JAVA ①

package com.company → used to organise classes belonging to same category or similar function.

main is declared as public static main{{{ }
public static void main(string[] args){ ——— }.

Printf in C → System.out.println("———");

print statement → system.out.print(" ");

println statement statement terminates with \n automatic

Each and every line in java must be terminated with ;

Java is a OOPs oriented programming language

Variables in Java:
container which stores a value in a Java program

Datatypes:
primitive → Numeric : integer, floatingpoint,
byte-8 double-64
short-16
int-32 float-32.
long-64

Non-numeric : character, boobean

Non-primitive → strings, arrays, user defined classes.

Write L at the end of the long number to indicate the datatype as long because Java takes any numbers as int other than long

The JAVA interprets a decimal number as double by default. To write F at the end to denote as a floating point.

We can store multiple strings in one using tripple double inverted commas → """ """ """;

Comments:

// → single line comment ctrl + /

/* */ → multi line comment

Typecasting: changing datatype from one to another.

Ex: int x = (int) y;

Constant value:

use final while defining with datatype → final int x=1;

Scanner class:

```
import java.util.Scanner;
:
Scanner sc = new Scanner(System.in);
int x = sc.nextInt();
```

The next function only reads the input till a break point.

Scanner class does not allow to read a single character input. So we can access it or write it as.

```
char c = sc.next().charAt(0);
```

Input:

We have to declare Scanner class for this operation

The argument of the object declared in Scanner class is taken as System.in → since the input is being taken from the system (:: Scanf) from the user.

The variables are printed with output by +variable name.

The condition in if condition must return a boolean value.

If break; is not used in switch cases, the compiler falls and executes all the following cases. till it breaks.

Enhanced switch case:

```
Switch (condition)
  {
  case label1 -> {        }
  case label2 -> {        }
   default    -> {        }
```

Ternary operator → if else statement.

condition ? True statement : false Statement.

---

logical AND - &, logical short circuit AND - &&.

logical OR - |, logical short circuit OR - ||

Not operator → !

For loop is preferable when exact numbes of iterations.

Integer.MIN_VALUE → minimum integer value

Math.max(a,b); → gives the biggest value among both

Math.random() → gives a random integer number → 0.0 <= x < 1

Break; continue; can be used

array name.length → length of the array.

declaring array in Java:

datatype[] arrayname = new datatype[size];

Variables can be printed as value in print statemen as " + variablenamet"

for each loop:

for(itrater : variable){} ← datatype must be same where variable is an array which has increment index in for each syntax in itself

Introduction Set up Getting started Operatoss & Control Statements loop Array.

JAVA (2)

The main class is defined as class main{

public static void main (String[] args){ }}.

A particular address character of a string can be accessed as

Stringname.charAt(position);

Sring length is given by stringname.length();

If two Strings are exactly equal but you dont want them to point towards the same address use new keyword while declaring the second.

⇒ String stringname = new String("s");

Sringname.equals(Stringname-2) → This function unlike the ==, checks the contents of two strings than the object references.

Stringname.indexOf(' '); returns the index of first occurence of the specified char or string. (-1) if true

The Java compiler actually returns "true" or "false" rather than binary for conditional statements.

Stringname - contains(" "); returns true or false where the given input is present in the string or not

Stringname.toLowerCase(); } both dont modify.
Stringname - toUpperCase(); } String but can create n.

Stringname-replace(target, replacement); replaces the target by the replacement. If multiple targets, then every targets are replaced.

Stringname.substring(index); starts the new string from the target index all the way till the end.

String name.substring(index1, index2); starts the new String from index1 till index2 of the string. (index2 not included)

Two Strings can be added using +

To convert any variables into String, we can use the function toString();

StringBuilder stringname = new StringBuilder("—");
→Stringname.append("-"); → adds the entered line to the first string itself.
→Stringname.insert(position, "—") → adds the entered line to the position of the first string itself.
→Stringname.replace(index1, index2, " ") → replaces the entered string deleting the words from index1 to index2 and replacing it with the entered value.
→Stringname.delete(index1, index2) → deletes characters between index1 and index2.

StringBuilder String can be converted into normal String by String stringname2 = Stringname1.toString();

Stringname.reverse(); → reverses the string only in StringBuilder class.

Syntax of a function/method:

~~void~~ Returntype functionname(Parameters){Body;}

Datatypes of the parameters sent between the main and function body is and must be same.

When you don't know the number of elements sent as parameter to the function, just use the syntax.

datatype functionname(datatype ...varargs){}

Wrapper class:

Java wrapper classes provide a mechanism to use primitive datatype as objects.

(byte → Byte) (short → Short) (int → Integer)
(long → Long) (float → Float) (double → Double)
(boolean → Boolean) (char → Character)

Java collections must be declared using the wrapperclass.

Java synchronization works with objects in Multithreading.

Autoboxing:

Autoboxing is the automatic conversion that the Java compiler makes between the primitive datatype and their corresponding wrapper class.

Unboxing:

The automatic conversion of wrapper object to its corresponding primitive datatype is known as unboxing.

Math.max(x,y) will return maximum value

Math.min(x,y) will return minimum value.

Math.floor(x,y) → with return the value x only.

Math.ceil(x,y) → with return the value x+1.

Math.round(x.y) → will return x if $y < 5$ and returns $x+1$ if $y \geq 5$.

Math.log(x) → will return logarithmic value of x w.r.t e.

Math.log10(x) → will return logarithmic value of x w.r.t 10.

Math.pow(x,y) → will return x^y

Math.sqrt(x) → will return the value of $\sqrt{x}$

Math.E → 2.718281828459045.

Math.PI → 3.141592653589793

Math.Sin(PI/x) → will return the sine value and

Math.Cos(PI x) → will return the cos value in Rad

Math.tan(PI/x) → will return the tan value in Ra

BigInteger is a class that has to be imported as import Java.math.BigInteger and then declare as BigInteger x = new BigInteger(" ");

BigInteger cant be added using + operator hence we use BigInteger x = a.add(b); for a+b.

Similarly for a.subtract(b) → a−b for BigIntegers

for multiplying a*b → a.multiply(b);
for dividing a/b → a.divide(b);
for power a^b → a.pow(b);
for modulus a%b → a.mod(b);
} for BigIntegers variables.

Classes and objects:

```
class classname{
  datatype datamembers;
  returntype membesfunction(){ }
}
```

Before main ↑

After/inside main ↓

classname obj = new classname();

Access the datamembers using dot operator as obj.datamember.

Access the member functions using dot operator as obj.memberfunction.

Constructor:

Has same name as that of the class.

Has not return types/datatypes.

It takes arguments as

~~const~~ classname(datatype arguments){ }

This.variable → Prints the variable value in the particular block.

---

variable name.toString() → will convert the variable to string.
