is used to store floating point numbers | float f = 98.67 f; |

(All decimal value that we use are by default of double datatype, therefore if we want to store in float we use "f", same for int & long).



| | | |
|---|---|---|
| double | double is used to store longer decimal numbers | double d = 45676.58975; |
| long | long is used to store numeric digits which is not able to stored in int | long l = 15878954832558315L; |
| boolean | It only stores t values i.e. true or false | boolean b = false; |

In float and long we have used f and l, it denotes that the number in the variable is float or long type, if we do not use this java consider float value as double and long value as int.

* Literals :

It is a synthentic representation of boolean, character, string, and numeric data.

Ex:- int a = 10;

Here 10 is called literal.

* Identifiers :

name of variable, methods, class, packages, etc. are known as identifiers.

Ex:- int a = 10;

Here a is identifier.

# comments in java

→ comments are someting which is written in source code but ignored by compiler.

* Two types of comment →

1) single line comment :
used to comment down a single line
(// is used for it)

2) multi line comment :
used to comment down multiple lines
(/* */ is used for it).

# OPERATORS IN JAVA.

* Unary operator :-
unary operator are used with only one operand
Ex : ++ that increment value by 1.

1) + (unary plus) →
not necessary to use since no. are positive without using it

2) - (unary minus) →
invents the sign of an expression
Ex: n1 = 20;
n2 = -n1; // n2 = -20
R = n2; // R = -20

3} ++ (increment operator) →
Increment value by 1.

Ex: n = 10;
++n ; // 11 {Here, first increment then print}
n++ ; // 11 {Here, first print then increment}

after this step current n value is 12, it will not show since we are not printing it after increment.

4} -- (Decrement operator) →
Decrement the value by 1.

Ex: n = 10;
--n ; // 9 {first, decrement then print}
n-- ; // 9 {first print then decrement}

// Actual n value is 8, it is not displaying cause we are not printing after decrement.

5} ! (not / logical complement operator) →
Invert the value of boolean. Basically used to convert true to false or vice versa.

Ex: boolean c = true;
f = !c ; // f = false

boolean f = false;
T = !f ; // T = true

* Arithmetic operator :-
These operators are used to perform arithmetic operations on variables and data.

1} + (Addition) →
To add two values / variables.
ex:- 10 + 20; // 30

2} - (substraction) →
To substract two values / variables.
ex:- 10 - 20 ; // -10

3} * (multiplication) →
To multiply two values / variables.
ex:- a = 10;
b = 10;
a * b ; // 10 * 10 = 100

4} / (Division) →
It is use to divide two values / variables.
It returns the quotient after division.
ex:- (9 / 2) ; // 4 (if no. are int result is int)
(9.0 / 2) ; // 4.5 } if any one no is float then }
(9.0 / 2.0) ; // 4.5 } result is also float

5} % (modulo) →
It returns the remainder after dividing two value / variables.
ex: (9 / 2) ; // 1
(7 / 4) ; // 3

* shift operator :-

if left shift operator (<<) →

left shift all bits towards the left by a certain no. of specified bits.

ex. 0 0 1 0 // 2

32 16 8 4 2 1

0 0 1 0.0 0 // 2<<2 = 8

It is the no of '0' to be added

org bits 2 '0' added

(Here, on right side no. of specified zeros get added & shift the org bits towards left. in our case 2 zeros are added and now we have to make sum of values where 1 is present and the sum = final result).

ex. 1 0 1 0 // 10

64 32 16 8 4 2 1

1 0 1 0.0 0 0 // 10<<3 = 80

80

Here value is 3, so 3 '0' are added after.

(. is our cursor) - just imagine.

ex. 1 0 1 0 // 10

32 16 8 4 2 1

1 0 1 0.0 0 // 10<<2 = 40

40

ex. 1 1 1 1 // 15

128 64 32 16 8 4 2 1

1 1 1 1.0 0 0 0 // 15 << 4 = 240

ANS = sum of

difference → ( >> is we to preserve sign bit (+, -)
>>> does not preserve sign bit)

Page No. 11
Date

2) (>>) ~~Java~~ Right shift operator →
It shifts all bits towards the right by a certain no. of specified bits.

ex: 1 0 0 0 // 8

2 1
1 0 | 0 0. // 8 >> 2 = 2

(Here, initially our cursor is at ., but we have to right shift by 2. so we will move our cursor towards left by 2. now last two right most bits are shifted towards right and our cursor ~~position~~ current position is denoted by (|). To get our ans we have to find decimal value of {binary (10) / bits which are at the left side of cursor}.

∴ 10 = 2

ex: 1 0 1 0 0 // 20

4 2 1
1 0 1. 0 0 // 20 >> 2 = 5 (. means cursor, just to understand)

5

ex: 1 0 1 0 // 10

1. 0 1 0 // 10 >>3 = 1

1

ex: 1 0 1 0 0 // 20

1 0. 1 0 0 // 20 >> 3 = 2

2

8 >> 2 = 2
-8 >> 2 = -2

ex. -8>>2 { - is the signed bit so here, first it find 2's (8) & in that no sign is present. just to eleminate signed bit; then it perform right shift on 2's (-8) }.

8 = 1000

1's complement = 0111

2's (-8) = 0111
+ 1
1000

3) (>>>) unsigned Right shift operator → works same as >> for positive values but not for negative values.

ex: 8>>>2 = 10|00 . = 2
2

8>>2 = 10|00. = 2 (both values are same only sign is different cause (>>) preserve sign bit. basically it perform same action for both +ve & -ve value & only change sign of result as per given values)
2

ex: -8>>2 = -2

-8>>>2 = 1073741319 (Both values are different cause it does not preserve sign bit)

(To preserve sign bit we need to add 0 in the MSB)

Todo, this do above calc.

* Relational operator :-

Relational operator are used to check the relationship betn two operands.

Ex :- a < b

| operator | description | Example |
|---|---|---|
| == | is equal to | 3 == 5 return false |
| != | not equal to | 3 != 5 return True |
| > | greater than | 3 > 5 return false |
| < | less than | 3 < 5 return true |
| >= | greater than or equal to | 3 >= 5 return false |
| <= | less than or equal to | 3 <= 5 return true |

* Assignment operator :-

Assignment operators are used to assign values to variables.

Ex: int age;
age = 5;

Here, (=) is assignment operator. it assign value on right side to the variable(age).

| operator | Description/ Equivalent to | Example |
|---|---|---|
| = | a = b; | a = b; |
| += | a = a+b; | a += b; |
| -= | a = a-b; | a -= b; |
| *= | a = a*b; | a *= b; |
| /= | a = a/b; | a /= b; |
| %= | a = a%b; | a %= b; |

* Ternary operator :-

It is used as one line if then else statement

Ex: syntax

variable = Expression ? expression 1 : expression 2

Here, how it works

→ if Expression is true, expression 1 is assigned to variable.

→ if Expression is false, expression 2 is assigned to variable.

```
int feb = 29;
Ex: result = (feb == 28) ? "not leap year" : "leap year";
print(result); // leap year
```

* Java instanceof operator :-
It check whether an object is an instance of a particular class.

Ex: String str = "programiz";
boolean result;
result = str instanceof String;
SOP (result); // true

* Bitwise operator :-

i) | (Bitwise OR) →
It returns the values based on logical OR Table.

| a | b | a \| b |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Ex: 10 // 1 0 1 0 = a
12 // 1 1 0 0 = b
(10|12) // 1 1 1 0 = 14 = a|b
binary value → 8 4 2 ↑1 (values are based on above table value)

(logic behind code)

Ex: int a = 10, int b = 12, result;
result = a|b; // 14

(All Bitwise operator works in same manner, they follow rules based on their respective table to assign value (0,1))

2) (Bitwise AND operator) & →

It returns value based on below AND table

| a | b | a & b |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Ex: 10 // 1 0 1 0 = a

12 // 1 1 0 0 = b

8 4 2 1

(10 & 12) // 1 0 0 0 = 8

we will compare a & b. using rules in above table we will assign values

3) (Bitwise XOR operator) ^ →

It returns value based on below XOR table

| a | b | a ^ b |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(Again we will assign value based on above table (like And, OR).

Ex: 10 // 1 0 1 0

12 // 1 1 0 0

8 4 2 1

(10 ^ 12) // 0 1 1 0 = 6

* Logical operators :-

4) (Bitwise complement operator) ~ →

It is an unary operator i.e. works with only one operand.

It changes 0 to 1 & 1 to 0.

~~(It returns the 2's complement of given no.)~~

# {Note : ~N is equal to -(N+1)} # {for +ve no.}

Ex: 35 = 00100011

~35 = -36

Ex: int n = 10;

r = ~n; // -11

# for -ve no.

{note: ~N is equal to (-N)-1}

Ex: a = -9

~a = 8

Ex: int n = -15;

r = ~n;

SOP (r); // 14

* logical operator :-

logical operators are used to check whether an expression is true or false. used in decision making

| operator | example | meaning |
|---|---|---|
| && | a && b | true only if both a and b are true |
| \|\| | a \|\| b | true if either a or b is true |
| ! | !a | true if a is false and vice versa. |

(S in scanner & System is capital)



# INPUTS IN JAVA

* we have scanner class available in java.util package to take input

To use this class we have to
1) import java.util package in our life.
2) create object of the scanner class.
3) use that object to take input from the keyboard.

(class that allows us to take input) (creating object) (take input from std input stream i.e. keyboard)

Syntax :-

```
Scanner input = new Scanner (System.in);
```

Scanner :- It is a class required to take input, it is present in java.util package.

input :- It is an object that we are creating to take input. ~~It can be name as per user.~~

new :- It is a keyword used to create an object in java.

System.in :- System is a class and in is a variable that denotes we are taking input from standard input ~~system~~ stream (i.e. keyboard).

* Types of inputs :-
if user wants to take

1) int input →

nextInt() is a function used to take input of int.

Syntax →

```
Scanner input = new Scanner (System.in);
    int rollno = input.nextInt();
```

2) float input →

nextFloat() is a function used to take input of float.

syntax →

```
Scanner input = new Scanner (System.in);
    float marks = input.nextFloat();
```

3) String input →

Two ways to take string input

(a) using next() →

It will take one word input till space occurs.

Syntax →

```
Scanner input = new Scanner (System.in);
    String s1 = input.next();
```

Input :- Hey gautam

Output :- Hey

(b) using nextLine() →

It will take all string input including space.

syntax →

```
Scanner input = new Scanner (System.in);
String s2 = input.nextLine();
```

Example:-

```
public static void main (String [] args) {
 Scanner input = new Scanner (System.in);
 SOP ("enter n1");
 int n1 = input.nextInt();
 SOP ("enter n2");
 int n2 = input.nextInt();
 int sum = n1+n2;
 SOP (sum);
```

# Type conversion?

when one type of data is assigned to another type of variable an automatic type conversion will take place under some condition.

* widening or automatic type conversion →

conditions →

1) Two types should be compatible.
2) Destination type should be greater then the source type.

byte → short → int → long → float → double

```
ex:- int i = 100;     // 100
     long l = i;      // 100
     float f = l;     // 100.0
```

* narrowing or explicit conversion: this happens when we assign a value of large data type to a smaller data type.
double → float → long → int → short → byte

ex:- double d = 100.4;
long l = (long) d;
int i = (int) l;

* Type casting →
when we convert one type of data to another type is known as type casting.
Ex :- int num = (int) (67.564f)

* Automatic type promotion in expressions →
when evaluating expressions, the intermediate value may exceed the range of operand & hence the expression value will be promoted.

Condition →
1) Java automatically promotes each byte, short or char operand to int when evaluating an expression.
2) if one operand is a long, float or double the whole expression is promoted a long, float or double.
ex:- byte a = 40;
byte b = 50; byte c = 100;
int d = (a*b)/c;
SOP (d) // a*b occur it become 2000 which is out of range of byte so it get promoted to int type.

→ If we want to store large value into small data type
e.g. byte b = 50;
b = (byte) (b* 2); → type casting int to byte.