Getters & Setters

In Java getters & setters are the methods used to access & modify the private fields of class respectively.

Getter methods :

→ Used to retrieve the value of a particular field.

→ Typically declared with the "public" access modifier & have a return type that matches type of attribute being accessed.

Setter methods :

→ Used to modify the value of a private field (or attribute.

Static Block

It will be executed at the time of classloading. Hence if we want to perform any activity at the time of class loading, we have to define that activity inside static block.

→ within a class, we can take any no. of static blocks.

* Encapsulation :

Is a mechanism that binds code, the data it manipulates into a single class.

→ while protecting both from outside & misuse. (Access control).

→ public, private, protected, default.

* Inheritance / Is-a Relationship :

→ Fundamental concept of OOP.

→ A class can inherit attributes, methods from another class.

→ Allow classes to be organized in hierarchical structure.

→ Higher level class ⟶ common behaviour
Lower level class ⟶ specialized behaviour.

Advantages
code reusability.

* super :———

- Super calls the constructor of parent class.
- used when method is overridden in subclass.
- Useful when field in subclass hides a field in super class.

* Polymorphism :———

If one thing existed in more than 1 form.

Advantage:

Flexibility to design application.

Types of polymorphism:

① Compile time | static | early binding.
② run time | dynamic | Late binding.

* Method Overloading	Method Overriding
① extending existed method function.	① Replacement for the existed method.
② static polymorphism.	② Dynamic polymorphism.
③ without inheritance can perform.	③ Inheritance is involved.
④ Same method name, diff parameters.	④ must have same method prototype in both parent & sub class, allowing modification in child class method.
⑤ Method hiding not possible	⑤ Method hiding is possible.
⑥ Can overload final methods.	⑥ Cannot Override final methods.
⑦ Constructor Overloading possible.	⑦ Constructor Overriding not possible.

* Method Signature :——

↓
Uniquely identifies a method within a class.

2-elements : Name & parameter list.

* Method Hiding:——

Overriding parent class static methods in child class.

- toString() method :——
  provides a String Representation of an object.
- Equals() method :——
  Compare the content / state of 2-objects.
- == operator :——
  reference comparison.
- Final modifier:——
  Applicable for classes, methods & variables.
  - class declared as final → can't extend that class.
  - method ——→ can't override that method.
  - variable ——→ can't reassign the value.

→ If Instance variable declared as final need to initialize as 3-types:

→ at the time of declaration.
→ at the instance block.
→ inside the constructor.

→ Static → 2 types.

⇒ At the time of declaration.
⇒ at the static block.

Abstract Modifier :——

Applicable for classes & methods but not for variables.

Abstract Method :——

→ Have only declaration, no Implementation.
→ Ends with semicolon ;

Advantage :-

by declaring advantage method in parent class we can provide guide lines to the child class.

Abstract class :

→ Instantiation not possible.

→ for any java class, if we are not allowed to create object, such type of class we declare with abstract modifier.

String :

→ Represents group of characters.

→ Space also considered as string.

Creating Strings :

→ By using new operator.

→ we can declare string type variable directly with its characters.

→ Can convert char type array to string object.

PACKAGES

Group of similar type of classes, interfaces and subpackages.

⇒ built in package (java.lang).

⇒ User defined package.

java.lang. package :

→ Automatically imported in java programs.

→ contains fundamentals like:

- Object class
- String class
- StringBuffer class.
- Wrapper class.

Import statement :

→ Using explicit import statement recommended.

2 types :

- Implicit import statement.
- Explicit.

- **Recursion in Java :——**

→ It is a programming Technique where a method call itself to solve a problem.

→ Useful when problem can be broken down into smaller sub-problems of same type.

characteristics

Base case : condition under which recursion ends.

Recursive case : part where method call itself.

- **Method Overloading :——**

In Java it is possible to define 2 (or) more methods within the same class & that share the same name, as long as their parameters are different.

- **Constructor Overloading :——**

In Java, a class contain more than 1 constructor & all these constructors having same name but different arguments.

- **Automatic promotion in overloading :——**

In Overloading, if compiler is unable to find the method with exact match, we won't get any compiler-time error immediately.

→ Compiler promotes argument to next level & checks all possible methods.

- **METHOD Vs CONSTRUCTOR :——**

Method	Constructor
• Set of instructions representing action of an entity.	• Initialization of object.
• Must provide return type.	• not provide return type.
• Allow access modifiers like static, final, abstract.	• 4 modifiers like public, private, protected, default.
• We can access at any part of the program.	• Executed normally at the time of object creation.
• Not mandatory to provide same class name as method name. (relevant to prg task).	• ~~executed~~ class name & constructor name must be same.
• Execute when we will access the methods.	

Advantages:

- Code Reusability
- Modularity
- Readability
- Encapsulation
- Debugging & Maintenance.

Disadvantages:

- Overhead
- Abstraction Complexity.

- **Can be written in 4 ways** : (diff parameters, diff return type)

① Without parameters & return type.

② With parameter & without returntype.

③ Without parameter & with return type.

④ With parameters, & with return type.

- **Instance Vs Static methods :**

	Static	Instance
– Belonging	class level	Instance level.
– Access	called on class	called on an instance of class.
– Accessibility	can access only, static members directly.	can access both static & instance members.
– Existence	exist as a single copy for the class.	Exit as multiple copies, one for each instance.
– Keyword Usage	static keyword	No static keyword.
– "this" keyword	cannot use this.	can use this.

→ this() constructor :—

- Used to call another constructor within the same-class.
- It must be 1st statement in constructor if used.
- Cannot use to this methods in 1 constructor.

→ this keyword :—

→ It is a reference variable that refers to the current object instance.

used in several contexts :—

- To refer to instance variable
- To invoke current class method.
- To pass an argument in the method.

→ Constructor chaining :—

- One constructor can call another constructor of the same class (or) its superclass.
- Allows for code reuse and helps in initializing object with different sets of parameters.

• Methods •

- It is a block of code that performs a specific task & can be called from other parts of the program.
- Essential for organizing code, promoting reusability & enhancing readability.

Considerations :—

- use meaningful names.
- choose appropriate types & no. of parameters.
- Decide whether method should return a value & appropriate return type.
- Define whether methods are public, private, protected, etc....
- Decide whether static (or) instance.

Importance of finalize() Method in Java :—

→ provided by the object class in java, which can be overriden to perform cleanup operations before an object is garbarge collected.

Constructor :———

Object creation is not enough, we should perform Initialization then only the object is in a position to provide the response properly.

⇒ When we creating an object some piece of the code will be executed automatically to perform Initialization of an object this piece of the code is nothing but constructor.

Rules :—

- Name of Constructor & name of class must be same.

→ Default Constructor :———

- For every class including abstract classes constructor concept is applicable.
- If we are not writing any constructor system will generate default constructor.

→ parameterized constructor :———

Initialize the objects of a class with specific values at the time of object creation.

→ Accepts one (or) more parameters.

→ Copy Constructor :———

→ Special type of Constructor.

→ Initializes a new object as a copy of an existing object of some class.

→ Accepts object of same class as parameters and intializes the new objects state with the values from existing objects.

- length vs length() method :——

→ length :
- It is a final variable applicable for Arrays.
- By using this, we can find length of Array.

→ length() method :
- final method applicable for strings.
- By using this we can find length of String.

Multidimensional Array :——

In Java, multi-Dimensional Arrays are implemented as Arrays of Arrays but in matrix form.

→ The advantage of this approach is to improve memory utilization.

OOP's

- class :——

→ Specification from group of objects. Each group representation is called class.

→ Have common properties & Behaviours.

→ Virtual Entity.

→ Virtual Encapsulation of properties & behaviours.

→ Generalization

→ Model (or) blue print for objects.

- Object :——

→ Individual Element in group of Elements having physical properties & Behaviours.

→ physical (or) Real Entity.

→ physical Encapsulation of the properties & Behaviours.

→ Specialization.

→ Instance of the class.

for :

- If we know no. of iterations in advance then we should use this.
- In entire for-loop the initialization section executes only once.

Break :

- Can use in 3-positions only.
  1. Inside switch
  2. Inside loop
  3. Inside label block.

continue :

To skip the current iteration, print the next statement. (pass current, continue remaining).

ARRAY

- Array is an index based collection of fixed number of homogeneous data elements.

Advantage :— We can represent multiple values in a single line.

→ Readability of the code will be improved.

Dis-advantage :—

Fixed in size, once we created we cannot increase (or) decrease the size.

Array Construction :—

Every array in java is an object. We can create by using new operator.

- Array Initialization :—

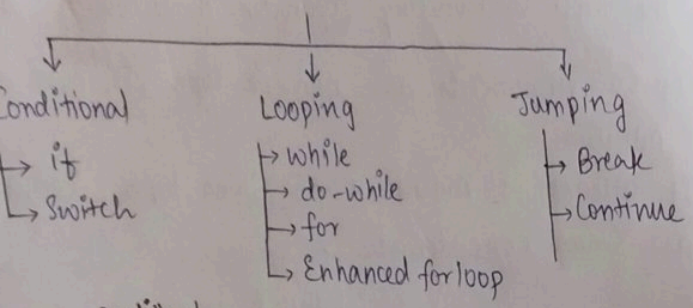
The every element is initialized with its default values automatically.

→ If we are not satisfied with default values then we can replace with our customized values.

Control Statements

⇓

Allows the flow of Execution.



Conditional.

If - Statement

- Bi-directional branching statement.

Types: if, if-else, else-if-ladder.

Note: else is not part of Java. It is optional.

Syntax: if (condition) {

.....

}

→ Condition should be boolean type.

Switch statement:

- multi-way selection method.
- We can use switch statement becoz it improves readability of the code.
- Curly braces mandatory.
- Every statement inside switch must be under some case (or) default.
  Independent statements not allowed.
- Every case label should be Compile time constant otherwise we will get errors.
- Every case label should be within the range of switch argument type.

Loops.

While

If we don't know no. of iterations in advance then we should go for while loop.

- It is also called Top tested loop (or) Conditional Control loop.

Do-while

↓

;-Semicolon mandatory

- It is bottom tested loop.
- Minimum no. of iterations is 1. Need to print the do statement at least once.

- Can print in 3-ways:
  1. Object reference (static method).
  2. In static context.
  3. Even with class name.

Local Variable

- Inside a method, constructor (or) block.
- created when method is called & destroyed when method ends.

Operators

↓ Symbol

performs Activity (or) Task.

Types:

- Arithmetic (+, −, %, /, *) (% → quotient, / → Reminder)
- Increment & Decrement (++, −−)
- Relational (<, <=, >, >=, ==, !=)
- Bitwise (OR, AND, E...OR (XOR))
- Logical
- Assignment (+=, −=, *=, /=, %=)
- Ternary / Conditional
- String Concatenation.

Note: In Java, if Integer is divided by zero, then it will show exception.

→ The most overloading operator in Java is "+".

- If 2-Arguments are numeric type, it (+) acts as Addition.
- If atleast 1-Argument is String type, then "+" acts as String concatenation.

Ex: a = a+b+c {a=10, b=20, c=30. String d = "codegnan";}

Sysout (a); → 60.

String ← d = a+b+c (x invalid) → Int.

d = a+b+c+d

Sysout (d) → 1120 Codegnan.

a = a+b+d;

Sys Out (d) → Invalid.

- ## Ternary Operator

Syntax: (condition) ? exp1 : exp2;

→ If condition is True, Exp 1 - evaluated.

→ If condition is false, Exp2 - evaluated.

Nested Ternary Operator.

Syntax: (condition) ? exp 1 : (condition) ? exp2 : (condition) ? exp 3 : (condition) ? exp4 : exp5;

JAVA

JAVA HISTORY

Origin - 1991
Developed by - James Gosling, At Sun MicroSystems.

Birth of Java - 1995
- Initially called "Oak".

Java Slogan - "Write Once, Run Anywhere".
(Java can run on any System with a JVM).

Java Features :

- Simple → Follows C, C++ Syntax
  → Memory allocation in Java is followed by JVM.
- Partial Object Oriented - becos it have primitive data types.
- Robust - we can handle any exceptions & errors.
- Secured - can only do compilation, not decompilation.
- Standard Library API - Supports rich library API's. Have all the packages.
- Platform Independent - bytecode can run on any platform with JVM.
- Strongly typed -
- Open Source -
- Distributed
- portable
- Garbage Collection.

Data types

(8 primitive datatypes)

Numerical

- Integral
  - byte (1)
  - Short (2)
  - Int (4)
  - long (8)
- Floating
  - float (4)
  - Double (8)

Non-Numerical

- boolean
- char (2)

Note: In C, the character range is 1-Byte but in Java, 2-bytes.

→ C & C++ asci code based values (0 to 265)
Java Unicode (more than 265). So it requires more memory.

Literal

Any constant value can be assigned to a variable is called Literal.

4 types.

- Binary
- Octal
- Decimal
- Hexa-Decimal

Base Value.

2
8
10
16 (x)
↓
(0 to 9 & a to f)

Basic code instructions:

- Starting letter of class name must be Capital.
- Filename should be saved with class name.
- IDE follows in line braces, those are recommended

Identifiers

- A name in program is called Identifier in Java.
- Whether the name maybe classname, method name (or) variable name (or) label name.

Rules to declare Identifiers:

- a-z, A-Z, 0-9, $ → only allowed characters
- should not start with digits.
- Spaces are not considered.
- Case sensitive. Can use both Uppercase & Lower case letters.
- No length & limit but 15 characters recommended.
- predefined / Reserved words should not be taken.
- pre-defined classes & interfaces can be taken.

modifiers related to package :

- default → same package
- public → Anywhere.
- private → within this same class
- protected → same package.

Interface :

Specification to a set of classes,

Ex: TV remote (model)

Declaration :

methods → declared as public & abstract by default.

variables → public, static, final.

~~Ext~~

Extends Vs Implement :

- A class can extend one class at a time.
- A class can implement any no. of interfaces.
- A class can extend 1 class & implement any no. of interfaces.
- An interface can extend any no. of interfaces.

marker Interface :

If an interface does not contain any methods & by implementing that interface our objects will get some ability.

Ex: Serializable, clonable, random access.

Adapter pattern :

Allows to incompatible interfaces to work together. It acts as a bridge b/w two incompatible interfaces.

Exception Handling

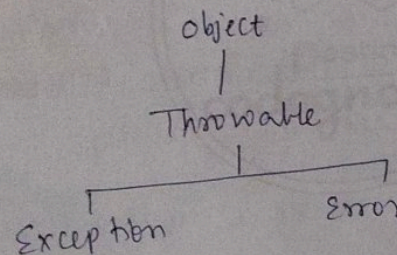
Exception :—

An unwanted & unexpected event that distributes normal flow of program.

Exception Handling :—

Defining an alternative way to continue rest of the program.

Exception Hierarchy



Errors : → Caused by system resource
→ Non-recoverable.

Types of Exceptions (3 types)

↳ checked
↳ un-checked
↳

→ **checked Exception :—**
checked at compile time.

→ **Unchecked Exception :—**
that happen directly in runtime.
without checking in compile time.

characteristics of Exception :—

- must be handled (or) declared using throws
- checked at compile time.
- Extend exception class but not run-time.
- Represent recoverable condition.

common checked exceptions :

→ class not found
→ Interrupted
→ clone not supported
→ no such method

Runtime Exception :-

- Base class for unchecked exception.
- Indicate programming errors that could be avoided.
- No compile time checking.

Abstract class	Interface
— Abstract keyword	— interface keyword
— abstract & concrete methods	— only abstract methods.
— Instance variable	— constant only
— have constructors	— No constructors
— No Multiple Inheri-tance	— Supports Multiple Inheritance
— partial Implementation	— ~~full~~ No implementation
— Can't be Instantiated	— Can't be Instantiated.

Interview Questions

- Code compiled into executable code.
- This can't run across all platforms.

When javac compiles a java program it generates an executable file called class file.

Class file have byte code. bytecodes are interpreted only by JVM. JVM's made available by all platforms by Sun-Microsystems.

Diff btw C++ / Java :-

C++	Java
• Have pointers ~~pla~~.	• No pointers.
• platform dependent	• Platform independent.
• no garbage collection	• garbage collection.
• ~~Not~~ Support Multi-Threading.	• multi-threading.

JIT (Just in time compiler) :-

- Compiles byte code into executable code.
- part of JVM.
- Cannot convert complete code, converts when it is needed.

Interview Questions

- Code compiled into executable code.
- This can't run across all platforms.

When javac compiles a java program it generates an executable file called class file.

class file have byte code. bytecodes are interpreted only by JVM. JVM's made available by all platforms by Sun-Microsystems.

Diff btw C++ / Java :——

C++	Java
• Have pointers ~~pta~~.	• No pointers.
• platform dependent	• Platform Independent.
• no garbage collection	• Garbage collection.
• Not Support Multi-Threading.	• Multi-Threading.

JIT (Just in time compiler) :——

- Compiles byte code into executable code.
- part of JVM.
- Cannot convert complete code, Converts when it is needed.

→ Is a relationship

Known → Inheritance

"extends" keyword

Reusability of Code

Has a Relationship

Composition / Aggregation

"new" keyword.

Reusability of code

Instance Of operator :—

- Used to test object is of which type.
- returns true if reference expression is subtype of destination type.
- returns false if reference expression is null.

• Null :—

↳ when a reference variable doesn't point at anywhere to any value. it is assigned null.

For top class, only 2-access modifiers are allowed.

public → visible everywhere

default → same package.

→ Object cloning :—

↓

way to create exact copy of an object.

java.lang.cloneable interface must be implemented if not will get clone not supported error.

→ we can't support package statement after import statement. It must be first statement in source file.

→ Access Specifiers & Modifiers :—

↓

In C++, we have these. But in java we have access modifiers & Non-access.

↓ ↓

(public, private, protected, default) (static, final, abstract).

* Access Modifiers :—

① class.

public

- same class
- same package → subclass / non-sub class.
- different package → sub-class / non-subclass

Default → Same

② Methods

→ public → same as public class
→ Default → No different package.
→ protected → not accessed on diff package non-subclass.
→ private → accessed only in same class.

③ Variables

→ same public class
default → no diff package
protected → No diff package sub-class
private → only in that class.

→ Overloading

→ Same method, diff parameters ~~temp~~
↳ Compile time.
→ Same method name & parameters
↳ Runtime &.

→ Exception Handling Keywords :—

- by
- catch
- Finally
- Throws.

try :—

- contains code that throws an exception.
- putting protecting shield for risky code.
- Can not contain alone. we must write catch (or) finally.
- Have multiple statements.
- exception stops at line where exception occurs.

- How to create constants in Java ?
  - fixed values.
  - can't be changed during execution of Program.
- Create constants in Java using Final keyword.

- >> & >>> ?
  - Unsigned shift operator, used to shift right.
  - Right shift operator, shifts all the bits in a value to right to a specific no. of times.

Ex: int a=15;
a= a>>3;

- Java Coding Standards/ Java Coding Conventions:

For: ① classes
- class name
  - capital letter
  - Noun
  - If multiple words the 1st letter of inner word must be capital.

② Interface
- should start with uppercase
- should be (Adjectives)
- Ex: Runnable, Serializable, marker, clonable.

③ Methods
- should start with small letters.
- Usually (verbs)
- Every inner word 1st letter capital.
- Ex: toString → Combination of verb & noun.
  - ↳ getMyName().

④ Variables
- should start with small letters.
- (Noun)
- Ex: String, value, empName.

⑤ Constants
- created using static & final keyword
- Contains only uppercase.
- If 2-words, then supported/ seperated by underscore.
- Ex: MAX_VALUE.

Byte code :—

→ machine independent language.

→ Contains set of instructions which are executed only by JVM.

Why main method is public, static & void :—

public → can be used outside class.

static → sometimes method can be called without the object requirement, then we declare that as static.

void → used return value. when method doesn't.

main method in Java :

main → starting point of execution.

String args[] → array of String objects we need to pass from command line arguments.

ASCII code :

→ American Standard code for Information Interchange.

→ Character range → 0 to 255.

→ Supports only English.

UNI code :

To support all languages in the world.

16 bits & range → 0 to 65,535.

→ Java uses ASCII code for all input elements except for string, identifiers, comments.

— character constant :

- enclosed in single quote.
- single digit (or) character.

— String Constant :

- enclosed in double quotes.
- Collection of characters.

Characteristics of Unchecked Exceptions:—

→ No compile time checking
→ Extend run-time exception class.
→ Usually indicate programming errors.
→ Can be avoided with proper coding.

Common unchecked Exception:—

- Null pointer
- Array Index out of bound
- IllegalArgument.
- Number Format
- classCast.

* Partially vs Fully checked:—

Fully checked only if all its child classes are also checked.

→ Key classes in Exception Hierarchy Handling:—

- Throwable class
- Error class
- Exception class.

① Throwable class:

- Root class for all Exceptions & Errors.
- Contains methods like getmessage(), printStackTrace(), getcause().
- All exceptions must extend throwable.

② Error class:

- Represent serious level programs.
- Should not caught by applications.
- Indicate consitions.

③ Exception class:

- Base class for un-expected exception.
- Divided into checked, unchecked.
- Application should handle these.

Type casting :

Converting one datatype into other.

- **Implicit :**
  - Convert smaller Type → larger type.
  - Convert Subclass → super class
    (Done Automatically by Java).

- **Explicit :**
  - Convert larger type → Smaller Type
  - Converting, Superclass → Subclass
    (must be done manually :).

Variable

Java variables used to store information / value in that, can be manipulated & accessed with in a program.

Division - 1

- primitive Type
  - value assigned to a primitive datatype
  - int x = 10 ;
- Referenced Type
  - referenced variables can be used to refer objects.
  - Student s = new Student();
    (object ↓ reference)

Division - 2

Based on purpose and position divided into 3 types.

- Instance Variable
- Static variable
- Local variable.

Instance Variable

- Inside class, outside method.
- JVM will provide default values.
- Instance variable created at the time of object creation & destroyed when object is deleted.
- Instance variable | Reference variable | Object variable | Object referenced Type.

Static variable

- Inside class, outside method with the "static" keyword.
- JVM will provide default values.
- Created at the time of class loading.
- Value of variable not varied from object to object.
- will be stored in method Area.

Basic code instructions:

- Starting letter of class name must be Capital.
- Filename should be saved with class name.
- IDE follows in line braces, those are recommended

Identifiers

- A name in program is called Identifier in Java.
- Whether the name maybe classname, method name (or) variable name (or) label name.

Rules to declare Identifiers:

- a-z, A-Z, 0-9, $ → only allowed characters
- should not start with digits.
- Spaces are not considered.
- Case sensitive. Can use both Uppercase & Lower case letters.
- No length & limit but 15 characters recommended.
- predefined / Reserved words should not be taken.
- pre-defined classes & interfaces can be taken.

Data types

(8 primitive datatypes)

Numerical

- Integral
  - byte (1)
  - Short (2)
  - Int (4)
  - long (8)
- Floating
  - Float (4)
  - Double (8)

Non-Numerical

- boolean
- char (2)

Note: In C, the character range is 1-Byte but in Java, 2-bytes.

→ C & C++ asci code based values (0 to 265)
Java Unicode (more than 265). So it requires more memory.

Literal

Any constant value can be assigned to a variable is called Literal.

4 types	Base value
• Binary	2
• octal	8
• Decimal	10
• Hexa-Decimal	16 (x)

(0 to 9 & a to f)

JAVA

JAVA HISTORY

Origin - 1991

Developed by - James Gosling, At Sun MicroSystems.

Birth of Java - 1995

- Initially called "Oak".

Java Slogan - "Write Once, Run Anywhere".

(Java can run on any System with a JVM).

Java Features :

- Simple → Follows C, C++ Syntax
  → Memory allocation in Java is followed by JVM.
- Partial Object Oriented - beco3 it have primitive data types.
- Robust - we can handle any exceptions & errors.
- Secured - can only do compilation, not decompil-ation.
- Standard Library API - Supports rich library APIs. Have all the packages.
- Platform Independent - bytecode can run on any platform with JVM.
- Strongly typed -
- Open Source -
- Distributed
- portable
- Garbage Collection.

*Programming Language : means set of instructions to interact with machine.

—3 types of PL's —

① Low level – machine Understandable – 0/1 – Binary.

② Mid level – Assembly level – cannot develop large applica-tions like amazon, flipkart.
(Combination highlevel & lowlevel language).

③ High level – Human Understandable language.
Human can implement or write own logic in prg.
Ex. Java, python, .net, etc.....

↳ History:

1995 – Sun microsystem – James Gosling – old name (Oak).
– inspired from a coffee bean in Indonesia.

2011 – Oracle Coroperation Undertaken.

Industry standard version – 1.8 (or) 21.

Lastest Java version – 24.

JAVA FULLSTACK COURSE.

* FULLSTACK → FRONTEND + BACKEND.
  (User Interface).

* Static Application → same content (Wikipedia).
  Dynamic Applications → User based content.

* Java can be used to develop both static & Dynamic Application.

* S.E. – Standard Edition (Stand alone Appli – (all task runs on same device without client/server).
  CUI - Command line Interface
  GUI - Graphical User Interfaces
  E.E – Enterprise Edition. (Distributed App) (Server based).
  M.E – Micro / mobile Edition.

* JAVA → core Java, advance Java, frameworks.
  - JDBC
  - servlets
  - Jsp's
  - Hibernate
  - Spring
  - SpringBoot.
  - Microservers.

JDBC – Combines Java Application with Database.

Languages:

① Machine / Low level language
② mid / Symbolic language / Assembly level Language.
③ Programming Language / High Level Language.