CHAPTER-1 SUMMARY

➔ ***INTRODUCTION TO PROGRAMMING***

- Machine/Binary Level Lang
- Assembly Level Lang
- High Level Lang

➔ ***JAVA INTRODUCTION***

- JAVA is a high-level , secure , Object oriented language.
- Created in 1991 by Green Team under Sun micro systems.
- Release first version java 1.0 in 1996
- Initially called with a name **OAK.**

➔ ***JAVA TERMINOLOGIES***.

- **JVM** :
  - acts as a run-time engine to run Java applications.
  - Calls main() methods (always look for specific method signature PSVM)
  - JVM is a part of JRE
  - **JVM is platform dependent.**

- **JDK**:
  - A complete Java development kit that includes everything including compiler, Java Runtime Environment (JRE), java debuggers, java docs.
  - **JDK is platform dependent**.

- **JRE**:
  - The Java Runtime Environment, or JRE, is a software layer that runs on top of a computer's operating system software
  - It provides the class libraries and other resources that a specific Java program needs to run.
  - JDK includes JRE
  - **JRE is platform dependent**.

- **Bytecode:**
  - The **javac** compiler of JDK compiles the java source code into bytecode so that it can be executed by JVM.
  - It is saved as a **.class** file by the compiler.
  - Byte Code can be defined as an **intermediate code** generated by the compiler after the compilation of source code.
  - Byte code makes **Java a platform-independent language.**

➔ ***JAVA FEATURES***

- Platform Independent , OOP's, Simple , Robust,
- Distributed , Multithreading,Portable , WORA.

➔ ***TWO PHASES IN A JAVA PROGRAM***

- Compilation phase
  - javac - compiler
  - Ex: **javac fileName.java**
  -
- Execution phase.
  - java - runtime tool
  - Ex: **java bytecodeName**

OBSERVATIONS

- **javac** (Java Compiler) will take care all java syntax checking and generates BYTECODE
- **JVM** will load byte code into the memory and start execution from the **main()** method.
  - order of public static can be of different ways
    - public static
    - static public
    - String[] args
    - String []args
    - String args[]
    - String… args
  - **static** keyword tells us that this method is accessible without instantiating the class.
  - System.in & System.out are Standard Input & Output Streams.
  - System is a class belonging to java.lang and **in & out** objects belong to PrintStream class.
  - String []args : used for reading the command line args/values while running the program.

- JDK(JVM+JRE) is platform dependent because each OS will have different instruction sets.

- **ByteCode is PLATFORM INDEPENDENT** , which makes Java as a platform independent language.

- We can compile a **class without a main method()** , but we can't run , it will raise a **Runtime Exception**.

- When a **class is public , fileName must be the same as className**.

- When **class is not public**.
  - **compile** with fileName
  - **run** with className
  - by default it will generate a byte with className.

- It is Possible to have multiple classes in one file , it will create multiple .class files.
- whenever we have multiple .class files while running , execute with individual names.

  java A
  java B

- Mark & Sweep is an algorithm used by garbage collector to delete unused variables.

- HashCode : Unique identification number allocated to objs by **JVM.**
- JAVA is case sensitive and follows UNICODE character representation.

- **VARIABLES**
  - containers for storing data values.
  - **type variableName = value;**
  - **_ , $** Only two special symbols/characters which are allowed in naming variables.
  - Final Variable : unchangeable and read-only , used for declaring constants.
  - Types:
    - **Local Variables** -> declared in main() method ,No default values.
    - **Instance Variables**: declared in class outside method/constructor/block , it has default values.
    - **Static Variables:** class variables declared with **static** keyword.

- **DATE TYPES**

  - **Defines the types of data to hold in variables.**
  - Two Groups
    - Primitive: **byte, short, int, long, float, double, boolean and char**
    - Non-Primitive: **String, Arrays and Classes.**

  - By default all **precious or fractional numbers** are **double** datatype in JAVA
  - By default all **decimals numbers are int** type.

  - To Declare float values use **f or F** at end of value.
    - Ex: float salary = 10.2323f;

  - To Declare Long type value use **l or L** at end of value.
    - Ex: long num = 10000000000L;

  - Each and Every data type will have its own size/range
    - **Formula**
      - -2^(n-1) to +2^(n-1)-1
  - For Declaring other number system formats we use,
    - Binary : 0B
    - Octal : 0o
    - Hex-decimal: 0x

❖ TYPE CASTING

- ➢ **casting**/converting one datatype to another datatype.
- ➢ **Widening** Casting (automatically)
  - Done by JVM
  - converting a smaller type to a larger type size
  - byte -> short -> char -> int -> long -> float -> double
  - No loss of data.

- ➢ **Narrowing** Casting (manually)
  - Done by programmer/user.
  - converting a larger type to a smaller size type
  - double -> float -> long -> int -> char -> short -> byte.
  - Possibility for loss of data.

- ➢ **Type Promotion**.
  - Whenever after applying some operation if the value is not fitted into the type , it will assign to next higher type.
    - Ex: byte a=100,b=35;
      byte c = a+b;

❖ OPERATORS

- ➢ Operators are used to perform operations on variables and values.
- ➢ Types.
  - 1.Arithmetic operators
    - \+ - / * ++ –
  - 2.Assignment operators
    - =
  - 3.Comparison operators
    - == != < <= >=
    - Will always give true or false.
  - 4.Logical operators
    - && || !
    - Will always give true or false.
  - 5.Bitwise operators
    - & | ^ ~
  - Ternary operators.
    - **variable = Expression1 ? Expression2: Expression3**
    - Always number of ?’s should equal to number of :’s
  - Shift Operator
    - << n*2^bits
    - >> n/2^bits
    - >>>

❖ PRECEDENCE & ASSOCIATIVITY

- The operator precedence represents how two exps are bind together.
- In an expression, it determines the grouping of operators with operands and decides how an expression will evaluate.
- The operators having higher precedence are evaluated first
- follow associativity if an exp has more than two operators of the same precedence.
- In such a case, an exp can be solved either left-to-right or right-to-left

❖ INPUT & OUTPUT

➢ *Scanner class*

- allows the user to take input from the console
- belongs to java.util package
- used to read the input of primitive types like int, double, long, short, float, and byte.
- nextInt(), nextFloat() , nextDouble() , nextByte() , nextLong() , next() ,nextLine()

- **Scanner obj = new Scanner(System.in);**

➢ *Buffered Reader*

- used to read a sequence of characters
- InputStreamReader() :
  - converts the input stream of bytes into a stream of characters.

- **BufferedReader bfn = new BufferedReader(new InputStreamReader(System.in));**

- Integer.parseInt() , Float.parseFloat() , Double.parseDouble()
- Integer, Float , Double are wrapper classes.

➢ OUTPUT

- **System.out.print(parameter);**
  - used to display a text on the console
  - prints the text on the console and the cursor remains at the end of the text at the console.

- **System.out.println(parameter);**
  - prints the text on the console and the cursor moves to the start of the next line at the console
  - The next printing takes place from the next line.
- **System.out.printf(parameter);**
  - Easiest of all methods as this is similar to printf in C.
  - Uses format specifiers like %d %s %t %n %x %o

- **STRINGS**
  - collection of characters(letters,symbols,other lang letters) surrounded by double quotes
  - Non-Primitive Type belongs to **java.lang.String**
  - **CREATION**
    - **Using literals**
      - String s1="Welcome";
      - Creates in **string constant pool**
      - Each time we create a string literal, the JVM checks the "string constant pool" first.
      - If the string already exists in the pool, a reference to the pooled instance is returned.
      - If the string doesn't exist in the pool, a new string instance is created and placed in the pool.
    - **Using new operator**
      - String s=new String("Welcome");
      - Created inside **Heap Memory location.**
  - **CONCATENATION.**
    - \+ operator can be used between strings to combine them
    - str + str —> str , str + int —> str
    - str + float —> str
  - **METHODS**
    - length() , toUpperCase(), toLowerCase() , charAt() , isEmpty() , indexOf()
    - trim() , equals() , hashCode() , replace() , startsWidth() , endsWidth()
    - valueOf() , toString() , compareTo() , split() , matches()

- **StringBuffer**
  - StringBuffer represents growable and writable character sequences.
  - StringBuffer s = new StringBuffer("GeeksforGeeks");
  - Is **synchronized** and Thread Safe.
  - Methods: append() , insert() , replace() ,reverse() , delete()
  - Created inside Heap Memory location.

- **StringBuilder**
  - **represents a mutable sequence of character**
  - non-synchronized and not a Thread Safe.
  - It’s more efficient than StringBuffer.
  - StringBuilder builder=new StringBuilder("hello");
  - Created inside Heap Memory location.