JAVA

Installation

Download → Oracle website

↓

Install using jdk

↓

Configure JAVA PATH

↓

java -version → $\boxed{\text{cp}}$

⇓

To check Java version

Package

⇓

Class

⇓

method

Class in Java have starts with Capital letter

Datatypes

$$\begin{tikzcd}
 & & \text{Datatypes} \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] \arrow[ddl] \arrow[ddrr] & & \\
\boxed{\text{int}} & \boxed{\text{float}} & \boxed{\text{double}} & \boxed{\text{Char}} & \\
 & \boxed{\text{Boolean}} & & & \boxed{\text{String}}
\end{tikzcd}$$

String → Acts as a predefined class

float → Can take only one value after decimal point.

double → Can take more values after decimal point

JAVA OPERATORS

Arithmetic	Relational	Decrement Operators	Logical Operators	Assignment operators	Increment operators
+, -, *, /, %	<, >, <=, >=, !=, ==	--	&&, ||, !	=, +=, *=, /=	++

CONDITIONAL STATEMENT

→ if

→ if -- else

→ Nested -- if else

→ Switch Case

```
switch (Variable)
  {
Case Condition1 : //code; break;
Case Condition2 : //code; break;
  =
} default : //code
```

```
if (condition)
   {
      //code
   }
else if (condition)
   {
      //code
   }
else if (condition)
   { //code
   }
 else { //code }
```

LOOPS

→ while
→ Do while } → • initiate • condition • increment/decrement
→ for
→ for... each

%
⇓
will give remainder

/
⇓
will divide no. and give Quotient

8%2
⇓
0

Statement

Break

To come out of loop or basis of certain condition

Continue

For skipping or basis of certain condition
⇓
```
for (i=1; i≤10; i++){
    if(i==5){
        continue;
    }
    Sopln(i);
```
int a[]={-,-,-,-};

Arrays

Single Dimension

int a[] = new int[5];
↓
Length of Array

declaration

String s[] = new String[];

s.length() → method which will return length of Array

Char c[~~p~~] = new char[ ];

Object a[] = new Object[];

Object class
⇓
can store all datatypes of data

B. Two Dimensional

int a[][] = new int[][];
columns
↓
rows

FOR EACH

```
for (Object i : a)
{
    System.out.println(i);
}
```

it will store all values of a in i and will read one by one

```
Object a[] = new Object[3]
a[0] = 1;
a[1] = 'A';
a[2] = "xi";
```

STRING FUNCTIONS

length() → no. of characters in String
Concat() → Joins two strings
Equals() → To compare 2 string
Contains() → To check certain letters are present or not in string
substring() → substring(0, 3) → starting index, → finish index — Starts from 0
replace() → replace('e', 'a') — will replace e with a in string
equalsIgnoreCase() → ~~~~ To compare 2 string by Ignoring Case sensitivity

Language

↓ Structural
↳ C, Visualbasic

↓ OOPs Language
↳ JAVA, C++ etc

An instance of a class Object → is a physical Entity (Rohit, Roli, Chotu, etc)

Class → is a logical Entity (Human Being)
↓
Collection of variables & methods
↳ a piece of code which will perform certain task.

OOPS

Creating Object

Employee a = new Employee()

Employee → Class
a → reference variable
Employee() → instantiation

METhOD SYNTAX

```
void methodname() {        → will not return anything
     = //code
}
```

Initializing Values to Class Variables

→ Using Object
→ Using Constructor → kind of method. Constructor ~~class~~ name is same as ~~constructor~~ Class name

```
Public class Employee {        → class name
    int empid;
    String empname;
    int Salary;
  Employee(int id, String name, int sal)
  {
    empid = id;
    empname = name;
    Salary = sal;
  }
```

Constructor name

Parameters

Employee empl = new Employee(101, "RAJ", 80000);

empl.display();

⇒ Using method

```
empl.setdata(101, "RAS", 80,000);
empl.display();
```

METHODS

⇒ may take parameters
⇒ may not take parameters
⇒ may return value
⇒ may not return any value → void

METHOD OVERLOADING

In one class if 2 methods [illegible] with same name is defined as method overloading

CONSTRUCTOR OVERLOADING

will take parameters like method

used to initialize the value

⇒ special kind of method
⇒ constructor name should be same as class name
⇒ will not return any value
⇒ will be invoked at time of object creation

A class contains more than one constructor with same name is called constructor overloading

Differentiating b/w 2 methods with same name

⇒ no. of parameters
⇒ order of parameters
⇒ datatype of parameters

this keyword

```
this.a = a
this.b = b
```

⇒ if we don't use external parameters and use local parameters (variables) only

CONSTRUCTOR	METHOD
→ will not return value	→ may return value
→ name same as class name	→ any name can be assigned
→ it is invoked at time of creating object	→ through object method can be called (object must be created)

$$\begin{tikzcd}
 & \text{CONSTRUCTOR} \arrow[dl] \arrow[dr] & \\
\boxed{\text{default Constructor}} \arrow[d] & & \boxed{\text{Parameterized Constructor}} \arrow[d] \\
\text{If zero Parameters are passed} & & \text{If Parameters are passed}
\end{tikzcd}$$

Static Keyword

If variable or method needs to be shared by multiple objects then variable/method are made or defined as static

Advantage

→ duplication avoid

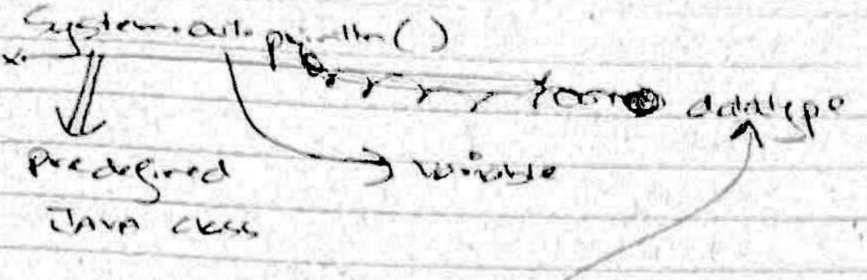
→ reusability

→ directly accessible by Classname

* NOTE → Static method/Static variable can only access Static stuff (direct access without any object creation)

→ Static method can access static stuff by creating object

→ Non static methods can access everything directly without any object creation.



```
Class System {
    static PrintStream out
}
```

To access above System.out.println()

method belonging to printStream type

Public static void main ()

- This is required as JVM understands this only apart from this JVM won't accept anything

JDK, JRE & JVM

(JDK) → Java development kit contains internally JRE & JVM - used for developing java applications
⇓
Full kit

(JRE) → Java Runtime Env. → used for running / installing application that are developed in Java (Java based application)

(JVM) → Java Virtual machine ⇒ used for executing Java Programs (compilation)

* Jvm cannot be installed alone it comes with JDK & JRE

PATH & CLASSPATH → Both are Env. Variables

$$\begin{tikzcd} \text{Path} \arrow[r, "\text{finds}"] & \text{JRE} \arrow[r, "\text{finds classes}"] & {} \end{tikzcd}$$

(Binary Libraries)

Bin files (Ex: Java.exe, Javac.exe)

→ Path finds Location for files (Binary Libraries or Jvm)
→ Class path points to the classes you have developed so that Jvm can find them and load them

.class file

INHERITANCE

Aquiring all the methods & variables from parent class to child class

```
Class A        // Parent class / Super Class / Base Class
{
  ---
}
                 → keyword
Class B extends A    // Child Class / Sub class / derived Class
{
  ---
  ---
}
```

Single Inheritance

Inheritance
- Single
- Hybrid
- Hierarchy
- multiple

→ Interface is used for this

METHOD OVER-RIDING

Re-define the same method from parent class into child class.

```
class A
{
void m1() {
---
}
}
class B extends A
{
void m1()
{
---
}
}
```

Overloading	Overriding
=> no inheritance required	=> at least 2 class required (1 Parent and other child)
=> Definition should not be same (Parameters) (no. of Parameters, Order, Datatype	=> Definition should be same (no. of Parameters, Order, Datatype)

FINAL KEYWORD

```
final int a = 100;   // the value of variable is
                        constant (cannot change)

final void m1()      // method we cannot override
{                       in child class
}

final class Test     // class cannot be extended
{
}
```

INTERFACE → contains final & static variables, abstract method.

→ Blue print of class

```
interface Test
    {
    //variables
    //methods
    }
```

→ interface contains only static & final variables (by default)

→ in interface by default methods are public
→ in interface methods are by default abstract

Abstract : A method that have only definition but no implementation

```
Void m1();
```

$$\begin{tikzcd} \text{Class} \arrow[d, "\text{implements keyword}"] \\ \text{Interface} \end{tikzcd}$$

```
interface Test
  {
  Void m1();
  }
class Test2 implements Test
  {
   Void m1();
    {
     //implement the body
    }
```

→ we cannot instantiate interface

Webdriver driver = new webdriver() ✗

PACKAGES → Collection of classes

- User defined
  - Created by user
- Predefined / Built-in Packages
  - Java Packages
  - → import statement will be used for these Packages

ACCESS MODIFIERS

- Private
  - it can be accessed only within the class
  - (if method or variable is private)
- Default
  - If no modifiers is defined it is considered as default
  - accessible only within the package
- Protected
  - accessible within package and outside of Package with inheritance
- Public
  - Everywhere accessible

EXCEPTION HANDLING

An Event which will terminate program unexpectedly

- Checked Exceptions
  - Exceptions are ~~not~~ identified by JAVA Compiler
  - Ex: IOException, FileNotFoundException, etc
- Unchecked Exceptions
  - Exceptions which are not identified by JAVA Compiler
  - Ex:- ArithmeticException, ArrayIndexOutOfBoundException

String st = "$\overset{1234}{\text{RAM}}$";
Unchecked: int i = Integer.ParseInt(st);
Arithmetic Exception, Null pointerException, NumberFormatException, ArrayIndexOutofBoundException

Handling Exception
* *

```
try
{
// Specify the statement which causes Exception
}
Catch (Exception type)
{                          [e.get Message()]
// write the code  =>
}
finally  => optional - used for recovery methods
{
// Some code
}
```

→ Exception occurs, catch block handles, finally block also executes
→ Exception occurs, catch block not handles, finally block execute
→ Exception not occurs, catch block will ignore, finally block execute

*
NOTE: [Unchecked Exception Handling] ⇓ Try -- Catch block

[Checked Exception handling] ↓ try --- Catch block ⇓ can be applied at statement level; ↓ throws keyword ⇓ can be applied at method level

Array Limitations

→ Fixed in size → ArrayList → with ArrayList size can't be fixed.

→ Can hold same data type elements → object
⇓
using object array any type of data can be stored

Array List Declaration

ArrayList al = new ArrayList();
⇓
allows all data type elements

ArrayList <String> al = new ArrayList<String>();
⇓
allows only String type data elements

ArrayList <Integer> al = new ArrayList<Integer>();
⇓
allows only Integer type elements

al.add() → adding nos in ArrayList
al.size() → returns no. of elements
al.add(2, "value") → inserting value in middle
al.remove(value) → remove element
al.remove(index) → alter no. of elements

contains key & value ⇓ HASHMAP

[ArrayList & HashMap from video in detail]

HashMap <Integer, String> hm = new HashMap<Integer, String>();

hm.put(100, "Raj");

key & value combining called key value pair